**Exercise 1: Implementing the Singleton Pattern**

**Program:**

public class Logger {

   private static Logger instance;

   private Logger() {

      System.out.println("Logger Initialized");

   }

   public static Logger getInstance() {

      if (instance == null) {

         instance = new Logger();

      }

      return instance;

   }

   public static void main(String[] var0) {

      Logger var1 = getInstance();

      Logger var2 = getInstance();

      if (var1 == var2) {

         System.out.println("Both l1 and l2 are same");

      } else {

         System.out.println("Different instances!");

      }

   }

}

public class Logger {

    private static Logger instance;

    private Logger(){

        System.out.println("Logger Initialized");

    }

    public static Logger getInstance(){

        if(instance == null){

            instance = new Logger();

        }

        return instance;

    }

    public static void main(String[] args){

        Logger l1=Logger.getInstance();

        Logger l2=Logger.getInstance();

        if(l1 == l2){

        System.out.println("Both l1 and l2 are same");

        }else{

        System.out.println("Different instances!");

        }

    }

}



**Exercise 2: Implementing the Factory Method Pattern**

**Program:**

public class FactoryMethodPatternExample {

   public FactoryMethodPatternExample() {

   }

   public static void main(String[] var0) {

      WordDocumentFactory var1 = new WordDocumentFactory();

      PDFDocumentFactory var2 = new PDFDocumentFactory();

      ExcelDocumentFactory var3 = new ExcelDocumentFactory();

      Document var4 = var1.createDocument();

      Document var5 = var2.createDocument();

      Document var6 = var3.createDocument();

      var4.openDoc();

      var4.readDoc();

      var4.closeDoc();

      var5.openDoc();

      var5.readDoc();

      var5.closeDoc();

      var6.openDoc();

      var6.readDoc();

      var6.closeDoc();

   }

}

**ConcreteWordDocument.class**

**Program:**

class ConcreteWordDocument extends Document {

   ConcreteWordDocument() {

   }

   public void openDoc() {

      System.out.println("Opening Word Document");

   }

   public void readDoc() {

      System.out.println("Reading Word Document");

   }

   public void closeDoc() {

      System.out.println("Closing Word Document");

   }

}

**Concrete PDFDocument.class**

**Program:**

class ConcretePDFDocument extends Document {

   ConcretePDFDocument() {

   }

   public void openDoc() {

      System.out.println("Opening PDF Document");

   }

   public void readDoc() {

      System.out.println("Reading PDF Document");

   }

   public void closeDoc() {

      System.out.println("Closing PDF Document");

   }

}

**Concrete ExcelDocument.class**

**Program:**

class ConcreteExcelDocument extends Document {

ConcreteExcelDocument() {

}

public void openDoc() {

System.out.println("Opening Excel Document");

}

public void readDoc() {

System.out.println("Reading Excel Document");

}

public void closeDoc() {

System.out.println("Closing Excel Document");

}

}

**DocumentFactory.class**

**Program:**

abstract class DocumentFactory {

   DocumentFactory() {

   }

   abstract Document createDocument();

}

**ExcelDocumentFactory.class**

**Program:**

class ExcelDocumentFactory extends DocumentFactory {

   ExcelDocumentFactory() {

   }

   public Document createDocument() {

      return new ConcreteExcelDocument();

   }

}

**PDFDocumentFactory.class**

**Program:**

class PDFDocumentFactory extends DocumentFactory {

   PDFDocumentFactory() {

   }

   public Document createDocument() {

      return new ConcretePDFDocument();

   }

}

**WordDocumentFactory.class**

**Program:**

class WordDocumentFactory extends DocumentFactory {

   WordDocumentFactory() {

   }

   public Document createDocument() {

      return new ConcreteWordDocument();

   }

}

