Refactoring

Nota: Lo trate de resolver masomenos como estaba en la explicacion de la primer fecha.
En el parcial lo resolvi aplicando template, y strategy.

Tambien se puede aplicar strategy, es totalmente valido.
Hay que prestar atencion al desarrollo de cada paso del refactoring, es importante detallar paso a paso.

Inciso 1 Enumerar los code smell:

> 1: Long Method / Metodo Largo. : Lineas 5 - 44
> 2: Switch statement / sentencia switch : Lineas 6 y 25
> 3: Duplicated code / codigo duplicado: Lineas 8-13 y 27-32
> 4: Reinventar la rueda: : Linea 11-13 y 30-32

Inciso 2 Indicar los refactorings para solucionarlos y explique los pasos necesarios para realizar los refactorings.

> Para el code smell numero 3 (codigo duplicado), lo que hacemos es extraer metodo de linea 8-13.

pasos:

- Creamos un metodo con encabezado 'private void crearDocumento(Document document)'
- Copiamos el codigo de las lineas 8-13 al metodo crearDocumento
- Reemplazamos las lineas 8-13 con un llamado al metodo crearDocumento
- Reemplazamos las lineas 27-32 con un llamado a crearDocumento

> Para el code smell numero 2 (switch statement), aplicamos replace conditional with polymorphism

pasos:

- Creamos dos subclases de ReportGenerator: PdfReportGenerator y XlsReporGenerator
- En cada subclase creamos el mensaje generateReport y copiamos las lineas del condicional correspondiente:
  - Para PdfReporGenerator copiamos las lineas 7-24
  - Para XlsReportGenerator copiamos las lineas 26-43

- Luego de copiar el codigo, borramos la condicion y el codigo de la rama del condicional en la superclase.
- Hacemos que el metodo generateReport de la clase ReportGenerator sea abstracto
- Eliminamos la variable de instancia 'tipo'

> Luego de crear la jerarquia, podemos ver que las subclases realizan pasos similares en el mismo orden en el metodo generateReport, pero los pasos son distintos segun la subclase. Por lo que refactorizamos aplicando Template Method

pasos:

- Extraer las partes de generateRepor en metodos (misma firma y cuerpo en las subclases)
  - Se crean los metodos:
    - createDocument (Document document) ((ESTE METODO YA SE HABIA EXTRAIDO))
    - saveExportedFile(DocumentFile docFile) lineas 22 y 43 del ReportGenerator
    - configureMetadata(Document document, DocumentFile docFile) para encapsular la configuracion de los metadatos (linea 14-16 para PDF y 33-35 para XLS)
    - setContent(Document document DocumentFile docFile) para setear los contenidos especificos. (linea 19-21 para PDF y 38-40 para Xls)
- Una vez que extraimos los metodos en las subclases, vemos que nos queda codigo y metodos duplicados en las subclases, entonces aplicamos lo siguiente:
  - Pull up para metodos identicos:
    - configureMetadata(Document docment documentFile)
    - setContent(Document document,DocumentFile docFile)
    - saveExportedFile(DocumentFile docFile)
  - Definimos los metodos abstractos en las superclase
    - setContent(Document document, DocumentFile docFile)
    - configureMetadata(Document document, DocumentFile docFile)
- Entonces en la superclase (clase abstracta ReportGenerator) nos queda lo siguiente:
  - generateReport es el template method
  - createDocument, configureMetadata,setContent,saveExported son los 'pasos' a seguir del metodo template generateReport.
  - Las subclsaes deberian implementar los metodos abstractos de:
    - setContent(Document document, DocumentFile docFile)
    - configureMetadata(Document document, DocumentFile docFile)

> Refactoring, eliminar parametros y poner variables de instancia en la superclase

pasos:

- Creamos dos variables de instancia en la clase abstracta, con modificador de acceso protected, para que las subclases puedan accederla:
  - protected Document document
    - Esta variable se debe setear desde constructor
  - protected DocumentFile docFile
    - Esta variable se crea la instancia en el metodo createDocumentFile
- Eliminamos los parametros de los metodos:
  - createDocument
  - configureMetadata
  - setContent
  - saveExported

> Para el code Smell numero 4:
> reemplazamos el for del metodo createDocument, por pipeline de java

Codigo final:

```
public abstract class ReportGenerator {
    protected Document document;
    protected DocumentFile docFile;
    public ReportGenerator(Document document)
    { this.document = document;  }

    // Metodo Template
    public final void generateReport()
    {
        this.createDocumentFile();
        this.configureMetadata();
        this.setContent();
        this.saveExportedFile();
    }

    protected abstract void configureMetadata();
    protected abstract void setContent();
    protected void createDocumentFile() {
        docFile = new DocumentFile();
        docFile.setTitle(document.getTitle());
        String authors = document.getAuthors().stream()

.collect(java.util.stream.Collectors.joining(", "));
        docFile.setAuthor(authors);
    }
```

```
    // metodo que ya venia con la clase ReportGenerator
    protected void saveExportedFile ()
    {
        .........
    }
}


public class PdfReportGenerator extends ReportGenerator {
    public PdfReportGenerator (Document document)
    { super(document) ; }
    @Override
    protected void configureMetadata()
    {
        this.docFile.contentType("application/pdf");
        this.docFile.setPageSize("A4");
    }

    @Override
    protected void setContent()
    {
        PDFExporter exporter = new PDFExporter();
        byte[] content = exporter.generatePDFFile(this.document);
        this.docFile.setContent(content);
    }
}



public class XlsReportGenerator extends ReporGenerator {
    public XlsReportGenerator (Document document)
    { super(document) ;}
    @Override
    protected void configureMetadata()
    {
        this.docFile.contentType("application/vnd.ms-excel");
        this.docFile.setSheetName(this.document.getSubtittle())
    }

    @Override
    protected void setContent()
    {
        ExcelWriter writer = new ExcelWriter();
        byte[] content = writer.generateExcelFile(this.document);
        this.docFile.setContent(content);
```

```
    }
}
```