

## Week 1 Assignment:

### Design Patterns and Principles:

#### Exercise 1: Implementing the Singleton Pattern

##### Logger.java:

```
public class Logger {
    private static Logger instance;

    private Logger() {
        System.out.println("Logger constructor is initiated");
    }

    public static Logger getInstance() {
        if (instance == null) {
            instance = new Logger();
        }
        return instance;
    }

    public void getLog() {
        System.out.println("Logging data...");
    }
}
```

##### Main.java:

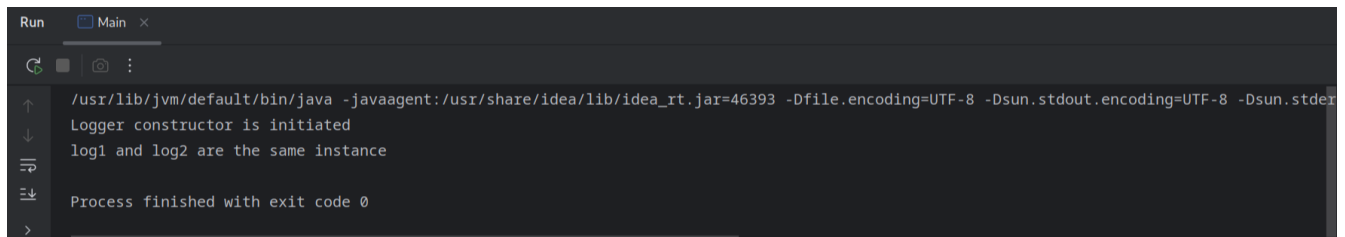
```
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/>
or
// click the <icon src="AllIcons.Actions.Execute"/> icon in
the gutter.
public class Main {
    public static void main(String[] args) {
        Logger log1 = Logger.getInstance();
        Logger log2 = Logger.getInstance();
    }
}
```

```

        if (log1 == log2) {
            System.out.println("log1 and log2 are the same
instance");
        } else {
            System.out.println("log1 and log2 are not the
same instance");
        }
    }
}

```

## Output:



```

Run Main x
/usr/lib/jvm/default/bin/java -javaagent:/usr/share/idea/lib/idea_rt.jar=46393 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
Logger constructor is initiated
log1 and log2 are the same instance
Process finished with exit code 0

```

## Exercise 2: Implementing the Factory Method Pattern

### Document.java:

```

public interface Document {
    public void getDocument();
}

```

### WordDocument.java:

```

public class WordDocument implements Document{
    public void getDocument() {
        System.out.println("This is a word document");
    }
}

```

### PdfDocument.java:

```

public class PdfDocument implements Document{
    public void getDocument() {
        System.out.println("This is a pdf document");
    }
}

```

### ExcelDocument.java:

```
public class ExcelDocument implements Document{
    public void getDocument() {
        System.out.println("This is an excel document");
    }
}
```

### DocumentFactory.java:

```
public interface DocumentFactory {
    public Document createDocument();
}
```

### WordDocumentFactory:

```
public class WordDocumentFactory implements
DocumentFactory{
    public Document createDocument() {
        return new WordDocument();
    }
}
```

### PdfDocumentFactory:

```
public class PdfDocumentFactory implements DocumentFactory{
    public Document createDocument() {
        return new PdfDocument();
    }
}
```

### ExcelDocumentFactory.java:

```
public class ExcelDocumentFactory implements
DocumentFactory{
    public Document createDocument() {
        return new ExcelDocument();
    }
}
```

### Main.java:

```
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/>
or
// click the <icon src="AllIcons.Actions.Execute"/> icon in
the gutter.
```

```

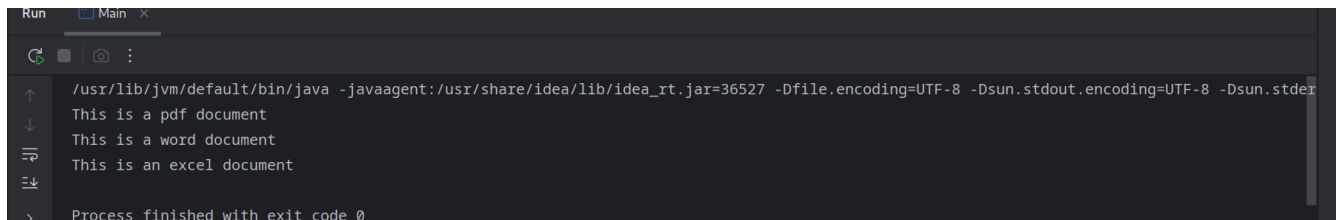
public class Main {
    public static void main(String[] args) {
        DocumentFactory pdfFactory = new
PdfDocumentFactory();
        DocumentFactory wordFactory = new
WordDocumentFactory();
        DocumentFactory excelFactory = new
ExcelDocumentFactory();

        Document pdf = pdfFactory.createDocument();
        Document word = wordFactory.createDocument();
        Document excel = excelFactory.createDocument();

        pdf.getDocument();
        word.getDocument();
        excel.getDocument();
    }
}

```

## Output:



```

Run Main x
/usr/lib/jvm/default/bin/java -javaagent:/usr/share/idea/lib/idea_rt.jar=36527 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
This is a pdf document
This is a word document
This is an excel document
Process finished with exit code 0

```

## Data Structures and Algorithms:

### Excercise 2: E-commerce Platform Search Function

#### Product.java:

```

import java.util.ArrayList;
import java.util.List;

public class Product {
    private List<String> productId;
    private List<String> productName;
}

```

```

    private List<String> category;

    Product(List<String> productId, List<String>
productName, List<String> category) {
        this.productId = productId;
        this.productName = productName;
        this.category = category;
    }

    public int searchProductId(String key) {
        return Search.linearSearch(productId, key);
    }

    public int searchProductName(String key) {
        return Search.binarySearch(productName, key);
    }

    public int searchCategory(String key) {
        return Search.binarySearch(category, key);
    }
}

```

### Search.java:

```

import java.util.List;

public class Search {
    public static int linearSearch(List<String> array,
String key) {
        for (int i=0; i<array.size(); i++) {
            if (array.get(i) == key) {
                return i;
            }
        }
        return -1;
    }
}

```

```

    public static int binarySearch(List<String> array,
String key) {
    int left = 0, right = array.size()-1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        int compare = array.get(mid).compareTo(key);
        if (compare == 0) {
            return mid;
        } else if (compare > 0) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return -1;
}
}

```

### Main.java:

```

import java.util.ArrayList;
import java.util.List;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/>
or
// click the <icon src="AllIcons.Actions.Execute"/> icon in
the gutter.
public class Main {
    public static void main(String[] args) {
        List<String> productId = new ArrayList<>();
        List<String> productName = new ArrayList<>();
        List<String> category = new ArrayList<>();

        productId.add("P001");
        productId.add("P002");
        productId.add("P003");
    }
}

```

```

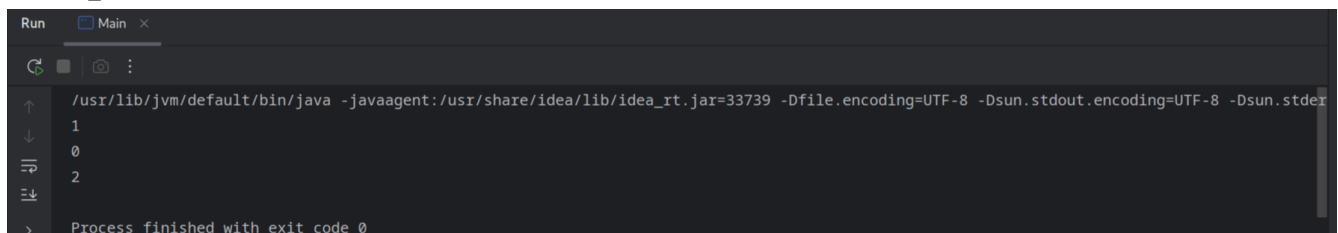
        productName.add("aaa");
        productName.add("bbb");
        productName.add("ccc");

        category.add("xxx");
        category.add("yyy");
        category.add("zzz");

        Product p = new Product(productId, productName,
category);
        System.out.println(p.searchProductId("P002"));
        System.out.println(p.searchProductName("aaa"));
        System.out.println(p.searchCategory("zzz"));
    }
}

```

## Output:



```

Run Main x
/usr/lib/jvm/default/bin/java -javaagent:/usr/share/idea/lib/idea_rt.jar=33739 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
1
0
2
Process finished with exit code 0

```

## Exercise 7: Financial Forecasting

### Predictor.java:

```

public class Predictor {

    private final double RATE = 0.05;
    private double principalAmount;

    Predictor(int principalAmount) {
        this.principalAmount = principalAmount;
    }

    public double getAmount(int years) {
        if (years == 0) {

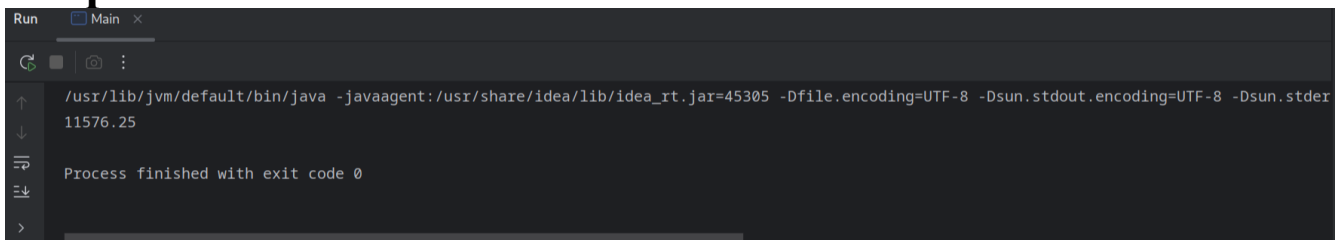
```

```
        return principalAmount;
    }
    return getAmount(years - 1) * (1 + RATE);
}
}
```

## Main.java:

```
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/>
or
// click the <icon src="AllIcons.Actions.Execute"/> icon in
the gutter.
public class Main {
    public static void main(String[] args) {
        Predictor p = new Predictor(10000);
        System.out.println(p.getAmount(3));
    }
}
```

## Output:



The screenshot shows the Run console of an IDE. At the top, there's a tab labeled 'Run' and 'Main'. Below the tab, there's a toolbar with icons for running, debugging, and other actions. The main area of the console displays the command used to run the program: `/usr/lib/jvm/default/bin/java -javaagent:/usr/share/idea/lib/idea_rt.jar=45305 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8`. The output of the program is `11576.25`. At the bottom, it says 'Process finished with exit code 0'.

```
Run Main x
/usr/lib/jvm/default/bin/java -javaagent:/usr/share/idea/lib/idea_rt.jar=45305 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
11576.25
Process finished with exit code 0
```