НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

Фаховий молодший бакалавр
(освітньо-професійний ступінь)

на тему: Розробка програми для формування звітів у MS Excel з використанням мови Java.

Виконав студент IV курсу, групи ***ОК- 42***
ОПП *«Обслуговування комп'ютерних систем та мереж»*
Спеціальності *123 Комп'ютерна інженерія*
Пішхавка Віталій Вікторович
(прізвище, ім'я по батькові)

Керівник _____
(підпис) (ім'я прізвище)

Нормоконтролер *Любомира Кужій*
(підпис) (ім'я прізвище)

Рецензент *Андрій Селемонавічус*
(підпис) (ім'я прізвище)

Голова ЕК *Олег Гіщак*
(підпис) (ім'я прізвище)

Члени ЕК *Любомира Кужій*
(підпис) (ім'я прізвище)

Андрій Селемонавічус
(підпис) (ім'я прізвище)

Дипломний проєкт захищений в ЕК « __ » _____2025 р.

з оцінкою « _____ »

Львів 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ФАХОВИЙ КОЛЕДЖ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Циклова комісія	***Комп'ютерних систем і мереж***
Освітньо-професійний ступінь	***Фаховий молодший бакалавр***
Освітньо-професійна програма	***Обслуговування комп'ютерних систем та мереж***
Спеціальність	***123 Комп'ютерна інженерія***

ЗАТВЕРДЖУЮ
Завідувач відділення
«Комп'ютерних систем і мереж»
_____ Володимир СТАХІВ
«____» _____ 2025 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Пішхавці Віталію Вікторовичу
(прізвище, ім'я та по батькові)

1. Тема проєкту Розробка програми для формування звітів у MS Excel використанням мови Java.

керівник проєкту *Селемонавічус Андрій Альвідасович*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом директора від «*20*» *березня 2025* року № *20 - ст*

2. Строк подання студентом проєкту «*10*» *червня* 2025 року

3. Вихідні дані до проєкту
   - формувати звіти у MS Excel;
   - в якості мови програмування використати Java;
   - в якості API використати бібліотеку Apache POI

4. Зміст розрахунково-пояснювальної записки

4.1 ОГЛЯД ТЕХНОЛОГІЇ APACHE POI
4.2 ОСНОВНІ КЛАСИ APACHE POI
4.3 РОЗРОБКА ПРОГРАМ ДЛЯ ФОРМУВАННЯ ЗВІТІВ У MS EXCEL
4.4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ
4.5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

5. Перелік графічного матеріалу

5.1.	*Структура бібліотеки POI*	
5.2.	*Розробка програми для генерації звітів*	
5.3.	*Приклади роботи розробленої програми*	
5.4.	*Кошторис витрат на розробку проектного рішення*	

6 Консультанти розділів проєкту

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		Завдання видав	Завдання отримав
Техніко-економічне обґрунтовування	*Тетяна Підкуймуха*		
Охорона праці та безпека життєдіяльності	*Роман Томків*		

7. Дата видачі завдання « *01*»*квітня 2025* року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту	Термін виконання	Примітка
1	Огляд технології Apache POI	10.04.25	
2	Опис основних класів бібліотеки Apache POI	20.04.25	
3	Розробка програм для формування звітів	10.05.25	
4	Питання охорони праці і техніки безпеки	25.05.25	
5	Техніко – економічне обґрунтування проекту	10.06.25	
6			
7			
8			

Студент _____ Віталій Пішхавка
(підпис) (ім'я, прізвище)

Керівник проєкту _____ Андрій Селемонавічус
(підпис) (ім'я, прізвище)

РЕФЕРАТ

Текстова частина дипломного проекту: 67с., 7 рис., 3 табл., 12 джерел, 4 демонстраційних аркуші.

Об’єкт дослідження – Програмна реалізація звітів MS Excel засобами Java

Мета роботи – створити програму що реалізує звіти Excel за допомогою бібліотеки Apache POI.

Результати роботи дозволили створити навчальні матеріали по вивченню технології програмування додатків MS Office.

Галузь використання – інформаційні системи

APACHE POI, Microsoft Excel, API, Java, HSSF, XSSF, БАЗА ДАНИХ, FOLDERDIALOG, CHANNELDIALOG

ЗМІСТ

ВСТУП

Ми живемо в столітті інфокомунікаційних систем та технологій, де неможливо уявити жодну організацію без сучасної обчислювальної техніки. Використаня баз даних робить процес обліку та збереження даних простим та надійним. Для спрощення пошуку потрібних даних використовуються так звані зведені відомості, яки призначені для швидкої вибірки даних запевними параметрами. Зведені відомості дозволяють робити аналіз результуючих даних, звертати увагу на проблемні моменти та вчасно реагувати на проблеми.

Якщо подібні вибірки даних потрібно робити регулярно, наприклад кожен день, то стає питання про автоматизацію процесу створення звітів. Якщо дані знаходяться у таблицях Excel то є можливість запрограмувати процес створення звітів тим самим прискорити час підготовки звіту.

В якості мови програмування у проекті було запропоновано використати об’єктну мову програмування Java, яка дозволяє розробляти велики корпоративні додатки як під операційну систему Windows так і Android. Для взаємодії з офісними програмами для Java існує спеціально розроблена бібліотека Appach POI, що має набір класів та інтерфейсів для роботи з офісними програмами.

Метою даного проекту є програмна реалізація створення звітів у середовищі Excel за допомогою мови програмування Java.

1 ОГЛЯД ТЕХНОЛОГІЇ APACHE POI

1.1 Компоненти бібліотеки apache poi

Часто для створення звітів у форматі файлів Microsoft Excel потрібний програмний додаток. Іноді навіть очікується, що додаток одержить файли Excel у якості вхідних даних. Наприклад, додаток, розроблений для фінансового відділу компанії, повинен буде генерувати всі свої результати в Excel.

Будь-який програміст на Java, що бажає створювати файли MS Office у якості вихідних даних, повинен використовувати для цього визначений API тільки для читання.

Apache POI — це популярний API, який дозволяє програмістам створювати, змінювати й відображати файли MS Office за допомогою програм Java. Це бібліотека з відкритим вихідним кодом, розроблена й розповсюджувана Apache Software Foundation для розробки або зміни файлів Microsoft Office з використанням програми Java. Він містить класи й методи для декодування даних, що вводяться користувачем, або файлу в документи MS Office.

Компоненти Apache POI

Apache POI містить класи й методи для роботи з усіма складеними документами OLE2 MS Office. Список компонентів цього API наведений нижче.

POIFS (Файлова система реалізації) — цей компонент є основним чинником усіх інших елементів POI. Він використовується для явного читання різних файлів.

HSSF (формат електронної таблиці) — використовується для читання й запису у форматі xls файлів MS-excel.

XSSF (XML Spreadsheet Format) — використовується для формату файлів xlsx Ms-excel.

HPSF (формат набору властивостей) — використовується для витягу наборів властивостей з файлів MS-office.

HWPF (формат текстового процесора) — використовується для читання й запису файлів розширень doc MS-word.

XWPF (Xml-Формат текстового процесора) — використовується для читання й запису файлів розширення docx в MS-word.

HSLF (формат макета слайда) — використовується для читання, створення й редагування презентацій PowerPoint.

HDGF (формат Horrible Diagram) — містить класи й методи для двійкових файлів MS-visio.

HPBF (формат Horrible Publisher) — використовується для читання й запису файлів MS-publisher .

У проекті ми використовуємо роботу з файлами Excel з використанням Java. Тому обговорення обмежується компонентами HSSF і XSSF.

Більш старі версії POI підтримують двійкові формати файлів, такі як doc, xls, ppt і т. д. Версія 3.5 і вище, POI підтримує формати файлів OOXML MS-office, такі як docx, xlsx, pptx і т. д.

Як і в Apache POI, існують інші бібліотеки, надавані різними постачальниками для генерації файлів Excel. До них відносяться Aspose для Java від Aspose, JXL від Commons Libraries і Jexcel від Team Dev.

1.2 Apache POI — API Java Excel

Розглянемо деяких різновиди Java Excel API і їх функції. Є багато постачальників, які надають API, пов'язані з Java Excel; розглянемо деякі з них:

Aspose Cells для Java

Aspose Cells для Java — це ліцензований API Java Excel, розроблений і розповсюджуваний постачальником Aspose. Остання версія цього API — 8.1.2, випущена в липні 2014 року. Це багатий і складний API (комбінація простих

класів Java і класів AWT) для розробки компонента Excel, який може читати, писати й маніпулювати електронними таблицями.

Загальні застосування цього API наступні:

Звіти Excel, створювати динамічні звіти Excel

Високоякісний рендеринг і друк Excel

Імпорт і експорт даних з таблиць Excel

Створювати, редагувати й конвертувати електронні таблиці

JXL

JXL — це сторонній фреймворк, розроблений для Selenium, який підтримує автоматизацію на основі даних у веб-браузерах (автоматичне відновлення даних у веб-браузерах). Однак він також використовується в якості загальної бібліотеки підтримки для API Jexcel, оскільки він має базові функції для створення, читання й запису електронних таблиць.

Основні функції полягають у наступному —

Генерація файлів Excel

Імпорт даних з робочих книг і електронних таблиць

Одержати загальну кількість рядків і стовпців

JXL підтримує тільки формат файлу .xls і не може обробляти більші обсяги даних.

Jexcel

Jexcel — це чисто ліцензований API, надаваний Team Dev. Використовуючи це, програмісти можуть легко читати, писати, відображати й змінювати книги Excel у форматах.xls і .xlsx . Цей API може бути легко вбудований в Java Swing і AWT. Остання версія цього API — Jexcel-2.6.12, випущена в 2009 році.

Основні характеристики полягають у наступному —

Автоматизувати додаток Excel, робочі зошити, електронні таблиці і т. д.

Вбудовувати книги в додаток Java Swing як звичайний компонент Swing

Додає слухачів подій у робочі книги й таблиці

Додає оброблювачі подій, щоб обробляти поведінка робочої книги й електронних таблиць

Apache POI — це бібліотека з відкритим вихідним кодом, надана Apache Software Foundation. Більшість розроблювачів малих і середніх додатків сильно залежать від Apache POI (HSSF + XSSF). Він підтримує всі основні функції бібліотек Excel (рис. 1.1); однак рендеринг і витяг тексту є його основними характеристиками.

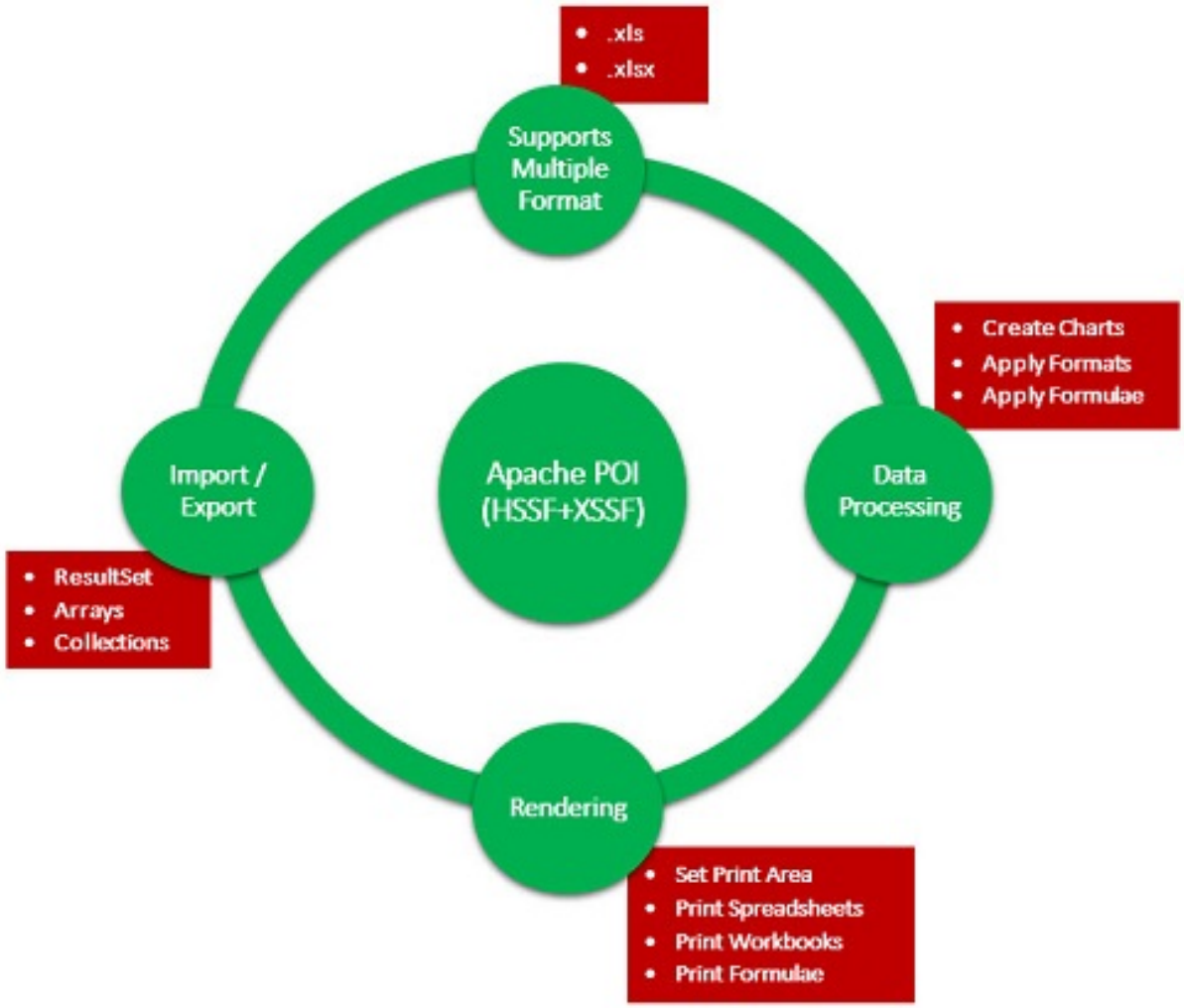


Рисунок 1.1 Функції Excel, які підтримує бібліотека POI

Розгянемо настроювання Apache POI у системах на базі Windows і Linux. Apache POI можна легко встановити й інтегрувати в поточне середовище Java, виконавши кілька простих кроків без яких-небудь складних процедур настроювання. Користувачу потрібні права адміністратора при установці.

Опишемо етапи установки Apache POI.

Крок 1. Перевірте вашу установку Java

Насамперед, вам необхідно встановити Java Software Development Kit (SDK) у вашій системі. Щоб переконатися в цьому, виконаєте кожну із двох команд залежно від платформи, на якій ви працюєте.

Якщо установка Java була виконана правильно, то на ній відобразиться поточна версія й специфікація вашої установки Java.

Якщо у вас немає Java SDK, завантажите його поточну версію з https://www.oracle.com/technetwork/java/javase/downloads/index.html і встановите його.

Крок 2. Встановіть середовище Java

Установите змінне середовища JAVA_HOME, щоб вона вказувала на місце розташування базової директорії, де встановлена java на вашім комп'ютері. Наприклад,

Windows

Установите JAVA_HOME в C: \ Programfiles \ java \ jdk1.7.0_60

Linux

Експорт JAVA_HOME = / usr / local / java-current

Додайте повний шлях розташування компілятора Java до системного шляху.

Платформа й опис
Windows Додайте рядок «C: \ Program Files \ Java \ jdk1.7.0_60 \ bin» у кінець системної змінної PATH.
Linux Експорт PATH = $ PATH: $ JAVA_HOME / bin /

Windows

Додайте рядок «C: \ Program Files \ Java \ jdk1.7.0_60 \ bin» у кінець системної змінної PATH.

Linux

Експорт PATH = $ PATH: $ JAVA_HOME / bin /

Виконаєте команду java -version з командного рядка, як описано вище.

Крок 3: Встановіть Apache POI Library

Завантажите останню версію Apache POI за адресу https://poi.apache.org/download.html і разархивируйте його вміст у папку, з якої необхідні бібліотеки можуть бути пов'язані з вашою програмою Java. Припустимо, що файли зібрані в папці на диску C.

На наступних зображеннях показані каталоги й файлова структура усередині завантаженої папки (рис. 1.2.).

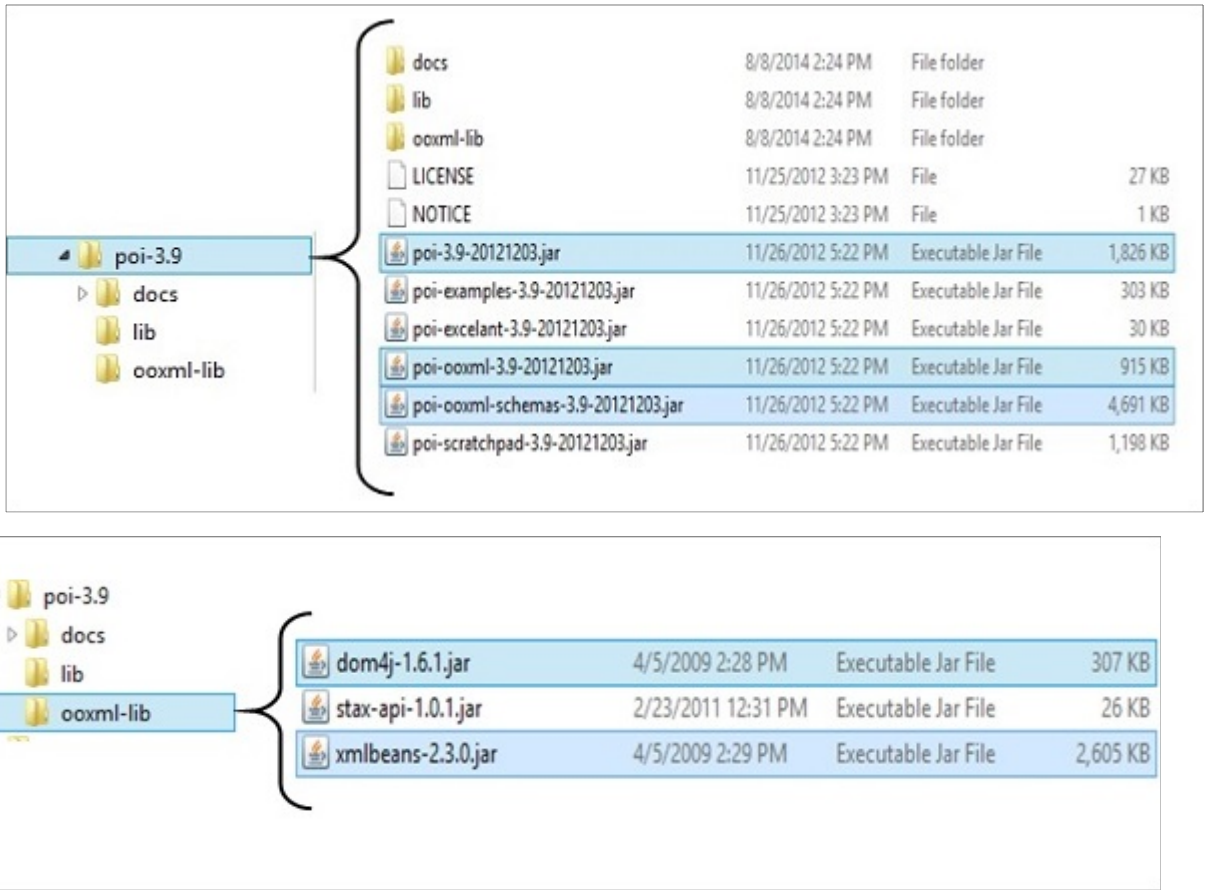


Рисунок 1.2 – Каталоги й файлова структура папки POI

Додайте повний шлях до п'ятьох jar, як показано на зображенні вище, до CLASSPATH.

Платформа й опис
Windows Додайте наступні рядки в кінець користувацької змінної CLASSPATH — «C: \ poi-3,9 \ poi -3.9-20121203.jar;» «C: \ poi -3,9 \ poi -OOXML-3.9-20121203.jar;» «C: \ poi -3,9 \ ooxml-схеми-3.9-20121203.jar;» «C: \ poi -3,9 \ OOXML Пб \ dom4j-1.6.1.jar;» «C:. \ poi -3,9 \ OOXML Пб \ Xmlbeans-2.3.0.jar;»;
Linux Експортувати CLASSPATH = $ CLASSPATH: /usr/share/poi-3.9/poi-3.9-20121203.tar: /usr/share/poi-3.9/poi-ooxml-schemas-3.9-20121203.tar: /usr/share/poi-3.9/poi-ooxml-3.9-20121203.tar: /usr/share/poi-3.9/ooxml-lib/dom4j-1.6.1.tar: /usr/share/poi-3.9/ooxml-lib/xmlbeans-2.3.0.tar

2 ОСНОВНІ КЛАСИ APACHE POI

Опишемо кілька класів і методів в Api-Інтерфейсі Apache POI, які мають вирішальне значення для роботи з файлами Excel за допомогою програм на Java.

Робоча книга

Це супер-інтерфейс усіх класів, які створюють або підтримують книги Excel. Він належить пакету org.apache.poi.ss.usermodel. Два класи, які реалізують цей інтерфейс, що випливають:

Hssfworkbook — у цьому класі є методи для читання й запису файлів Microsoft Excel у форматі .xls. Він сумісний з версіями Ms-office 97–2003.

Xssfworkbook — у цьому класі є методи для читання й запису Xml-Файлів Microsoft Excel і Openoffice у форматі .xls або .xlsx. Це сумісне з версіями Ms-office 2007 або пізніше.

Hssfworkbook — у цьому класі є методи для читання й запису файлів Microsoft Excel у форматі .xls. Він сумісний з версіями Ms-office 97–2003.

Xssfworkbook — у цьому класі є методи для читання й запису Xml-Файлів Microsoft Excel і Openoffice у форматі .xls або .xlsx. Це сумісне з версіями Ms-office 2007 або пізніше.

Hssfworkbook

Це клас високого рівня в пакеті org.apache.poi.hssf.usermodel . Він реалізує інтерфейс Workbook і використовується для файлів Excel у форматі .xls. Нижче перераховані деякі методи й конструктори цього класу.

Конструктори класів

Конструктор і опис
Hssfworkbook () Створює новий об'єкт Hssfworkbook з нуля.
Hssfworkbook (каталог Directorynode, логічні preservenodes)

Створює новий об'єкт Hssfwworkbook усередині певного каталогу.
Hssfworkbook (каталог Directorynode, Poifsfilesystem fs, логічні preservenodes) Враховуючи об'єкт Poifsfilesystem і певний каталог у ньому, він створює об'єкт Ssfworkbook для читання зазначеної книги.
Hssfworkbook (java.io.Inputstream s) Створює новий об'єкт Hssfworkbook, використовуючи вхідний потік.
Hssfworkbook (java.io.Inputstream s, логічні preservenodes) Створює файлову систему POI навколо вашого вхідного потоку.
Hssfworkbook (Poifsfilesystem fs) Створює новий об'єкт Hssfworkbook, використовуючи об'єкт Poifsfilesystem.
Hssfworkbook (Poifsfilesystem fs, логічні preservenodes) Враховуючи об'єкт Poifsfilesystem, він створює новий об'єкт Hssfworkbook для читання зазначеної книги.

Hssfworkbook ()

Створює новий об'єкт Hssfworkbook з нуля.

Hssfworkbook (каталог Directorynode, логічні preservenodes)

Створює новий об'єкт Hssfwworkbook усередині певного каталогу.

Hssfworkbook (каталог Directorynode, Poifsfilesystem fs, логічні preservenodes)

Враховуючи об'єкт Poifsfilesystem і певний каталог у ньому, він створює об'єкт Ssfworkbook для читання зазначеної книги.

Hssfworkbook (java.io.Inputstream s)

Створює новий об'єкт Hssfworkbook, використовуючи вхідний потік.

Hssfworkbook (java.io.Inputstream s, логічні preservenodes)

Створює файлову систему POI навколо вашого вхідного потоку.

Hssfworkbook (Poifsfilesystem fs)

Створює новий об'єкт Hssfworkbook, використовуючи об'єкт Poifsfilesystem.

Hssfworkbook (Poifsfilesystem fs, логічні preservenodes)

Враховуючи об'єкт Poifsfilesystem, він створює новий об'єкт Hssfworkbook для читання зазначеної книги.

Часто використовувані параметри усередині цих конструкторів:

каталог — це каталог файлової системи POI для обробки.

fs — це файлова система POI, яка містить потік робочої книги.

preservenodes — це необов'язковий параметр, який вирішує, чи зберігати інші вузли, такі як макроси. Він споживає багато пам'яті, тому що зберігає всю систему Poifilesystem у пам'яті (якщо встановлена).

каталог — це каталог файлової системи POI для обробки.

fs — це файлова система POI, яка містить потік робочої книги.

preservenodes — це необов'язковий параметр, який вирішує, чи зберігати інші вузли, такі як макроси. Він споживає багато пам'яті, тому що зберігає всю систему Poifilesystem у пам'яті (якщо встановлена).

Примітка. Клас Hssfworkbook містить кілька методів; однак вони сумісні тільки з форматом xls. У цьому керівництві основна увага приділяється останньої версії форматів файлів Excel. Отже, методи класу Hssfworkbook тут не перераховані. Якщо вам потрібні ці методи класу, звернетеся до API класу Poi-hssfworkbook за адресою https://poi.apache.org/apidocs/org/apache/poi/hssf/usermodel/HSSFWorkbook.html.

Xssfworkbook

Це клас, який використовується для вистави форматів файлів Excel як високого, так і низького рівня. Він належить пакету org.apache.xssf.usemodel і реалізує інтерфейс Workbook . Нижче перераховані методи й конструктори цього класу.

Xssfworkbook ()

Створює новий об'єкт Xssfworkbook з нуля.
Xssfworkbook (файл java.io.File) Створює об'єкт Xssfworkbook із заданого файлу.
Xssfworkbook (java.io.Inputstream is) Створює об'єкт Xssfworkbook, буферизуя весь вхідний потік на згадку й потім відкриваючи для нього об'єкт Opcpackage.
Xssfworkbook (шлях java.lang.String) Створює об'єкт Xssfworkbook з повним шляхом до файлу.

Xssfworkbook ()

Створює новий об'єкт Xssfworkbook з нуля.

Xssfworkbook (файл java.io.File)

Створює об'єкт Xssfworkbook із заданого файлу.

Xssfworkbook (java.io.Inputstream is)

Створює об'єкт Xssfworkbook, буферизуя весь вхідний потік на згадку й потім відкриваючи для нього об'єкт Opcpackage.

Xssfworkbook (шлях java.lang.String)

Створює об'єкт Xssfworkbook з повним шляхом до файлу.

Методи класу

Метод і опис
createsheet () Створює аркуш XSSFS для цієї книги, додає його на аркуші й повертає виставу високого рівня.
createsheet (ім'я аркуша java.lang.String) Створює новий аркуш для цієї робочої книги й повертає виставу високого рівня.

Createfont () Створює новий шрифт і додає його в таблицю шрифтів робочої книги.
createcellstyle () Створює новий Xssfcellstyle і додає його в таблицю стилів робочої книги.
Createfont () Створює новий шрифт і додає його в таблицю шрифтів робочої книги.
setprintarea (int sheetindex, int startcolumn, int endcolumn, int startrow, int endrow) Встановлює область друку даного аркуша відповідно до зазначених параметрів.

createsheet ()

Створює аркуш XSSFS для цієї книги, додає його на аркуші й повертає виставу високого рівня.

createsheet (ім'я аркуша java.lang.String)

Створює новий аркуш для цієї робочої книги й повертає виставу високого рівня.

Createfont ()

Створює новий шрифт і додає його в таблицю шрифтів робочої книги.

createcellstyle ()

Створює новий Xssfcellstyle і додає його в таблицю стилів робочої книги.

Createfont ()

Створює новий шрифт і додає його в таблицю шрифтів робочої книги.

setprintarea (int sheetindex, int startcolumn, int endcolumn, int startrow, int endrow)

Установлює область печатки даного аркуша відповідно до зазначених параметрів.

Інші методи цього класу див. У повному документі API за адресою: https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFWorkbook.html. для повного списку методів.

Листи

Sheet — це інтерфейс у пакеті org.apache.poi.ss.usermodel, і це суперінтерфейс усіх класів, які створюють електронні таблиці високого або низького рівня з конкретними іменами. Найпоширенішим типом електронної таблиці є робочий аркуш, який представлений у вигляді сітки гнізд.

Hssfsheet

Це клас у пакеті org.apache.poi.hssf.usermodel . Він може створювати таблиці Excel і дозволяє форматувати стиль аркуша й дані аркуша.

Конструктори класів

Конструктор і опис
Hssfsheet (робоча книга Hssfworkbook) Створює новий аркуш HSSFS, викликуваний Hssfworkbook для створення аркуша з нуля.
Hssfsheet (робоча книга Hssfworkbook, аркуш Internalsheet) Створює аркуш HSSFS, що представляє даний об'єкт аркуша.

Hssfsheet (робоча книга Hssfworkbook)

Створює новий аркуш HSSFS, викликуваний Hssfworkbook для створення аркуша з нуля.

Hssfsheet (робоча книга Hssfworkbook, аркуш Internalsheet)

Створює аркуш HSSFS, що представляє даний об'єкт аркуша.

Xssfsheet

Це клас, який представляє высокоуровневое вистава таблиці Excel. Він перебуває в пакеті org.apache.poi.hssf.usermodel .

Конструктори класів

Конструктор і опис

Xssfsheet () Створює новий аркуш XSSFS — викликається Xssfworkbook для створення аркуша з нуля.
Xssfsheet (частина Packagepart, rel) Створює Xssfsheet, що представляє дану частину пакета й відносини.

Xssfsheet ()

Створює новий аркуш XSSFS — викликається Xssfworkbook для створення аркуша з нуля.

Xssfsheet (частина Packagepart, rel)

Створює Xssfsheet, що представляє дану частину пакета й відносини.

Методи класу

Метод і опис
addmergedregion (Cellrangeaddress region) Додає об'єднану область гнізд (отже, ці гнізда утворюють одиницю).
autosizecolumn (стовпець int) Регулює ширину стовпця відповідно до вмісту.
итератора () Цей метод є псевдонімом для rowiterator (), щоб дозволити цикли foreach.
addhyperlink (гіперпосилання Xssfhyperlink) Реєструє гіперпосилання в колекції гіперпосилань на цьому аркуші

addmergedregion (Cellrangeaddress region)

Додає об'єднану область гнізд (отже, ці гнізда утворюють одиницю).

autosizecolumn (стовпець int)

Регулює ширину стовпця відповідно до вмісту.

итератора ()

Цей метод є псевдонімом для rowiterator (), щоб дозволити цикли foreach.

addhyperlink (гіперпосилання Xssfhyperlink)

Реєструє гіперпосилання в колекції гіперпосилань на цьому аркуші

Для інших методів цього класу, звернетеся до повного API за адресою: https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFSheet.html.

Рядок

Це інтерфейс у пакеті org.apache.poi.ss.usermodel . Він використовується для высокоуровневого вистави рядка електронної таблиці. Це суперінтерфейс усіх класів, які представляють рядки в бібліотеці POI.

Xssfrow

Це клас у пакеті org.apache.poi.xssf.usermodel . Він реалізує інтерфейс рядків, тому він може створювати рядка в електронній таблиці. Нижче перераховані методи й конструктори цього класу.

Методи класу

Метод і опис
createcell (int columnindex) Створює нові гнізда в рядку й повертає її.
setheight (коротка висота) Установлює висоту в коротких одиницях.

createcell (int columnindex)

Створює нові гнізда в рядку й повертає її.

setheight (коротка висота)

Установлює висоту в коротких одиницях.

Для інших методів цього класу перейдіть по зазначенім посиланню https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFRow.html.

Комірка

Це інтерфейс у пакеті org.apache.poi.ss.usermodel . Це суперінтерфейс усіх класів, які представляють комірки в рядках електронної таблиці.

Комірки можуть мати різні атрибути, такі як порожнє, числове, дата, помилка і т. д. Комірки повинні мати свої власні номери (на основі 0) перед додаванням у рядок.

Xssfcell

Це клас у пакеті org.apache.poi.xssf.usermodel . Він реалізує інтерфейс Cell. Це высокоуровневое вистава гнізд у рядках електронної таблиці.

Зведення по полю

Нижче перераховані деякі поля класу Xssfcell разом з їхнім описом.

Тип комірки й опис
CELL_TYPE_BLANK Представляє порожню комірку
CELL_TYPE_BOOLEAN Представляє логічне гніздо (true або false)
CELL_TYPE_ERROR Представляє значення помилки в комірці
CELL_TYPE_FORMULA Представляє результат формули в комірці
CELL_TYPE_NUMERIC Представляє числові дані в комірці
CELL_TYPE_STRING Представляє рядок (текст) у комірку

Методи класу

Метод і опис

setcellstyle (стиль Cellstyle) Установлює стиль для комірки.
setcelltype (int celltype) Установлює тип гнізд (числовий, формули або рядки).
setcellvalue (логічне значення) Встановлює логічне значення для комірки.
setcellvalue (значення java.util.Calendar) Установлює значення дати для комірки.
setcellvalue (подвійне значення) Установлює числове значення для комірки.
setcellvalue (java.lang.String str) Установлює строкове значення для комірки.
sethyperlink (гіперпосилання гіперпосилання) Призначає гіперпосилання на цю комірку.

setcellstyle (стиль Cellstyle)

Установлює стиль для комірки.

setcelltype (int celltype)

Встановлює тип комірок (числовий, формули або рядки).

setcellvalue (логічне значення)

Встановлює логічне значення для комірки.

setcellvalue (значення java.util.Calendar)

Установлює значення дати для комірки.

setcellvalue (подвійне значення)

Установлює числове значення для комірки.

setcellvalue (java.lang.String str)

Встановлює строкове значення для комірки.

sethyperlink (гіперпосилання гіперпосилання)

Призначає гіперпосилання на цю комірку.

Для інших методів і полів цього класу перейдіть по наступнім посиланню: https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFCell.html.

Xssfcellstyle

Це клас у пакеті org.apache.poi.xssf.usermodel . Він надасть можливу інформацію про формат вмісту в комірки електронної таблиці. Він також надає варіанти для зміни цього формату. Він реалізує інтерфейс Cellstyle.

Зведення по полю

У наступній таблиці перераховано кілька полів, які успадковані від інтерфейсу Cellstyle.

Поле й опис
ALIGN_CENTER Центр вирівняти вміст комірки
ALIGN_CENTER_SELECTION Горизонтальне вирівнювання по центру
ALIGN_FILL Комірка відповідає розміру контенту
ALIGN_JUSTIFY Підігнати вміст комірки по ширині
ВИРІВНЯТИ ПО ЛІВОМУ КРАЮ Вирівняти по лівому краю вміст комірки
ALIGN_RIGHT Вирівняйте вміст комірки вправо

BORDER_DASH_DOT Стиль комірки з тире й крапкою
BORDER_DOTTED Стиль комірки з пунктирною рамкою
BORDER_DASHED Стиль комірки з пунктирною обрамівкою
BORDER_THICK Стиль комірки з товстою обрамівкою
BORDER_THIN Стиль гнізда з тонкою облямівкою
VERTICAL_BOTTOM Вирівняйте вміст комірки по вертикалі
VERTICAL_CENTER Вирівняйте вміст комірки по центру
VERTICAL_JUSTIFY Вирівняйте й вирівняйте вміст комірки по вертикалі
VERTICAL_TOP Вертикальне вирівнювання зверху

ALIGN_CENTER

Центр вирівняти вміст комірки

ALIGN_CENTER_SELECTION

Горизонтальне вирівнювання по центру

ALIGN_FILL

комірки відповідає розміру контенту

ALIGN_JUSTIFY

Підігнати вміст комірки по ширині

ВИРІВНЯТИ ПО ЛІВОМУ КРАЮ

Вирівняти по лівому краю вміст комірки

ALIGN_RIGHT

Вирівняйте вміст комірки вправо

BORDER_DASH_DOT

Стиль комірки з тире й крапкою

BORDER_DOTTED

Стиль комірки з пунктирною рамкою

BORDER_DASHED

Стиль комірки з пунктирною обрамівкою

BORDER_THICK

Стиль комірки з товстою обрамівкою

BORDER_THIN

Стиль комірки з тонкою обрамівкою

VERTICAL_BOTTOM

Вирівняйте вміст комірки по вертикалі

VERTICAL_CENTER

Вирівняйте вміст комірки по центру

VERTICAL_JUSTIFY

Вирівняйте й вирівняйте вміст комірки по вертикалі

VERTICAL_TOP

Вертикальне вирівнювання зверху

Конструктори класів

Конструктор і опис
Xssfcellstyle (int cellxfid, int cellstylexfid, Stylestable stylessource, тема Themestable) Створює стиль комірки із частин, що поставляються

Xssfcellstyle (Stylestable stylessource) Створює порожню комірку Стиль

Методи класу

Метод і опис
setalignment (коротке вирівнювання) Установлює тип горизонтального вирівнювання для комірки
setborderbottom (коротка границя) Установлює тип границі для нижньої границі комірки
setbordercolor (сторона Xssfcellborder.Borderside, колір Xssfcolor) Установлює колір для обраної границі
setborderleft (Коротка границя) Установлює тип границі для лівої границі комірки
setborderright (коротка границя) Установлює тип границі для правої границі комірки
setbordertop (коротка границя) Установлює тип границі для верхньої границі гнізда
setfillbackgroundcolor (колір Xssfcolor) Установлює колір заливання тла, представлений у вигляді значення Xssfcolor.
setfillforegroundcolor (колір Xssfcolor) Установлює колір заливання переднього плану, представлений у вигляді значення Xssfcolor.
setfillpattern (короткий фп) Визначає інформацію про заливання гнізд для заливання гнізд по

шаблонові й суцільному кольору.
setfont (шрифт шрифту) Установлює шрифт для цього стилю.
setrotation (коротке обертання) Установлює ступінь повороту для тексту в комірки.
setverticalalignment (коротке вирівнювання) Установлює тип вертикального вирівнювання для комірки.

setalignment (коротке вирівнювання)

Установлює тип горизонтального вирівнювання для комірки

setborderbottom (коротка границя)

Установлює тип границі для нижньої границі комірки

setbordercolor (сторона Xssfcellborder.Borderside, колір Xssfcolor)

Установлює колір для обраної границі

setborderleft (Коротка границя)

Установлює тип границі для лівої границі комірки

setborderright (коротка границя)

Установлює тип границі для правої границі комірки

setbordertop (коротка границя)

Установлює тип границі для верхньої границі комірки

setfillbackgroundcolor (колір Xssfcolor)

Установлює колір заливання тла, представлений у вигляді значення Xssfcolor.

setfillforegroundcolor (колір Xssfcolor)

Установлює колір заливання переднього плану, представлений у вигляді значення Xssfcolor.

setfillpattern (короткий фп)

Визначає інформацію про заливання гнізд для заливання комірки по шаблонові й суцільному кольору.

setfont (шрифт шрифту)

Установлює шрифт для цього стилю.

setrotation (коротке обертання)

Установлює ступінь повороту для тексту в комірки.

setverticalalignment (коротке вирівнювання)

Установлює тип вертикального вирівнювання для комірки.

Для інших методів і полів у цьому класі перейдіть по наступнім посиланню: https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFCellStyle.html.

Hssfcolor

Це клас у пакеті org.apache.poi.hssf.util . Він надає різні кольори як вкладені класи. Звичайно ці вкладені класи представлені з використанням своїх власних індексів. Він реалізує інтерфейс Color.

Вкладені класи

Усі вкладені класи цього класу є статичними, і в кожного класу є свій індекс. Ці вкладені колірні класи використовуються для форматування комірок, такого як уміст комірок, границі, передній план і тло. Нижче перераховані деякі із вкладених класів.

Методи класу

Важливий тільки один метод цього класу, який використовується для одержання значення індексу.

Getindex ()

Цей метод використовується для одержання значення індексу вкладеного класу.

Інші методи й вкладені класи див. По наступнім посиланню: https://poi.apache.org/apidocs/org/apache/poi/hssf/util/HSSFColor.html.

Xssfcolor

Це клас у пакеті org.apache.poi.xssf.usermodel . Він використовується для встановлення кольору в електронній таблиці. Він реалізує інтерфейс Color. Нижче перераховані деякі з його методів і конструкторів.

Xssfcolor () Створює новий екземпляр Xssfcolor.
Xssfcolor (byte [] rgb) Створює новий екземпляр Xssfcolor, використовуючи RGB.
Xssfcolor (java.awt.Color clr) Створює новий екземпляр Xssfcolor, використовуючи клас Color з пакета awt.

Xssfcolor ()
Створює новий екземпляр Xssfcolor.
Xssfcolor (byte [] rgb)
Створює новий екземпляр Xssfcolor, використовуючи RGB.
Xssfcolor (java.awt.Color clr)
Створює новий екземпляр Xssfcolor, використовуючи клас Color з пакета awt.

Методи класу

setauto (логічне авто)

Установлює логічне значення, що вказує, що ctcolor є автоматичним і системний ctcolor є залежним.

setindexed (int indexed)

Установлює індексоване значення ctcolor як системний ctcolor.

Для інших методів перейдіть по наступнім посиланню: https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFColor.html.

Xssffont

Це клас у пакеті org.apache.poi.xssf.usermodel . Він реалізує інтерфейс Font і тому може обробляти різні шрифти в книзі.

Конструктор класів

Xssffont ()

Створює новий екземпляр Xssfont.

Методи класу

Метод і опис
setbold (логічне напівжирне) Установлює логічне значення для атрибута «напівжирний».
setcolor (короткий колір) Установлює індексований колір для шрифту.
setcolor (Xssfcolor color) Установлює колір для шрифту в стандартному альфа RGB колірнім значенні.
setfontheight (коротка висота) Установлює висоту шрифту в пунктах.
setfontname (ім'я java.lang.String) Установлює ім'я для шрифту.
setitalic (логічний курсив) Установлює логічне значення для властивості ‘italic’.

setbold (логічне напівжирне)

Установлює логічне значення для атрибута «напівжирний».

setcolor (короткий колір)

Установлює індексований колір для шрифту.

setcolor (Xssfcolor color)

Установлює колір для шрифту в стандартному альфа RGB колірнім значенні.

setfontheight (коротка висота)

Установлює висоту шрифту в пунктах.

setfontname (ім'я java.lang.String)

Встановлює ім'я для шрифту.

setitalic (логічний курсив)

Встановлює логічне значення для властивості ‘italic’.

Для інших методів перейдіть по наступнім посиланню: https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFFont.html.

Xssfhyperlink

Це клас у пакеті org.apache.poi.xssf.usermodel . Він реалізує інтерфейс Hyperlink. Він використовується для установки гіперпосилання на вміст гнізда електронної таблиці.

поля

Поля цього класу наступні. Тут поля означають типи використовуваних гіперпосилань.

LINK_DOCUMENT

Використовується для посилання на будь-який інший документ

LINK_EMAIL

Використовується для посилання на електронну пошту

LINK_FILE

Використовується для зв'язку будь-якого іншого файлу в будь-якому форматі

LINK_URL

Використовується для посилання на веб-URL

Методи класу

setaddress (адреса java.lang.String)

Адреса гіперпосилання.

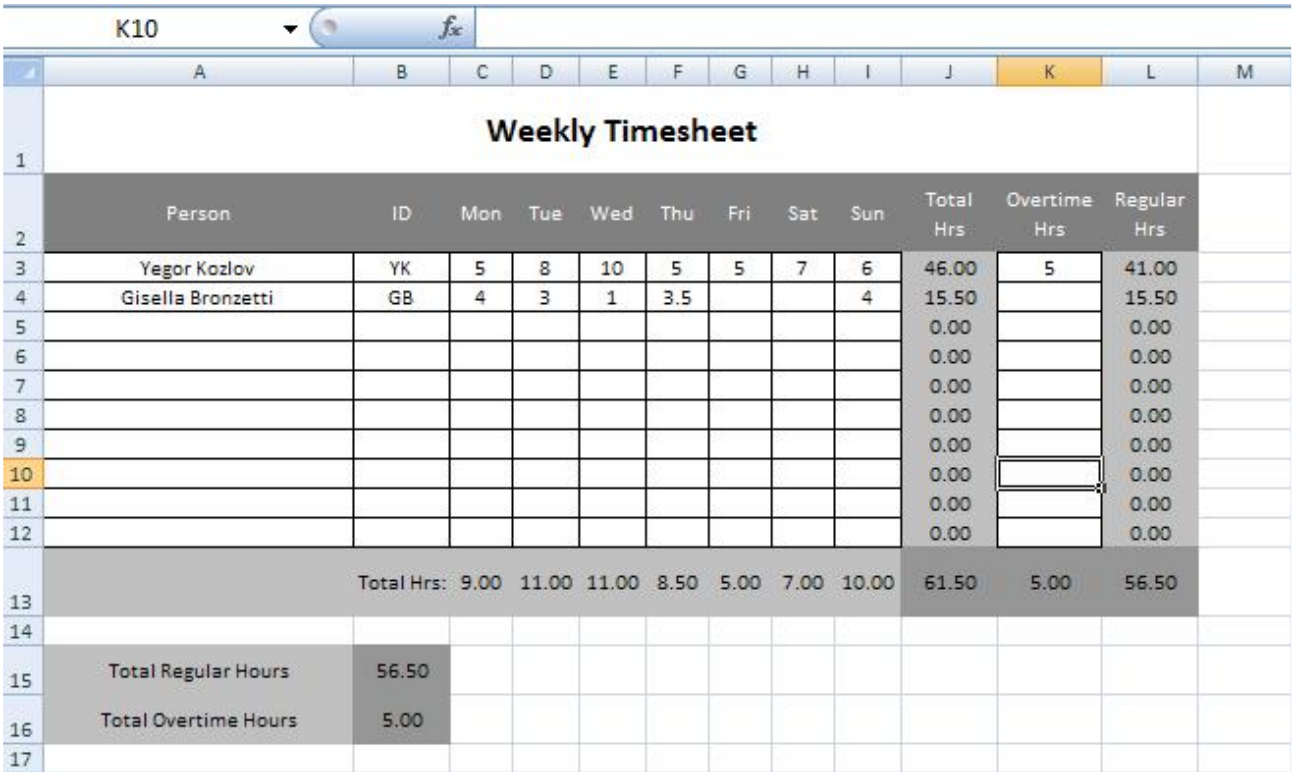
Для інших методів перейдіть по наступнім посиланню: https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFHyperlink.html.

Це клас у пакеті org.apache.poi.xssf.usermodel . Він реалізує інтерфейс Creationhelper. Він використовується як клас підтримки для оцінки формул і настроювання гіперпосилань.

3 РОЗРОБКА ПРОГРАМ ДЛЯ ФОРМУВАННЯ ЗВІТІВ У MS EXCEL

3.1 Розробка програми Weekly Timesheet

Створюємо програму, що створює щотижневий табель обліку робочого часу, з автоматичним розрахунком загальної кількості годин (рис. 3.1).



K10

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Weekly Timesheet												
2	Person	ID	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total Hrs	Overtime Hrs	Regular Hrs	
3	Yegor Kozlov	YK	5	8	10	5	5	7	6	46.00	5	41.00	
4	Gisella Bronzetti	GB	4	3	1	3.5			4	15.50		15.50	
5										0.00		0.00	
6										0.00		0.00	
7										0.00		0.00	
8										0.00		0.00	
9										0.00		0.00	
10										0.00		0.00	
11										0.00		0.00	
12										0.00		0.00	
13		Total Hrs:	9.00	11.00	11.00	8.50	5.00	7.00	10.00	61.50	5.00	56.50	
14													
15	Total Regular Hours	56.50											
16	Total Overtime Hours	5.00											
17													

Рисунок 3.1 – Вигляд табелю обліку робочого часу

Наведемо текст програми на мові Java.

```
import java.io.FileOutputStream;
import java.util.HashMap;
import java.util.Map;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.BorderStyle;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellStyle;
import org.apache.poi.ss.usermodel.FillPatternType;
import org.apache.poi.ss.usermodel.Font;
import org.apache.poi.ss.usermodel.HorizontalAlignment;
import org.apache.poi.ss.usermodel.IndexedColors;
import org.apache.poi.ss.usermodel.PrintSetup;
   private static final String[] titles = {
```

```
        "Person",   "ID", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun",
        "Total\nHrs", "Overtime\nHrs", "Regular\nHrs"
};
private static final Object[][] sample_data = {
        {"Yegor Kozlov", "YK", 5.0, 8.0, 10.0, 5.0, 5.0, 7.0, 6.0},
        {"Gisella Bronzetti", "GB", 4.0, 3.0, 1.0, 3.5, null, null, 4.0},
};
private TimesheetDemo() {}
public static void main(String[] args) throws Exception {
    Workbook wb;
    if(args.length > 0 && args[0].equals("-xls")) wb = new HSSFWorkbook();
    else wb = new XSSFWorkbook();
    Map<String, CellStyle> styles = createStyles(wb);
    Sheet sheet = wb.createSheet("Timesheet");
    PrintSetup printSetup = sheet.getPrintSetup();
    printSetup.setLandscape(true);
    sheet.setFitToPage(true);
    sheet.setHorizontallyCenter(true);
    //title row
    Row titleRow = sheet.createRow(0);
    titleRow.setHeightInPoints(45);
    Cell titleCell = titleRow.createCell(0);
    titleCell.setCellValue("Weekly Timesheet");
    titleCell.setCellStyle(styles.get("title"));
    sheet.addMergedRegion(CellRangeAddress.valueOf("$A$1:$L$1"));
    //header row
    Row headerRow = sheet.createRow(1);
    headerRow.setHeightInPoints(40);
    Cell headerCell;
    for (int i = 0; i < titles.length; i++) {
        headerCell = headerRow.createCell(i);
        headerCell.setCellValue(titles[i]);
        headerCell.setCellStyle(styles.get("header"));
    }
    int rownum = 2;
    for (int i = 0; i < 10; i++) {
        Row row = sheet.createRow(rownum++);
        for (int j = 0; j < titles.length; j++) {
            Cell cell = row.createCell(j);
            if(j == 9){
         //the 10th cell contains sum over week days, e.g. SUM(C3:I3)
                String ref = "C" +rownum+ ":I" + rownum;
                cell.setCellFormula("SUM("+ref+")");
                cell.setCellStyle(styles.get("formula"));
```

```
        } else if (j == 11){
            cell.setCellFormula("J" +rownum+ "-K" + rownum);
            cell.setCellStyle(styles.get("formula"));
        } else {
            cell.setCellStyle(styles.get("cell"));
        }
    }
}
//row with totals below
Row sumRow = sheet.createRow(rownum++);
sumRow.setHeightInPoints(35);
Cell cell;
cell = sumRow.createCell(0);
cell.setCellStyle(styles.get("formula"));
cell = sumRow.createCell(1);
cell.setCellValue("Total Hrs:");
cell.setCellStyle(styles.get("formula"));
for (int j = 2; j < 12; j++) {
    cell = sumRow.createCell(j);
    String ref = (char)('A' + j) + "3:" + (char)('A' + j) + "12";
    cell.setCellFormula("SUM(" + ref + ")");
    if(j >= 9) cell.setCellStyle(styles.get("formula_2"));
    else cell.setCellStyle(styles.get("formula"));
}
rownum++;
sumRow = sheet.createRow(rownum++);
sumRow.setHeightInPoints(25);
cell = sumRow.createCell(0);
cell.setCellValue("Total Regular Hours");
cell.setCellStyle(styles.get("formula"));
cell = sumRow.createCell(1);
cell.setCellFormula("L13");
cell.setCellStyle(styles.get("formula_2"));
sumRow = sheet.createRow(rownum++);
sumRow.setHeightInPoints(25);
cell = sumRow.createCell(0);
cell.setCellValue("Total Overtime Hours");
cell.setCellStyle(styles.get("formula"));
cell = sumRow.createCell(1);
cell.setCellFormula("K13");
cell.setCellStyle(styles.get("formula_2"));
//set sample data
for (int i = 0; i < sample_data.length; i++) {
    Row row = sheet.getRow(2 + i);
```

```
            for (int j = 0; j < sample_data[i].length; j++) {
                if(sample_data[i][j] == null) continue;
                if(sample_data[i][j] instanceof String) {
row.getCell(j).setCellValue((String)sample_data[i][j]);
                } else {
row.getCell(j).setCellValue((Double)sample_data[i][j]);
                }
            }
```

3.2 Розробка програми звіту

З великої таблиці, яка має багато різної інформації, ми робимо короткий аналітичний звіт для прийняття рішення. Для прикладу ми створюємо звіт для аналізу кредитної історії (рис. 3.2). Повний текст програми знаходиться у додатку А.

	A	B	C	D	E	F
1	Ім'я	Номер угоди	Дата видачі	Дата погашення	Тип кредиту	Сума
2	Куліш	№4572	23.11.2007	23.11.2012	2 car loans	4 320,13
3	Палій	№5261	26.09.2008	26.09.2013	2 car loans	22 070,27
4	Попова	№5138	08.08.2008	07.08.2009	3 consumer loans	3 893,69
5	Чорний	№4877	25.04.2008	24.04.2019	3 consumer loans	1 108 305,83
6	Шевченко	№4888	05.05.2008	04.05.2009	3 consumer loans	0,00
7	Мельник	№4397	02.08.2007	02.08.2022	1 mortgage loans	820 497,68
8	Шевченко	№4942	27.05.2008	26.05.2023	1 mortgage loans	369 704,50
9	Бойко	№5128	04.08.2008	04.08.2033	1 mortgage loans	38 929,53
10	Коваленко	№4746	28.02.2008	27.02.2013	3 consumer loans	38 095,94
11	Бондаренко	№4863	21.04.2008	21.04.2015	2 car loans	54 958,42
12	Ткаченко	№4313	05.06.2007	04.06.2012	2 car loans	1 212,13
13	Ковальчук	№4955	30.05.2008	29.05.2014	2 car loans	14 688,38
14	Кравченко	№5398	26.08.2009	25.08.2014	2 car loans	27 775,46
15	Олійник	№5294	09.10.2008	09.10.2015	2 car loans	41 683,05
16	Шевчук	№4975	05.06.2008	05.06.2015	2 car loans	79 218,73
17	Коваль	№5184	28.08.2008	28.08.2015	2 car loans	80 999,39
18	Поліщук	№4957	30.05.2008	29.05.2015	2 car loans	55 700,93
19	Бондар	№5301	10.10.2008	10.10.2023	1 mortgage loans	476 308,46

Рисунок 3.2 – Приклад звіту в MS Excel

Наведемо текст програми на мові Java.

```
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.VerticalAlignment;
import org.apache.poi.ss.usermodel.Workbook;
```

```
import org.apache.poi.ss.util.CellRangeAddress;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public final class TimesheetDemo {
   private static final String[] titles = {
         "Person",   "ID", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun",
         "Total\nHrs", "Overtime\nHrs", "Regular\nHrs"
   };

   private static final Object[][] sample_data = {
         {"Yegor Kozlov", "YK", 5.0, 8.0, 10.0, 5.0, 5.0, 7.0, 6.0},
         {"Gisella Bronzetti", "GB", 4.0, 3.0, 1.0, 3.5, null, null, 4.0},
   };
   private TimesheetDemo() {}
   public static void main(String[] args) throws Exception {
      Workbook wb;
   private static Map<String, CellStyle> createStyles(Workbook wb){
      Map<String, CellStyle> styles = new HashMap<>();
      style.setAlignment(HorizontalAlignment.CENTER);
      style.setVerticalAlignment(VerticalAlignment.CENTER);
      styles.put("header", style);
      style = wb.createCellStyle();
      style.setAlignment(HorizontalAlignment.CENTER);
      style.setWrapText(true);
      style.setBorderRight(BorderStyle.THIN);
      style.setRightBorderColor(IndexedColors.BLACK.getIndex());
      style.setBorderLeft(BorderStyle.THIN);
      style.setLeftBorderColor(IndexedColors.BLACK.getIndex());
      style.setBorderTop(BorderStyle.THIN);
      style.setTopBorderColor(IndexedColors.BLACK.getIndex());
      style.setBorderBottom(BorderStyle.THIN);
      style.setBottomBorderColor(IndexedColors.BLACK.getIndex());
      styles.put("cell", style);
      style = wb.createCellStyle();
      style.setAlignment(HorizontalAlignment.CENTER);
      style.setVerticalAlignment(VerticalAlignment.CENTER);
style.setFillForegroundColor(IndexedColors.GREY_25_PERCENT.getIndex());
      style.setFillPattern(FillPatternType.SOLID_FOREGROUND);
      style.setDataFormat(wb.createDataFormat().getFormat("0.00"));
      styles.put("formula", style);
      style = wb.createCellStyle();
      style.setAlignment(HorizontalAlignment.CENTER);
      style.setVerticalAlignment(VerticalAlignment.CENTER);
style.setFillForegroundColor(IndexedColors.GREY_40_PERCENT.getIndex());
      style.setFillPattern(FillPatternType.SOLID_FOREGROUND);
```

```
    style.setDataFormat(wb.createDataFormat().getFormat("0.00"));
    styles.put("formula_2", style);
    return styles;
}
}
```

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.1 Розрахунок витрат на розробку програмного продукту

Витрати на розробку та впровадження програмного продукту (К) включають:

$$К=К1+К2,$$

де К1 – витрати на розробку програмного продукту, грн.

К2 – витрати на налагодження та дослідну експлуатацію програмного продукту на ПК, грн.

Витрати на розробку програмного продукту включають:

- витрати на оплату праці розробників (Воп);
- нарахування на зарплату (Нз);
- витрати на куповані вироби (Вк);
- накладні витрати (Нв);
- інші витрати (Він).

Для розробки програмного продукту потрібні 4 спеціалісти-розробники, а саме:

- керівник проекту (Кп);
- консультант з економічної частини (Ке);
- консультант з охорони праці (Коп);
- студент-дипломник (Сд).

Згідно з штатним розписом ВСП «Фаховий коледж інформаційних технологій «НУ ЛП», одна година навантажень становить для:

- керівник проекту (Кп) – 118,13 грн;
- консультант з економічної частини (Ке) - 118,13 грн;
- консультант з охорони праці (Коп) - 103,48 грн;
- студент-дипломник (Сд) - 8,73 грн.

Денна оплата студента дипломника визначається:

Стипендія / 173,

де 173 - місячний фонд робочого часу, год.

1510,00 / 173 = 8,73 грн/год

Розрахунок витрат на оплату праці всіх спеціалістів проекту визначається за формулою:

$$Воп=\sum Рі * ti * Зпі,$$

де Рі – чисельність розробників проекту і-спеціальності, роб.;

ti – час, витрачений за розробку проекту розробником і-спеціальності, год.;

Зпі – погодинна заробітна плата розробника і-спеціальності, грн..

Таким чином, витрати на оплату праці розробників складають:

Зкп = 1 * 14 * 118,13 = 1 653,82 грн

Зке = 1 * 1 * 118,13 = 118,13 грн

Зкоп = 1 * 1 * 103,48 = 103,48 грн

Зсд = 1 * 180 * 8,73 = 1 571,4 грн

Сумарні витрати на оплату праці:

Воп = (1* 4 * 118,13) + (1*1 * 118,13) + (1*1 * 103,48) + (1 * 180 * 8,73) =

= 1 653,82 + 118,13 + 103,48 + 1 571,4 = 3 446,83 грн

Розрахунок витрат на оплату праці розробників наведено у таблиці 4.1.

Таблиця 4.1 – Розрахунок витрат на оплату праці

Спеціальність розробника	Кількість розробників, роб.	Час роботи, год.	Погодинна заробітна плата розробника, грн.	Витрати на оплату праці, грн.
Керівник проекту	1	14	118,13	1 653,82
Консультант з економічної частини	1	1	118,13	118,13
Консультант з охорони праці	1	1	103,48	103,48
Студент-дипломник	1	180	8,73	1 571,4
Всього	4	-	-	3 446,83

Нарахування на зарплату визначаються за формулою:

$$Нз = (Воп- Зсд) * 22,0 / 100,$$

де Воп – витрати на оплату праці, тис.грн.

22 – норматив нарахувань на зарплату, %

$$Нз= (3\ 446,83-1\ 571,4) * 22,0 / 100 = 412,59 \text{ грн}$$

Витрати на куповані вироби визначаються за їх фактичними цінами з врахуванням найменування, номенклатури та необхідної кількості в проекті. Транспортно-заготівельні витрати становлять 10% від суми витрат на куповані вироби.

У таблиці 4.2 наведено розрахунок витрат на куповані вироби

Таблиця 4.2 – Розрахунок витрат на куповані вироби.

Найменування купованих виробів	Одиниця виміру	Кількість, шт.	Ціна за одиницю, грн.	Сума витрат, грн.
Папір А4 New Future Laser 80 г/м2	Пачка	1	250	250
Роздрук пояснювальної записки	Аркуш	80	3	240
Папка для дипломного проєкту	-	1	230	230
Разом	-	-	-	520
Транспортно-заготівельні витрати (10%)	-	-	-	52
Всього	-	-	-	572

Витрати на куповані вироби становлять:

$$Вк = 250 + 240 + 230 + 52 = 572 \text{ грн}$$

Накладні витрати становлять 30% від витрат на оплату праці:

$$Нв = 3\ 446,83 * 30 / 100 = 1\ 034,05 \text{ грн}$$

Інші витрати обчислюються по їх питомій вазі у структурі собівартості (10%) :

$$Він = (Воп + Нз + Нв + Вк) * 10 / 90$$

$$Він = (3\ 446,83 + 412,59 + 1\ 034,05 + 572) * 10 / 90 = 607,27 \text{ грн}$$

Витрати на розробку програмного продукту визначаються за формулою:

$$К1 = Воп + Нз + Нв + Вк + Він,$$

$$К1 = 3\ 446,83 + 412,59 + 1\ 034,05 + 572 + 607,27 = 6\ 072,74 \text{ грн}$$

4.2 Розрахунок витрат на налагодження та дослідну експлуатацію програмного продукту на ПК

Програма була розроблена на протязі 30 днів із розрахунком 6 годин на день (t 180 год.).

Потужність комп'ютерної техніки (Р) включає ПК який споживає 1,2 кВт/год. Вартість однієї машино-години роботи визначається за формулою:

$$Sm.r. = P * Tф,$$

де Р – потужність комп'ютерної техніки, кВт;

Тф - вартість 1 кВт-год електроенергії, грн. (Тф = 7,50).

$$Sm.r.= 1,2 * 7,50 = 9 грн/год$$

Витрати на налагодження та дослідну експлуатацію програмного продукту на ПК визначаються за формулою:

$$К2 = Smr * t,$$

де Sm.r. – вартість однієї машино-години роботи, грн \ год.

t – машинний час, витрачений на налагодження та дослідну експлуатацію програмного продукту, год.

$$К2 = 9 * 180 = 1\ 620 грн$$

Таким чином, витрати на розробку та впровадження програмного продукту становлять:

$$К = 5\ 952,74 + 1\ 620 = 7\ 552,74 грн$$

У таблиці 4.3 наведено кошторис витрат на розробку та впровадження програмного продукту.

Таблиця 4.3 Кошторис витрат на розробку та впровадження програмного

Найменування елементів витрат	Сума витрат, грн.
Витрати на оплату праці	3 446,83
Нарахування на зарплату	412,59
Витрати на куповані вироби	572
Накладні витрати	1 034,05
Інші витрати	607,27
Витрати на налагодження та дослідну експлуатацію	1 620
Всього:	7 552,74

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

Охорона праці – це надзвичайно важлива система, яка складається з різноманітних правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів. Вона має на меті збереження життя, здоров'я та працездатності людини у процесі трудової діяльності. У цьому контексті, важливими об'єктами охорони праці є людина в процесі праці, середовище виробництва та організації праці на ньому.

На підприємствах вкрай важливо створювати безпечні умови праці, адже це не тільки зберігає здоров'я та життя працівників, але й сприяє ефективнішому виробництву. Відповідно, слід дотримуватися вимог, викладених у нормативних документах. Це значно знизить наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з відеодисплейними терміналами.

5.1 Електробезпека

Для забезпечення безпечної та ефективної експлуатації ЕОМ та ВДТ необхідно дотримуватись певних принципів та правил. Перш за все, регулярні щомісячні діагностичні огляди та чистка пристроїв допоможуть уникнути накопичення пилу, які можуть спричинити утворення електростатичного струму та пробою. Під час оглядів також необхідно оцінювати стан радіокомпонентів та замінювати їх на компоненти аналогічних номіналів за необхідності, щоб забезпечити найвищу продуктивність пристроїв. Крім того, монтаж електропроводів, кабелів та ЕОМ, а також їх ремонт та обслуговування має проводитись за допомогою діелектричних засобів та при повному відімкненні від загальної мережі для усунення можливості утворення струмів короткого замикання або струмів перенавантаження.

Електропроводи та кабелі, а також ЕОМ з ВДТ і ПП мають відповідати виконанням та ступеню захисту класу зони за НПАОП 40.1-1.32-01. Для

забезпечення безпеки працівників, які працюють з ЕОМ та персональними комп'ютерами, необхідно дотримуватись вимог електробезпеки, встановлених нормативними документами, зокрема: «Правила улаштування електроустановок» (ПУЕ) та «Правилах охорони праці під час експлуатації електронно-обчислюваних машин» (НПАОП 0.00-1.28-10.). Також важливо враховувати, що лінія електромережі для живлення ЕОМ з ВДТ має бути виконана як окрема групова трипровідна мережа з фазового, нульового та захисного провідника. Усі провідники мають відповідати вимогам НПАОП 40.1-1.32-01. Нульовий захисний провідник має бути прокладений від стійки групового розподільного щита. ЕОМ з ВДТ має під'єднуватись до електромережі лише за допомогою справних штепсельних з’єднань і електророзеток заводського виготовлення.

У приміщенні, де одночасно експлуатується або обслуговується більше ніж п'ять персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Неприпустимим є підключення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ до звичайної двопровідної електромережі, в тому числі — з використанням перехідних пристроїв. Загалом, дотримання цих принципів та правил забезпечить безпечну та ефективну роботу ЕОМ та ВДТ.

5.2 Техніка безпеки при роботі за комп’ютером

Під час роботи на комп'ютері можуть діяти шкідливі фактори, які можуть вплинути на здоров'я користувача. Інформаційна безпека та електробезпека забезпечуються за допомогою ряду профілактичних заходів. Один з найважливіших аспектів є саме електробезпека. Для усунення ризику ураження струмом необхідно розміщувати обладнання та кабелі відповідно до вимог безпеки. Це може бути досягнуто за допомогою захисного заземлення,

використання безпечних розеток та електропроводки, розрахованих на потужність системи, а також ізоляції всіх проводів.

Для забезпечення ефективності та продуктивності роботи комп'ютера важливо регулярно чистити внутрішні частини від пилу, користуватися окремими вогнетривкими столами для комп'ютерів та іншого устаткування. Щоб запобігти іскрінню, необхідно рідше вставляти та виймати вилки з розеток.

Освітлення на робочому місці повинно відповідати характеру зорової роботи, який визначається трьома параметрами: об'єктом розрізнення, фоном та контрастом об'єкта і фону. Потрібно забезпечити рівномірне розподілення яскравості на моніторі та навколишньому просторі, відсутність різких тіней, відблисків та стабільну освітленість під час роботи. Вибирати оптимальну спрямованість світлового потоку та необхідний склад світла. Для забезпечення здоров'я користувача можна використовувати спеціальні світильники та підсвічування, які забезпечують оптимальні умови для зорової роботи. КПО>1,5%.

Правила безпеки при роботі за комп'ютером:

- Увімкніть кондиціонер у приміщенні, щоб уникнути перегрівання пристроїв та забезпечити комфортну температуру для користувача.
- Переконайтесь у стабільному розташуванні обладнання на столі. Не забудьте, що нестабільна підставка для монітора може призвести до його падіння та пошкодження. Відкрийте монітор так, щоб було зручно спостерігати екран – прямо (не збоку) і трохи зверху вниз, з нахилом екрана, його нижній край ближче до користувача.
- Перевірте загальний стан обладнання, справність електропроводки, кабелів, вилок, розеток та заземлення захисного екрана. Якщо щось несправне, виправте це перед роботою.
- Налаштуйте освітлення робочого місця. Краще мати якісне та достатнє освітлення, щоб зменшити навантаження на очі та запобігти їх перевтомленню.

-Регулюйте та фіксуйте висоту крісла та зручний нахил спинки для користувача. Важливо, щоб користувач почував себе комфортно та міг працювати тривалий час без відчуття дискомфорту.

-Під’єднайте необхідне обладнання до системного блоку. Кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері, щоб уникнути пошкодження пристроїв та пошкодження даних.

- Увімкніть комп'ютерне обладнання послідовно: монітор, системний блок, принтер (якщо потрібно друкувати). Це дозволить уникнути перезавантаження системи та забезпечити її стабільну роботу.

-Налаштуйте яскравість монітора та контрастність. Не робіть зображення надто яскравим, щоб не перевтомлювати очі. Також важливо відпочивати та робити паузи під час тривалої роботи за комп'ютером, щоб не перенапружувати очі та запобігти втомі.

5.3 Пожежна безпека

Пожежна безпека об'єкта – це важливий стан об'єкта, про який регламентом визначено імовірність виникнення та розвитку пожежі та впливу на людей її небезпечних факторів, а також забезпечується захист матеріальних цінностей. Цей стан регулюється нормативними актами, зокрема ДБН В.1.1-7-2002 "Пожежна безпека об'єктів будівництва", ДСТУ 2272:2006 "Пожежна безпека. Терміни та визначення основних понять", НАПБ А.01.001-2004 "Правила пожежної безпеки в Україні".

Важливою складовою пожежної безпеки є належне ознайомлення всіх працівників з правилами пожежної безпеки, проходження протипожежних інструктажів та перевірки знань з питань пожежної безпеки. Зокрема, автоматична пожежна сигналізація повинна завжди бути у ввімкненому, черговому стані.

Приміщення для роботи повинно відповідати нормам, зазначеним у НАПБ А.01.001.-2004 про будівлі та приміщення для ЕОМ. Зокрема, підлога та стіни в такому приміщенні повинні входити в групу горючості Г1. Заборонено зберігати легкозаймисті та горючі речовини в будь-яких кількостях в приміщенні з ЕОМ. Важливо не залишати ЕОМ без нагляду під час роботи та вимикати їх від глобальної електромережі по закінченню роботи з ними. На робочому місці має бути встановлений легкий доступ до двох газових вогнегасників як засобів первинної пожежної безпеки. Меблі та обладнання повинні розміщуватися таким чином, щоб забезпечувався вільний евакуаційний прохід до дверей виходу з приміщення (завширшки не менше 1 м). Евакуаційні шляхи та виходи необхідно постійно утримувати вільними, нічим не захаращувати. Документи, папір та інші горючі матеріали слід зберігати на відстані не менше 1 м від електрощитів і електрокабелів; 0,5 м від електросвітильників; 0,6 м від сповіщувачів пожежної сигналізації та 0,15 м від приладів центрального водяного опалення. Засоби протипожежного захисту слід утримувати у справному стані. Відстань від найбільш віддаленого місця до вогнегасника не повинна бути більшою за 20 м.

Приміщення, у яких розміщені ПЕОМ, слід оснащувати переносними вуглекислотними вогнегасниками з розрахунку один вогнегасник ВВК-2 або один ВВПА-400 на три ПЕОМ, але не менше ніж один вогнегасник зазначених типів на приміщення.

Крім того, рекомендується забезпечити додатковий простір для працівників, який забезпечить їм додаткові можливості для руху та роботи з ПЕОМ, а також забезпечити доступ до відповідних джерел електроживлення. З метою забезпечення повної пожежної безпеки рекомендується проводити щорічні перевірки на відповідність пожежним нормам та стандартам. Для гасіння пожеж класу В (горючі рідини) та класу Е (електроустановки, що знаходяться під напругою) вуглекислотні вогнегасники (ВВК) ефективно застосовуються. Їх механізм дії полягає в тому, що вони працюють шляхом витіснення кисню з зони горіння та охолодження матеріалів, що горять.

Вогнегасники ВВК не залишають слідів після використання, що робить їх ідеальними для гасіння пожеж в електронному обладнанні та на об'єктах з цінними матеріалами.Вуглекислотний вогнегасник ВВК-2 наведено на рисунку 5.1

Рисунок 5.1 – Вогнегасник вуглекислотний ВВК-2.

5.4 Виробниче приміщення та робоче місце

Згідно з вимогами ДСанПіН 3.3.2.007-98, приміщення, в яких планується робота з ВДТ, повинні відповідати проєктній документації будинку, погодженій з уповноваженими державними органами. Якщо ви плануєте працювати з відомчими документами та матеріалами на комп'ютері, то вам необхідно взяти до уваги правила розташування робочого місця, які забезпечать вам максимальний комфорт та безпеку під час роботи.

Розміщення робочих місць з ВДТ у підвальних приміщеннях та на цокольних поверхах заборонено. Якщо ваше робоче місце знаходиться в приміщенні, то зверніть увагу на доступність основних умов, необхідних для

продуктивної роботи. На одного працівника площа робочого місця має становити не менше ніж 6,0 м^2, а об'єм – не менше ніж 20,0 м^3.

У залежності від того, яку роботу ви виконуєте, має бути враховано чинні санітарні норми освітлення, температури, відносної вологості повітря, сили та ступеня вібрації, звукового шуму, вогнестійкості, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря.

Щоб забезпечити високий рівень комфорту та безпеки під час роботи з комп'ютером, на кожну кімнату, де працюють співробітники, повинні бути наявні елементи природного та штучного освітлення відповідно до ДБН В.2.5-28-2006. Для досягнення максимального рівня безпечності та охорони праці при роботі з комп’ютером виробничі приміщення необхідно обладнати аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації й вогнегасниками. На вікнах повинні бути встановлені сонцезахисні плівки.

При розташуванні елементів робочого місця користувача ПК необхідно враховувати робочу позу користувача, простір для розміщення користувача, можливість огляду елементів робочого місця, можливість ведення записів, розміщення документації та матеріалів, якими користуватиметься працівник. Робочі місця з ПК мають бути розташовані від стіни з вікнами на відстані не менше ніж 1,5 м, від інших стін – на відстані не менше ніж 1 м. Недопустиме таке розташування ПК, при якому працівник повернутий обличчям або спиною до вікон кімнати або до задньої частини ПК, у яку вмонтовані вентилятори. Загальні рекомендації до робочої пози та робочого місця наведені на рисунку 5.2

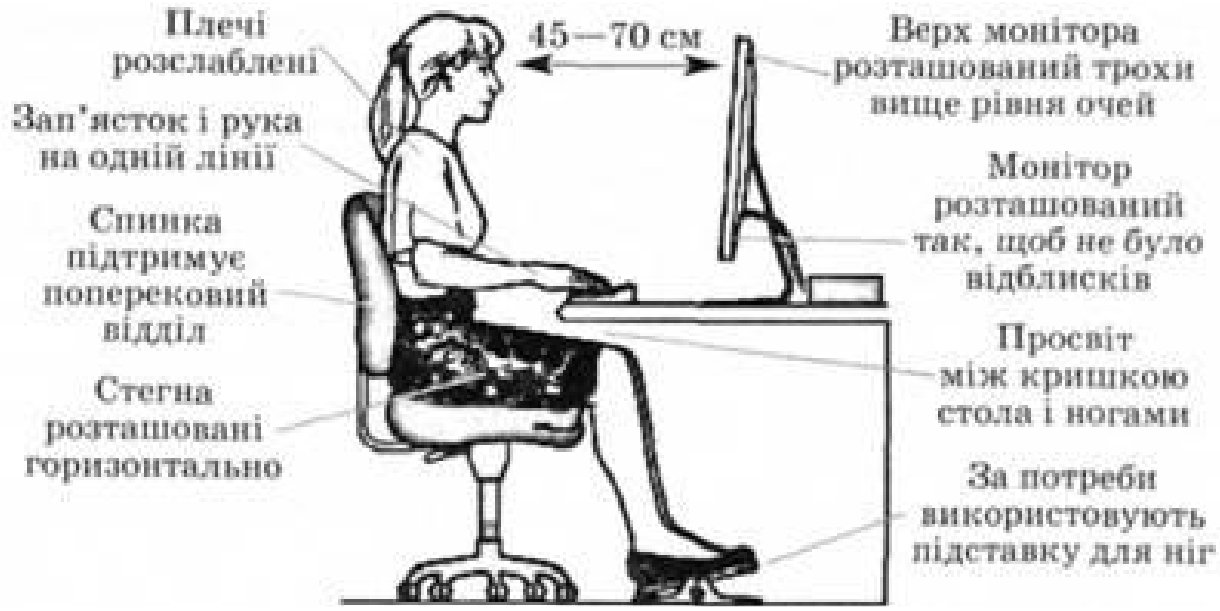


Рисунок 5.2 – Рекомендації до робочої пози та робочого місця

5.5 Висновки до розділу з охорони праці та безпеки життєдіяльності

Дипломний проєкт був виконаний з дотриманням усіх зазначених нижче стандартів та норм, що гарантували мою безпеку під час роботи:

- Електробезпека. Під час використання комп'ютера, були виконані всі вимоги електробезпеки, вказані в технічній документації виробника;
- Техніка безпеки при роботі за комп'ютером. Всі вимоги щодо безпеки користувача під час роботи за комп'ютером, такі як правильне розташування апаратури, ергономіка робочого місця, перерви в роботі та забезпечення зорового комфорту, були дотримані належним чином;
- Пожежна безпека. Умови зберігання та експлуатації, що забезпечують пожежну безпеку, згідно з технічною документацією виробників комп'ютерів, були виконані протягом усієї роботи;
- Виробниче приміщення та робоче місце. Робота над дипломним проєктом відбувалась в приміщенні з робочим місцем, яке відповідало всім стандартам та ергономічним нормам.

ВИСНОВКИ

У дипломному проекті ми обґрунтували вибір програмного забезпечення описали процес встановлення та підключення бібліотеки Apache POI до проекту Java. Навели особливості програмування різних версій програми Excel. Ми розробили програми, що створюють файли Excel засобами мови Java, записують дані та можуть їх зчитати. Також ми розробили програми з можливістю змінювати параметри шрифту, розміри та границі комірок програми Excel з використанням бібліотеки Apache POI.

Найбільш складна робота у проекті це створення програм, що зчитують данні різного типу з комірок Excel та дозволяють запрограмувати люьі формули. Також для проведення лабораторних робіт ми створили варіанти індивідуальних завдань для закріплення знань з даної тематики.

У практичній частині проекту створено програми, що дозволяють за допомогою мови програмування Java створити файли Excel з використанням будь якого форматування, застосуванням формул та можливістю зчитування даних з комірок Excel.

Даний дипломний проект може бути використаний у якості методичного матеріалу для вивчення студентами мови об’єктного програмування Java.

ПЕРЕЛІК ПОСИЛАНЬ

1 Герберт Шилдт “Java 8. Полное руководство”. Вильямс.2015 [1376].

2 Брюс Еккель “Філософія Java”. Питер, 2009 [640]

3 Хабибулин И. Самоучитель Java. СПб.: БХВ, 2008. 720 с.

4 Хорстманн К.С., Корнелл Г. Java 2. Библиотека профессионала. Т. 1, Основы.М.: Вильямс, 2009. 816 с.

5 Хорстманн К.С., Корнелл Г. Java 2. Библиотека профессионала. Т. 2, Тонкости программирования. М.: Вильямс, 2010. 992 с.

6 Монахов В. Язык программирования Java и среда NetBeans. СПб.: БХВ, 2009. 720 с.

7 Шилдт Г. Java: руководство для начинающих. М.: Вильямс, 2008. 720 с.

8 Шилдт Г. Java: методики программирования Шилдта. М.: Вильямс, 2008. 512 с.

9 Фишер Т. Р. Java. Карманный справочник. М.: Вильямс, 2008. 224 с.

10 Шилдт Г. Библиотека SWING для Java: руководство для начинающих. М.: Вильямс, 2007. 704 с.

11 Шилдт Г. Полный справочник по Java SE 6. М.: Вильямс, 2010. 1040 с.

12 Эккель Б. Философия Java. СПб.: Питер., 2009. 640 с.

Додаток А

Текст програми звіту.

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.formula.ConditionalFormattingEvaluator;
import org.apache.poi.ss.formula.EvaluationConditionalFormatRule;
import org.apache.poi.ss.formula.WorkbookEvaluatorProvider;
import org.apache.poi.ss.usermodel.BuiltinFormats;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellStyle;
import org.apache.poi.ss.usermodel.ColorScaleFormatting;
import org.apache.poi.ss.usermodel.ComparisonOperator;
import org.apache.poi.ss.usermodel.ConditionalFormattingRule;
import org.apache.poi.ss.usermodel.ConditionalFormattingThreshold.RangeType;
import org.apache.poi.ss.usermodel.DataBarFormatting;
import org.apache.poi.ss.usermodel.ExtendedColor;
import org.apache.poi.ss.usermodel.FontFormatting;
import org.apache.poi.ss.usermodel.IconMultiStateFormatting;
import org.apache.poi.ss.usermodel.IconMultiStateFormatting.IconSet;
import org.apache.poi.ss.usermodel.IndexedColors;
import org.apache.poi.ss.usermodel.PatternFormatting;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.SheetConditionalFormatting;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.util.CellRangeAddress;
import org.apache.poi.ss.util.CellReference;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public final class ConditionalFormats {
   private ConditionalFormats() {}
   public static void main(String[] args) throws IOException {
      final boolean isHSSF = args.length > 0 && args[0].equals("-xls");
      try (Workbook wb = isHSSF ? new HSSFWorkbook() : new XSSFWorkbook()) {
         sameCell(wb.createSheet("Same Cell"));
         multiCell(wb.createSheet("MultiCell"));
         overlapping(wb.createSheet("Overlapping"));
         errors(wb.createSheet("Errors"));
         hideDupplicates(wb.createSheet("Hide Dups"));
         formatDuplicates(wb.createSheet("Duplicates"));
         inList(wb.createSheet("In List"));
         expiry(wb.createSheet("Expiry"));
         shadeAlt(wb.createSheet("Shade Alt"));
         shadeBands(wb.createSheet("Shade Bands"));
         iconSets(wb.createSheet("Icon Sets"));
         colourScales(wb.createSheet("Colour Scales"));
         dataBars(wb.createSheet("Data Bars"));
         evaluateRules(wb, "Overlapping");
         String file = "cf-poi.xls";
```

```
            if (wb instanceof XSSFWorkbook) {
                file += "x";
            }
            try (FileOutputStream out = new FileOutputStream(file)) {
                wb.write(out);
            }
            System.out.println("Generated: " + file);
        }
    }
    static void sameCell(Sheet sheet) {
        sheet.createRow(0).createCell(0).setCellValue(84);
        sheet.createRow(1).createCell(0).setCellValue(74);
        sheet.createRow(2).createCell(0).setCellValue(50);
        sheet.createRow(3).createCell(0).setCellValue(51);
        sheet.createRow(4).createCell(0).setCellValue(49);
        sheet.createRow(5).createCell(0).setCellValue(41);
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ConditionalFormattingRule rule1 =
sheetCF.createConditionalFormattingRule(ComparisonOperator.GT, "70");
        PatternFormatting fill1 = rule1.createPatternFormatting();
        fill1.setFillBackgroundColor(IndexedColors.BLUE.index);
        fill1.setFillPattern(PatternFormatting.SOLID_FOREGROUND);
        ConditionalFormattingRule rule2 =
sheetCF.createConditionalFormattingRule(ComparisonOperator.LT, "50");
        PatternFormatting fill2 = rule2.createPatternFormatting();
        fill2.setFillBackgroundColor(IndexedColors.GREEN.index);
        fill2.setFillPattern(PatternFormatting.SOLID_FOREGROUND);
        CellRangeAddress[] regions = {
                CellRangeAddress.valueOf("A1:A6")
        };
        sheetCF.addConditionalFormatting(regions, rule1, rule2);
        sheet.getRow(0).createCell(2).setCellValue("<== Condition 1: Cell Value Is greater than 70 (Blue
Fill)");
        sheet.getRow(4).createCell(2).setCellValue("<== Condition 2: Cell Value Is less than 50 (Green Fill)");
    }
    static void multiCell(Sheet sheet) {
        Row row0 = sheet.createRow(0);
        row0.createCell(0).setCellValue("Units");
        row0.createCell(1).setCellValue("Cost");
        row0.createCell(2).setCellValue("Total");
        Row row1 = sheet.createRow(1);
        row1.createCell(0).setCellValue(71);
        row1.createCell(1).setCellValue(29);
        row1.createCell(2).setCellValue(2059);
        Row row2 = sheet.createRow(2);
        row2.createCell(0).setCellValue(85);
        row2.createCell(1).setCellValue(29);
        row2.createCell(2).setCellValue(2059);
        Row row3 = sheet.createRow(3);
        row3.createCell(0).setCellValue(71);
        row3.createCell(1).setCellValue(29);
        row3.createCell(2).setCellValue(2059);
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
```

```
        ConditionalFormattingRule rule1 = sheetCF.createConditionalFormattingRule("$A2>75");
        PatternFormatting fill1 = rule1.createPatternFormatting();
        fill1.setFillBackgroundColor(IndexedColors.BLUE.index);
        fill1.setFillPattern(PatternFormatting.SOLID_FOREGROUND);
        CellRangeAddress[] regions = {
            CellRangeAddress.valueOf("A2:C4")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheet.getRow(2).createCell(4).setCellValue("<== Condition 1: Formula Is =$B2>75   (Blue Fill)");
    }
    static void overlapping(Sheet sheet) {
        for (int i=0; i<40; i++) {
            int rn = i+1;
            Row r = sheet.createRow(i);
            r.createCell(0).setCellValue("This is row " + rn + " (" + i + ")");
            String str = "";
            if (rn%2 == 0) {
                str = str + "even ";
            }
            if (rn%3 == 0) {
                str = str + "x3 ";
            }
            if (rn%5 == 0) {
                str = str + "x5 ";
            }
            if (rn%10 == 0) {
                str = str + "x10 ";
            }
            if (str.length() == 0) {
                str = "nothing special...";
            }
            r.createCell(1).setCellValue("It is " + str);
        }
        sheet.autoSizeColumn(0);
        sheet.autoSizeColumn(1);
        sheet.getRow(1).createCell(3).setCellValue("Even rows are blue");
        sheet.getRow(2).createCell(3).setCellValue("Multiples of 3 have a grey background");
        sheet.getRow(4).createCell(3).setCellValue("Multiples of 5 are bold");
        sheet.getRow(9).createCell(3).setCellValue("Multiples of 10 are red (beats even)");
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ConditionalFormattingRule rule1 =
sheetCF.createConditionalFormattingRule("MOD(ROW(),10)=0");
        FontFormatting font1 = rule1.createFontFormatting();
        font1.setFontColorIndex(IndexedColors.RED.index);
        ConditionalFormattingRule rule2 =
sheetCF.createConditionalFormattingRule("MOD(ROW(),2)=0");
        FontFormatting font2 = rule2.createFontFormatting();
        font2.setFontColorIndex(IndexedColors.BLUE.index);
        ConditionalFormattingRule rule3 =
sheetCF.createConditionalFormattingRule("MOD(ROW(),5)=0");
        FontFormatting font3 = rule3.createFontFormatting();
        font3.setFontStyle(false, true);
        ConditionalFormattingRule rule4 =
```

```
sheetCF.createConditionalFormattingRule("MOD(ROW(),3)=0");
        PatternFormatting fill4 = rule4.createPatternFormatting();
fill4.setFillBackgroundColor(IndexedColors.GREY_25_PERCENT.index);
        fill4.setFillPattern(PatternFormatting.SOLID_FOREGROUND);
        CellRangeAddress[] regions = {
                CellRangeAddress.valueOf("A1:F41")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheetCF.addConditionalFormatting(regions, rule2);
        sheetCF.addConditionalFormatting(regions, rule3);
        sheetCF.addConditionalFormatting(regions, rule4);
    }
    static void errors(Sheet sheet) {
        sheet.createRow(0).createCell(0).setCellValue(84);
        sheet.createRow(1).createCell(0).setCellValue(0);
sheet.createRow(2).createCell(0).setCellFormula("ROUND(A1/A2,0)");
        sheet.createRow(3).createCell(0).setCellValue(0);
sheet.createRow(4).createCell(0).setCellFormula("ROUND(A6/A4,0)");
        sheet.createRow(5).createCell(0).setCellValue(41);
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ConditionalFormattingRule rule1 = sheetCF.createConditionalFormattingRule("ISERROR(A1)");
        FontFormatting font = rule1.createFontFormatting();
        font.setFontColorIndex(IndexedColors.WHITE.index);
        CellRangeAddress[] regions = {
                CellRangeAddress.valueOf("A1:A6")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheet.getRow(2).createCell(1).setCellValue("<== The error in this cell is hidden. Condition: Formula
Is   =ISERROR(C2)   (White Font)");
        sheet.getRow(4).createCell(1).setCellValue("<== The error in this cell is hidden. Condition: Formula
Is   =ISERROR(C2)   (White Font)");
    }
    static void hideDupplicates(Sheet sheet) {
        sheet.createRow(0).createCell(0).setCellValue("City");
        sheet.createRow(1).createCell(0).setCellValue("Boston");
        sheet.createRow(2).createCell(0).setCellValue("Boston");
        sheet.createRow(3).createCell(0).setCellValue("Chicago");
        sheet.createRow(4).createCell(0).setCellValue("Chicago");
        sheet.createRow(5).createCell(0).setCellValue("New York");
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        // Condition 1: Formula Is   =A2=A1   (White Font)
        ConditionalFormattingRule rule1 = sheetCF.createConditionalFormattingRule("A2=A1");
        FontFormatting font = rule1.createFontFormatting();
        font.setFontColorIndex(IndexedColors.WHITE.index);
        CellRangeAddress[] regions = {
                CellRangeAddress.valueOf("A2:A6")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheet.getRow(1).createCell(1).setCellValue("<== the second (and subsequent) " +
                "occurences of each region name will have white font colour.  " +
                "Condition: Formula Is   =A2=A1   (White Font)");
    }
    static void formatDuplicates(Sheet sheet) {
```

```
        sheet.createRow(0).createCell(0).setCellValue("Code");
        sheet.createRow(1).createCell(0).setCellValue(4);
        sheet.createRow(2).createCell(0).setCellValue(3);
        sheet.createRow(3).createCell(0).setCellValue(6);
        sheet.createRow(4).createCell(0).setCellValue(3);
        sheet.createRow(5).createCell(0).setCellValue(5);
        sheet.createRow(6).createCell(0).setCellValue(8);
        sheet.createRow(7).createCell(0).setCellValue(0);
        sheet.createRow(8).createCell(0).setCellValue(2);
        sheet.createRow(9).createCell(0).setCellValue(8);
        sheet.createRow(10).createCell(0).setCellValue(6);
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ConditionalFormattingRule rule1 =
sheetCF.createConditionalFormattingRule("COUNTIF($A$2:$A$11,A2)>1");
        FontFormatting font = rule1.createFontFormatting();
        font.setFontStyle(false, true);
        font.setFontColorIndex(IndexedColors.BLUE.index);
        CellRangeAddress[] regions = {
                CellRangeAddress.valueOf("A2:A11")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheet.getRow(2).createCell(1).setCellValue("<== Duplicates numbers in the column are highlighted.
" +
                "Condition: Formula Is =COUNTIF($A$2:$A$11,A2)>1   (Blue Font)");
    }
    static void inList(Sheet sheet) {
        sheet.createRow(0).createCell(0).setCellValue("Codes");
        sheet.createRow(1).createCell(0).setCellValue("AA");
        sheet.createRow(2).createCell(0).setCellValue("BB");
        sheet.createRow(3).createCell(0).setCellValue("GG");
        sheet.createRow(4).createCell(0).setCellValue("AA");
        sheet.createRow(5).createCell(0).setCellValue("FF");
        sheet.createRow(6).createCell(0).setCellValue("XX");
        sheet.createRow(7).createCell(0).setCellValue("CC");
        sheet.getRow(0).createCell(2).setCellValue("Valid");
        sheet.getRow(1).createCell(2).setCellValue("AA");
        sheet.getRow(2).createCell(2).setCellValue("BB");
        sheet.getRow(3).createCell(2).setCellValue("CC");
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ConditionalFormattingRule rule1 =
sheetCF.createConditionalFormattingRule("COUNTIF($C$2:$C$4,A2)");
        PatternFormatting fill1 = rule1.createPatternFormatting();
        fill1.setFillBackgroundColor(IndexedColors.LIGHT_BLUE.index);
        fill1.setFillPattern(PatternFormatting.SOLID_FOREGROUND);
        CellRangeAddress[] regions = {
                CellRangeAddress.valueOf("A2:A8")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheet.getRow(2).createCell(3).setCellValue("<== Use Excel conditional formatting to highlight items
that are in a list on the worksheet");
    }
    static void expiry(Sheet sheet) {
        CellStyle style = sheet.getWorkbook().createCellStyle();
```

```
        style.setDataFormat((short)BuiltinFormats.getBuiltinFormat("d-mmm"));
        sheet.createRow(0).createCell(0).setCellValue("Date");
        sheet.createRow(1).createCell(0).setCellFormula("TODAY()+29");
        sheet.createRow(2).createCell(0).setCellFormula("A2+1");
        sheet.createRow(3).createCell(0).setCellFormula("A3+1");
        for(int rownum = 1; rownum <= 3; rownum++) {
           sheet.getRow(rownum).getCell(0).setCellStyle(style);
        }
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ConditionalFormattingRule rule1 = sheetCF.createConditionalFormattingRule("AND(A2-
TODAY()>=0,A2-TODAY()<=30)");
        FontFormatting font = rule1.createFontFormatting();
        font.setFontStyle(false, true);
        font.setFontColorIndex(IndexedColors.BLUE.index);
        CellRangeAddress[] regions = {
              CellRangeAddress.valueOf("A2:A4")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheet.getRow(0).createCell(1).setCellValue("Dates within the next 30 days are highlighted");
    }
    static void shadeAlt(Sheet sheet) {
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ConditionalFormattingRule rule1 = sheetCF.createConditionalFormattingRule("MOD(ROW(),2)");
        PatternFormatting fill1 = rule1.createPatternFormatting();
        fill1.setFillBackgroundColor(IndexedColors.LIGHT_GREEN.index);
        fill1.setFillPattern(PatternFormatting.SOLID_FOREGROUND);
        CellRangeAddress[] regions = {
              CellRangeAddress.valueOf("A1:Z100")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheet.createRow(0).createCell(1).setCellValue("Shade Alternating Rows");
        sheet.createRow(1).createCell(1).setCellValue("Condition: Formula Is  =MOD(ROW(),2)   (Light Green
Fill)");
    }
    static void shadeBands(Sheet sheet) {
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ConditionalFormattingRule rule1 = sheetCF.createConditionalFormattingRule("MOD(ROW(),6)<3");
        PatternFormatting fill1 = rule1.createPatternFormatting();
fill1.setFillBackgroundColor(IndexedColors.GREY_25_PERCENT.index);
        fill1.setFillPattern(PatternFormatting.SOLID_FOREGROUND);
        CellRangeAddress[] regions = {
              CellRangeAddress.valueOf("A1:Z100")
        };
        sheetCF.addConditionalFormatting(regions, rule1);
        sheet.createRow(0).createCell(1).setCellValue("Shade Bands of Rows");
        sheet.createRow(1).createCell(1).setCellValue("Condition: Formula Is  =MOD(ROW(),6)<2   (Light
Grey Fill)");
    }
    static void iconSets(Sheet sheet) {
        sheet.createRow(0).createCell(0).setCellValue("Icon Sets");
        Row r = sheet.createRow(1);
        r.createCell(0).setCellValue("Reds");
        r.createCell(1).setCellValue(0);
```

```
        r.createCell(2).setCellValue(0);
        r.createCell(3).setCellValue(0);
        r = sheet.createRow(2);
        r.createCell(0).setCellValue("Yellows");
        r.createCell(1).setCellValue(5);
        r.createCell(2).setCellValue(5);
        r.createCell(3).setCellValue(5);
        r = sheet.createRow(3);
        r.createCell(0).setCellValue("Greens");
        r.createCell(1).setCellValue(10);
        r.createCell(2).setCellValue(10);
        r.createCell(3).setCellValue(10);
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
       CellRangeAddress[] regions = { CellRangeAddress.valueOf("B1:B4") };
        ConditionalFormattingRule rule1 =
sheetCF.createConditionalFormattingRule(IconSet.GYR_3_TRAFFIC_LIGHTS);
        IconMultiStateFormatting im1 = rule1.getMultiStateFormatting();
        im1.getThresholds()[0].setRangeType(RangeType.MIN);
        im1.getThresholds()[1].setRangeType(RangeType.PERCENT);
        im1.getThresholds()[1].setValue(33d);
        im1.getThresholds()[2].setRangeType(RangeType.MAX);
        sheetCF.addConditionalFormatting(regions, rule1);
        regions = new CellRangeAddress[] { CellRangeAddress.valueOf("C1:C4") };
        ConditionalFormattingRule rule2 =
sheetCF.createConditionalFormattingRule(IconSet.GYR_3_FLAGS);
        IconMultiStateFormatting im2 = rule1.getMultiStateFormatting();
        im2.getThresholds()[0].setRangeType(RangeType.PERCENT);
        im2.getThresholds()[0].setValue(0d);
        im2.getThresholds()[1].setRangeType(RangeType.PERCENT);
        im2.getThresholds()[1].setValue(33d);
        im2.getThresholds()[2].setRangeType(RangeType.PERCENT);
        im2.getThresholds()[2].setValue(67d);
        sheetCF.addConditionalFormatting(regions, rule2);
        regions = new CellRangeAddress[] { CellRangeAddress.valueOf("D1:D4") };
        ConditionalFormattingRule rule3 =
sheetCF.createConditionalFormattingRule(IconSet.GYR_3_SYMBOLS_CIRCLE);
        IconMultiStateFormatting im3 = rule1.getMultiStateFormatting();
        im3.setIconOnly(true);
        im3.getThresholds()[0].setRangeType(RangeType.MIN);
        im3.getThresholds()[1].setRangeType(RangeType.NUMBER);
        im3.getThresholds()[1].setValue(3d);
        im3.getThresholds()[2].setRangeType(RangeType.NUMBER);
        im3.getThresholds()[2].setValue(7d);
        sheetCF.addConditionalFormatting(regions, rule3);
    }
    static void colourScales(Sheet sheet) {
        sheet.createRow(0).createCell(0).setCellValue("Colour Scales");
        Row r = sheet.createRow(1);
        r.createCell(0).setCellValue("Red-Yellow-Green");
        for (int i=1; i<=7; i++) {
            r.createCell(i).setCellValue((i-1)*5.0);
        }
        r = sheet.createRow(2);
```

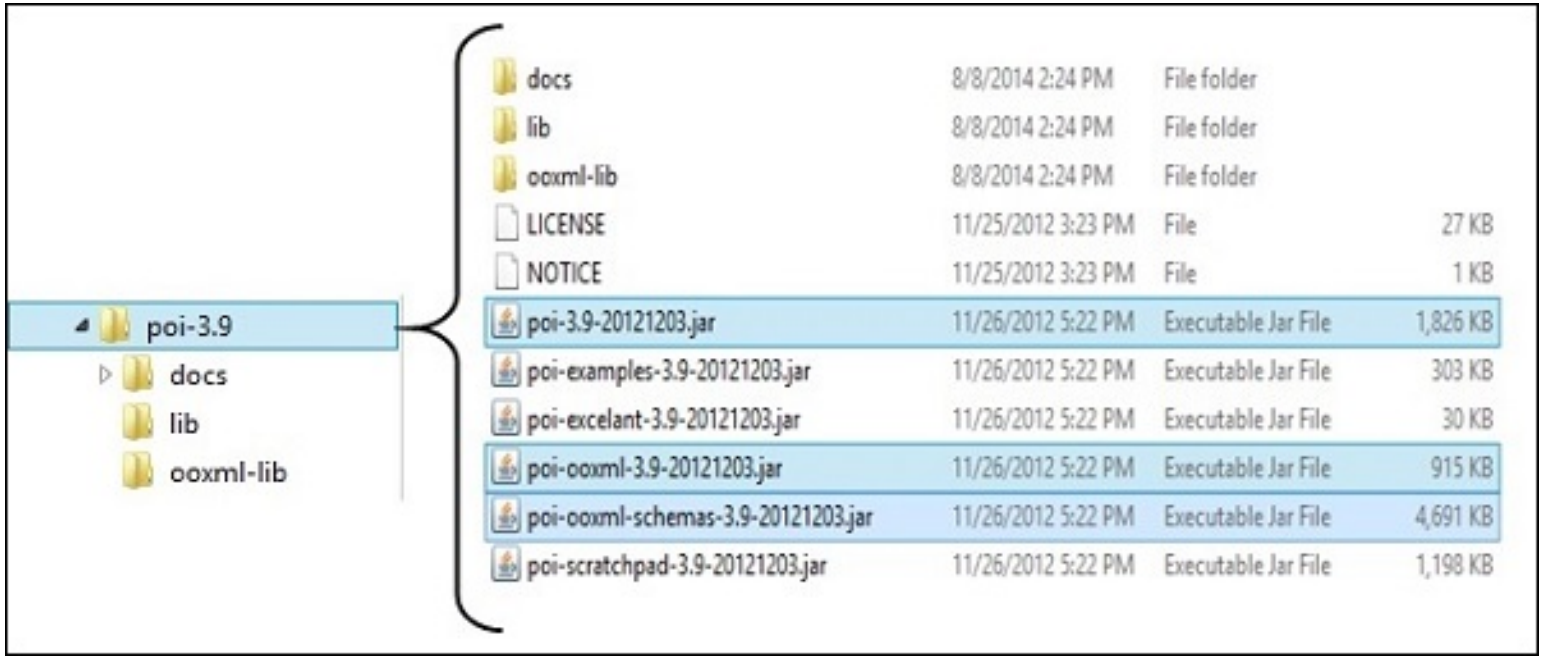
```
    r.createCell(0).setCellValue("Red-White-Blue");
    for (int i=1; i<=9; i++) {
       r.createCell(i).setCellValue((i-1)*5.0);
    }
    r = sheet.createRow(3);
    r.createCell(0).setCellValue("Blue-Green");
    for (int i=1; i<=16; i++) {
       r.createCell(i).setCellValue((i-1));
    }
    sheet.setColumnWidth(0, 5000);
    SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
    CellRangeAddress[] regions = { CellRangeAddress.valueOf("B2:H2") };
    ConditionalFormattingRule rule1 =
          sheetCF.createConditionalFormattingColorScaleRule();
    ColorScaleFormatting cs1 = rule1.getColorScaleFormatting();
    cs1.getThresholds()[0].setRangeType(RangeType.MIN);
    cs1.getThresholds()[1].setRangeType(RangeType.PERCENTILE);
    cs1.getThresholds()[1].setValue(50d);
    cs1.getThresholds()[2].setRangeType(RangeType.MAX);
    ((ExtendedColor)cs1.getColors()[0]).setARGBHex("FFF8696B");
    ((ExtendedColor)cs1.getColors()[1]).setARGBHex("FFFFEB84");
    ((ExtendedColor)cs1.getColors()[2]).setARGBHex("FF63BE7B");
    sheetCF.addConditionalFormatting(regions, rule1);
    regions = new CellRangeAddress[] { CellRangeAddress.valueOf("B3:J3") };
    ConditionalFormattingRule rule2 =
          sheetCF.createConditionalFormattingColorScaleRule();
    ColorScaleFormatting cs2 = rule2.getColorScaleFormatting();
    cs2.getThresholds()[0].setRangeType(RangeType.MIN);
    cs2.getThresholds()[1].setRangeType(RangeType.PERCENTILE);
    cs2.getThresholds()[1].setValue(50d);
    cs2.getThresholds()[2].setRangeType(RangeType.MAX);
    ((ExtendedColor)cs2.getColors()[0]).setARGBHex("FFF8696B");
    ((ExtendedColor)cs2.getColors()[1]).setARGBHex("FFFCFCFF");
    ((ExtendedColor)cs2.getColors()[2]).setARGBHex("FF5A8AC6");
    sheetCF.addConditionalFormatting(regions, rule2);
    regions = new CellRangeAddress[] { CellRangeAddress.valueOf("B4:Q4") };
    ConditionalFormattingRule rule3=
          sheetCF.createConditionalFormattingColorScaleRule();
    ColorScaleFormatting cs3 = rule3.getColorScaleFormatting();
    cs3.setNumControlPoints(2);
    cs3.getThresholds()[0].setRangeType(RangeType.MIN);
    cs3.getThresholds()[1].setRangeType(RangeType.MAX);
    ((ExtendedColor)cs3.getColors()[0]).setARGBHex("FF5A8AC6");
    ((ExtendedColor)cs3.getColors()[1]).setARGBHex("FF63BE7B");
    sheetCF.addConditionalFormatting(regions, rule3);
 }
 static void dataBars(Sheet sheet) {
    sheet.createRow(0).createCell(0).setCellValue("Data Bars");
    Row r = sheet.createRow(1);
    r.createCell(1).setCellValue("Green Positive");
    r.createCell(2).setCellValue("Blue Mix");
    r.createCell(3).setCellValue("Red Negative");
    r = sheet.createRow(2);
```

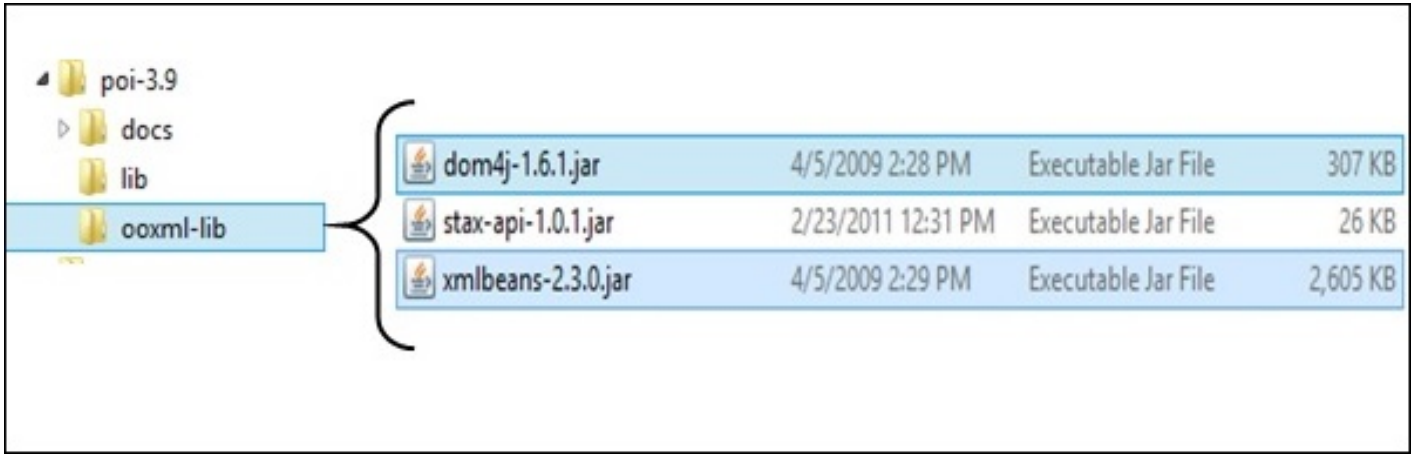
```
        r.createCell(1).setCellValue(0);
        r.createCell(2).setCellValue(0);
        r.createCell(3).setCellValue(0);
        r = sheet.createRow(3);
        r.createCell(1).setCellValue(5);
        r.createCell(2).setCellValue(-5);
        r.createCell(3).setCellValue(-5);
        r = sheet.createRow(4);
        r.createCell(1).setCellValue(10);
        r.createCell(2).setCellValue(10);
        r.createCell(3).setCellValue(-10);
        r = sheet.createRow(5);
        r.createCell(1).setCellValue(5);
        r.createCell(2).setCellValue(5);
        r.createCell(3).setCellValue(-5);
        r = sheet.createRow(6);
        r.createCell(1).setCellValue(20);
        r.createCell(2).setCellValue(-10);
        r.createCell(3).setCellValue(-20);
        sheet.setColumnWidth(0, 3000);
        sheet.setColumnWidth(1, 5000);
        sheet.setColumnWidth(2, 5000);
        sheet.setColumnWidth(3, 5000);
        SheetConditionalFormatting sheetCF = sheet.getSheetConditionalFormatting();
        ExtendedColor color = sheet.getWorkbook().getCreationHelper().createExtendedColor();
        color.setARGBHex("FF63BE7B");
       CellRangeAddress[] regions = { CellRangeAddress.valueOf("B2:B7") };
        ConditionalFormattingRule rule1 = sheetCF.createConditionalFormattingRule(color);
        DataBarFormatting db1 = rule1.getDataBarFormatting();
        db1.getMinThreshold().setRangeType(RangeType.MIN);
        db1.getMaxThreshold().setRangeType(RangeType.MAX);
        sheetCF.addConditionalFormatting(regions, rule1);
        color = sheet.getWorkbook().getCreationHelper().createExtendedColor();
        color.setARGBHex("FF5A8AC6");
        regions = new CellRangeAddress[] { CellRangeAddress.valueOf("C2:C7") };
        ConditionalFormattingRule rule2 = sheetCF.createConditionalFormattingRule(color);
        DataBarFormatting db2 = rule2.getDataBarFormatting();
        db2.getMinThreshold().setRangeType(RangeType.MIN);
        db2.getMaxThreshold().setRangeType(RangeType.MAX);
        sheetCF.addConditionalFormatting(regions, rule2);
        color = sheet.getWorkbook().getCreationHelper().createExtendedColor();
        color.setARGBHex("FFF8696B");
        regions = new CellRangeAddress[] { CellRangeAddress.valueOf("D2:D7") };
        ConditionalFormattingRule rule3 = sheetCF.createConditionalFormattingRule(color);
        DataBarFormatting db3 = rule3.getDataBarFormatting();
        db3.getMinThreshold().setRangeType(RangeType.MIN);
        db3.getMaxThreshold().setRangeType(RangeType.MAX);
        sheetCF.addConditionalFormatting(regions, rule3);
    }
    static void evaluateRules(Workbook wb, String sheetName) {
        final WorkbookEvaluatorProvider wbEvalProv = (WorkbookEvaluatorProvider)
wb.getCreationHelper().createFormulaEvaluator();
```

```
        final ConditionalFormattingEvaluator cfEval = new ConditionalFormattingEvaluator(wb,
wbEvalProv);
        cfEval.clearAllCachedValues();
        final Sheet sheet = wb.getSheet(sheetName);
        for (Row r : sheet) {
           for (Cell c : r) {
              final List<EvaluationConditionalFormatRule> rules = cfEval.getConditionalFormattingForCell(c);
              if (rules == null || rules.isEmpty()) {
                 continue;
              }
              final CellReference ref = ConditionalFormattingEvaluator.getRef(c);
              if (rules.isEmpty()) {
                 continue;
              }
              System.out.println("\n"
               + ref.formatAsString()
               + " has conditional formatting.");
              for (EvaluationConditionalFormatRule rule : rules) {
                 ConditionalFormattingRule cf = rule.getRule();
                 StringBuilder b = new StringBuilder();
                 b.append("\tRule ")
                  .append(rule.getFormattingIndex())
                  .append(": ");
                 if (cf.getColorScaleFormatting() != null) {
             b.append("\n\t\tcolor scale (caller must calculate bucket)");
                 }
                 if (cf.getDataBarFormatting() != null) {
             b.append("\n\t\tdata bar (caller must calculate bucket)");
                 }
                 if (cf.getMultiStateFormatting() != null) {
            b.append("\n\t\ticon set (caller must calculate icon bucket)");
                 }
                 if (cf.getPatternFormatting() != null) {
                    final PatternFormatting fill = cf.getPatternFormatting();
                    b.append("\n\t\tfill pattern ")
                     .append(fill.getFillPattern())
                     .append(" color index ")
                     .append(fill.getFillBackgroundColor());
                 if (cf.getFontFormatting() != null) {
                    final FontFormatting ff = cf.getFontFormatting();
                    b.append("\n\t\tfont format ")
                     .append("color index ")
                     .append(ff.getFontColorIndex());
                    if (ff.isBold()) {
                       b.append(" bold");
                    }
                    if (ff.isItalic()) {
                       b.append(" italic");
                    }
                    b.append(" underline index ")
                     .append(ff.getUnderlineType());
                 }
                 System.out.println(b);
```

```
            }
        }
    }
  }
}
```

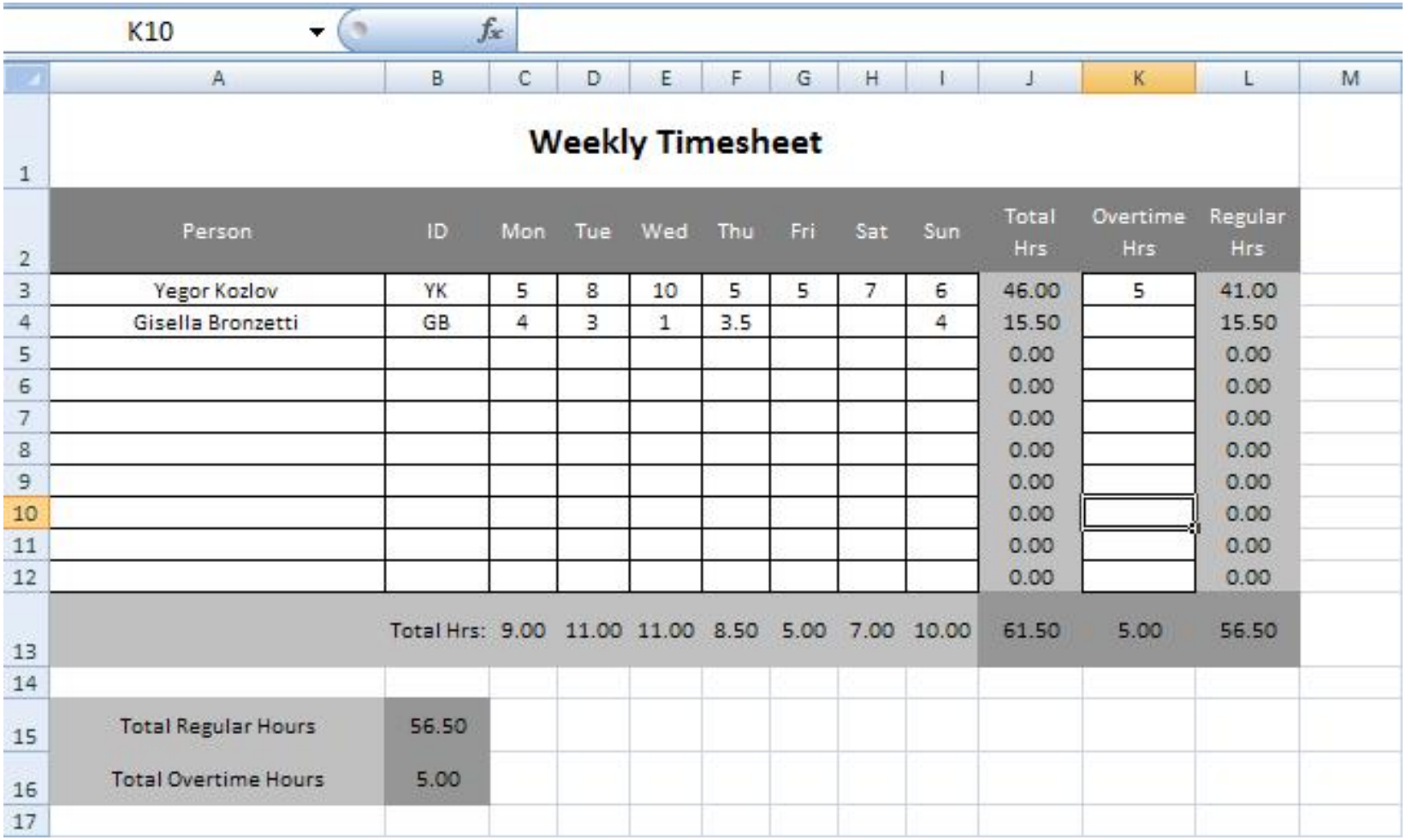
КОПІЇ ДЕМОНСТРАЦІЙНИХ АРКУШІВ





					Розробка програми для формування звітів у MS Excel використанням мови Java.
Зм.	Аркуш	№ докум.	Підпис	Дата	Структура бібліотеки POI
Розроб.		Пішхавка	В.В.		
Керівник		Селемонавічус	А.А.		
Реценз.					
Н. контр.		Кужій	Л.І		
Затверд.					

Літера	Маса	Масштаб
Аркуш	Аркушів	



K10

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Weekly Timesheet**												
2	Person	ID	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total Hrs	Overtime Hrs	Regular Hrs	
3	Yegor Kozlov	YK	5	8	10	5	5	7	6	46.00	5	41.00	
4	Gisella Bronzetti	GB	4	3	1	3.5			4	15.50		15.50	
5										0.00		0.00	
6										0.00		0.00	
7										0.00		0.00	
8										0.00		0.00	
9										0.00		0.00	
10										0.00		0.00	
11										0.00		0.00	
12										0.00		0.00	
13		Total Hrs:	9.00	11.00	11.00	8.50	5.00	7.00	10.00	61.50	5.00	56.50	
14													
15	Total Regular Hours	56.50											
16	Total Overtime Hours	5.00											
17													

Зм.	Аркуш	№ докум.	Підпис	Дата	Розробка програми для формування звітів у MS Excel використанням мови Java.		
Розроб.		Пішхавка В.В.			Розробка програми для генерації звітів	Літера	Маса	Масштаб
Керівник		Селемонавічус А.А.						
Реценз.						Аркуш	Аркушів	
Н. контр.		Кужій Л.І.						
Затверд.								

	A	B	C	D	E	F
1	Ім'я	Номер угоди	Дата видачі	Дата погашення	Тип кредиту	Сума
2	Куліш	№4572	23.11.2007	23.11.2012	2 car loans	4 320,13
3	Палій	№5261	26.09.2008	26.09.2013	2 car loans	22 070,27
4	Попова	№5138	08.08.2008	07.08.2009	3 consumer loans	3 893,69
5	Чорний	№4877	25.04.2008	24.04.2019	3 consumer loans	1 108 305,83
6	Шевченко	№4888	05.05.2008	04.05.2009	3 consumer loans	0,00
7	Мельник	№4397	02.08.2007	02.08.2022	1 mortgage loans	820 497,68
8	Шевченко	№4942	27.05.2008	26.05.2023	1 mortgage loans	369 704,50
9	Бойко	№5128	04.08.2008	04.08.2033	1 mortgage loans	38 929,53
10	Коваленко	№4746	28.02.2008	27.02.2013	3 consumer loans	38 095,94
11	Бондаренко	№4863	21.04.2008	21.04.2015	2 car loans	54 958,42
12	Ткаченко	№4313	05.06.2007	04.06.2012	2 car loans	1 212,13
13	Ковальчук	№4955	30.05.2008	29.05.2014	2 car loans	14 688,38
14	Кравченко	№5398	26.08.2009	25.08.2014	2 car loans	27 775,46
15	Олійник	№5294	09.10.2008	09.10.2015	2 car loans	41 683,05
16	Шевчук	№4975	05.06.2008	05.06.2015	2 car loans	79 218,73
17	Коваль	№5184	28.08.2008	28.08.2015	2 car loans	80 999,39
18	Поліщук	№4957	30.05.2008	29.05.2015	2 car loans	55 700,93
19	Бондар	№5301	10.10.2008	10.10.2023	1 mortgage loans	476 308,46

					Розробка програми для формування звітів у MS Excel використанням мови Java.
Зм.	Аркуш	№ докум.	Підпис	Дата	Приклади роботи розробленої програми
Розроб.		Пішхавка	В.В.		
Керівник		Селемонавічус	А.А.		
Реценз.					
Н. контр.		Кужій	Л.І.		
Затверд.					

Літера	Маса	Масштаб
Аркуш	Аркушів	

Витрати на розробку та впровадження проектного рішення

Найменування елементів витрат	Сума витрат, грн.
Витрати на оплату праці	3 446,83
Нарахування на зарплату	412,59
Витрати на куповані вироби	572
Накладні витрати	1 034,05
Інші витрати	607,27
Витрати на налагодження та дослідну експлуатацію	1 620
Всього:	7 552,74

Зм.	Аркуш	№ докум.	Підпис	Дата	Розробка програми для формування звітів у MS Excel з використанням мови Java.		
					Кошторис витрат на розробку проектного рішення	Літера	Маса	Масштаб
Розроб.		Пішхавка В.В.						
Керівник		Селемонавічус А.А.				Аркуш	Аркушів	
Реценз.								
Н. контр.		Кукжій Л.І.						
Затверд.								