Internet and Web Programming (CSE3002) Digital Assignment 2

Name: Om Ashish Mishra

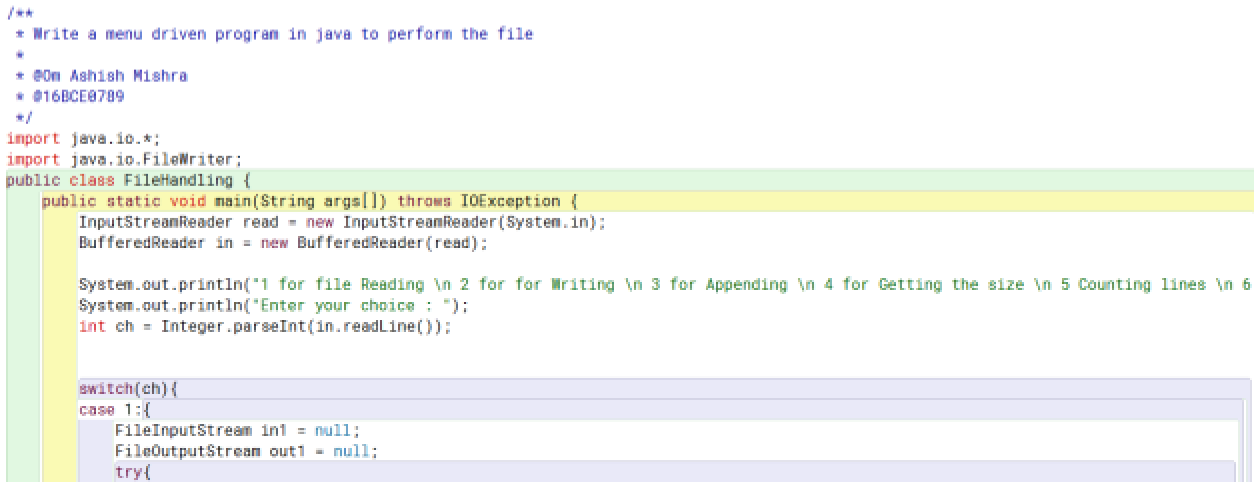
Registration Number: 16BCE0789

Slot: E2

1. Write a menu driven program in java to perform the following file operations.
    - Read the File
    - Write to a File
    - Appending to a File
    - Getting the Size of a File
    - Count lines of a particular file
    - Copying a File to Another File

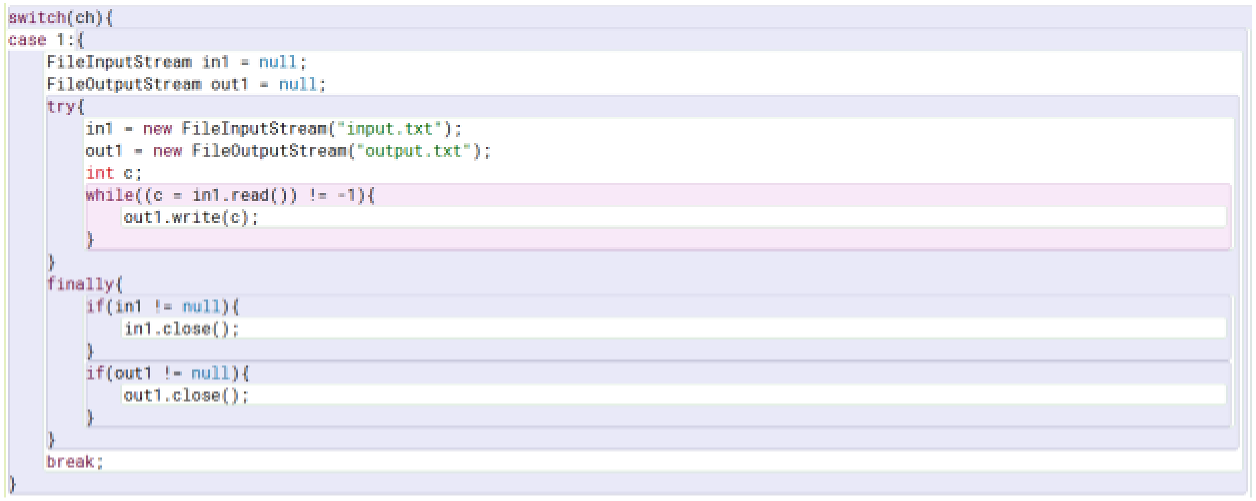
The Answer:

The Code:



```
/**
 * Write a menu driven program in java to perform the file
 *
 * @Om Ashish Mishra
 * @16BCE0789
 */
import java.io.*;
import java.io.FileWriter;
public class FileHandling {
    public static void main(String args[]) throws IOException {
        InputStreamReader read = new InputStreamReader(System.in);
        BufferedReader in = new BufferedReader(read);

        System.out.println("1 for file Reading \n 2 for for Writing \n 3 for Appending \n 4 for Getting the size \n 5 Counting lines \n 6
        System.out.println("Enter your choice : ");
        int ch = Integer.parseInt(in.readLine());


        switch(ch){
        case 1:{
            FileInputStream in1 = null;
            FileOutputStream out1 = null;
            try{
```

Reading

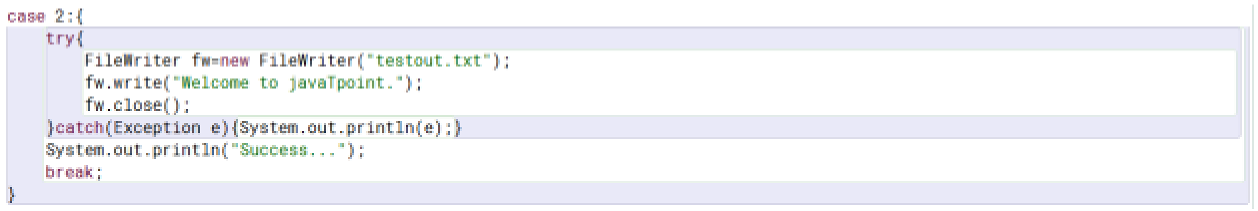


```
switch(ch){
case 1:{
    FileInputStream in1 = null;
    FileOutputStream out1 = null;
    try{
        in1 = new FileInputStream("input.txt");
        out1 = new FileOutputStream("output.txt");
        int c;
        while((c = in1.read()) != -1){
            out1.write(c);
        }
    }
    finally{
        if(in1 != null){
            in1.close();
        }
        if(out1 != null){
            out1.close();
        }
    }
    break;
}
```

The Output

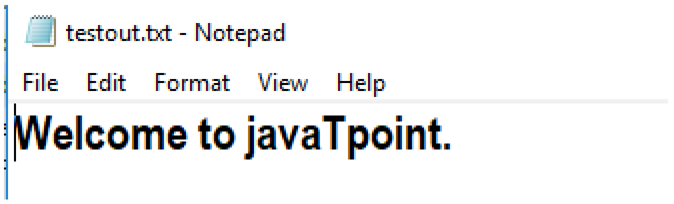
This reads the details of the file in output.txt

Writing



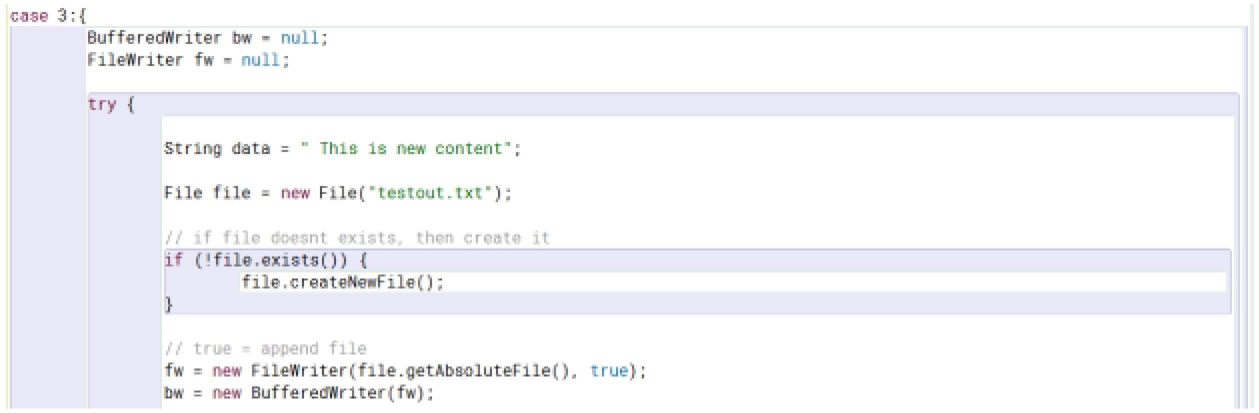
```
case 2:{
    try{
        FileWriter fw=new FileWriter("testout.txt");
        fw.write("Welcome to javaTpoint.");
        fw.close();
    }catch(Exception e){System.out.println(e);}
    System.out.println("Success...");
    break;
}
```

The output:

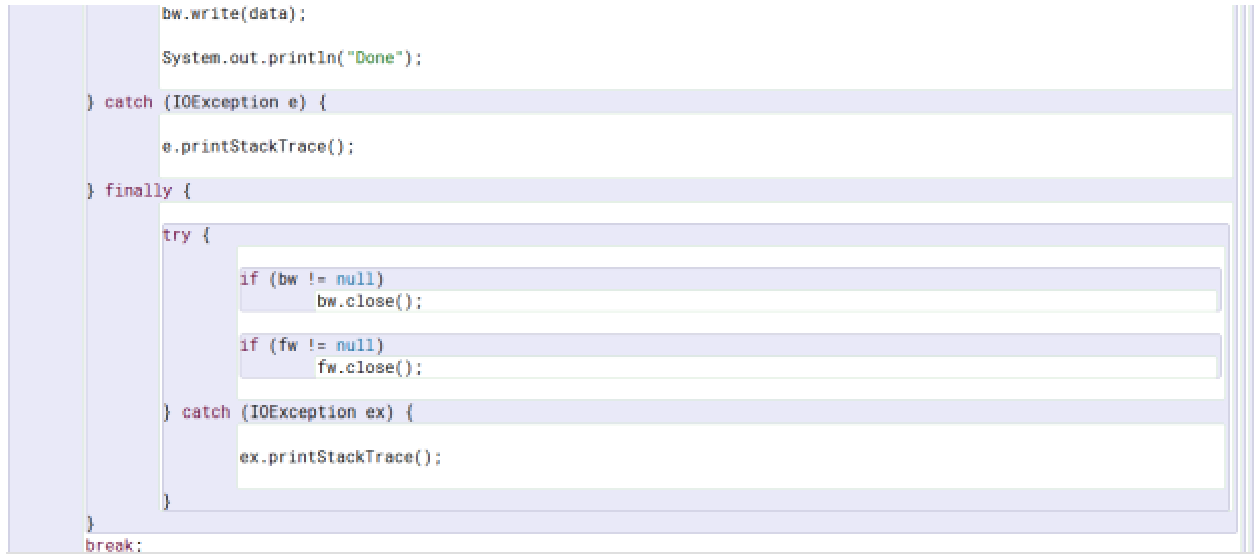


Welcome to javaTpoint.

Appending

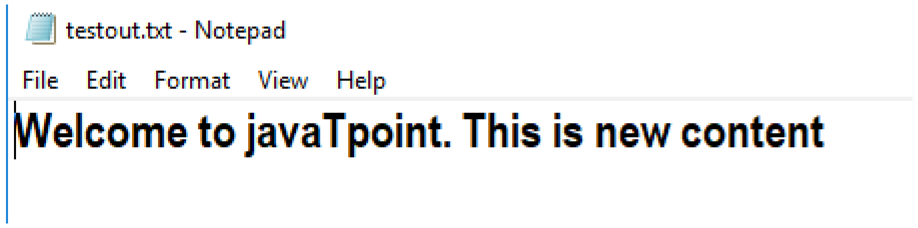


```
case 3:{
        BufferedWriter bw = null;
        FileWriter fw = null;

        try {

                String data = " This is new content";

                File file = new File("testout.txt");

                // if file doesnt exists, then create it
                if (!file.exists()) {
                        file.createNewFile();
                }

                // true = append file
                fw = new FileWriter(file.getAbsoluteFile(), true);
                bw = new BufferedWriter(fw);
```



```
                bw.write(data);

                System.out.println("Done");

        } catch (IOException e) {

                e.printStackTrace();

        } finally {

                try {

                        if (bw != null)
                                bw.close();

                        if (fw != null)
                                fw.close();

                } catch (IOException ex) {

                        ex.printStackTrace();

                }
        }
        break;
```

The Output

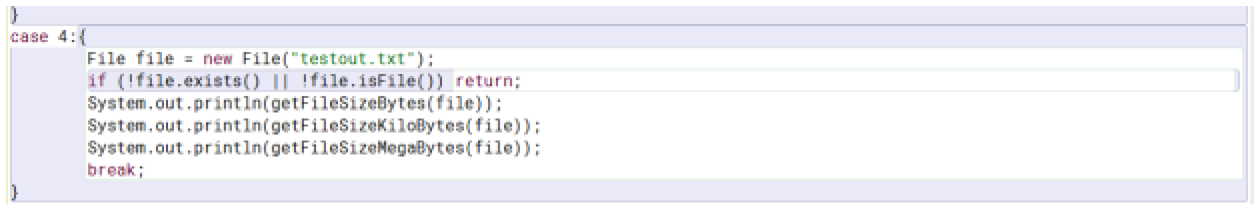


testout.txt - Notepad

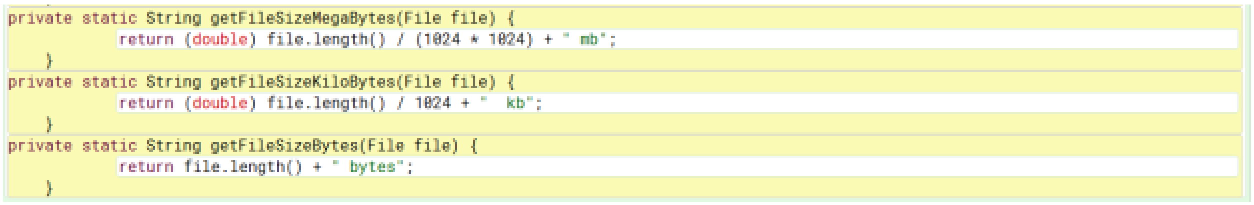
File Edit Format View Help

Welcome to javaTpoint. This is new content

Size of the file:

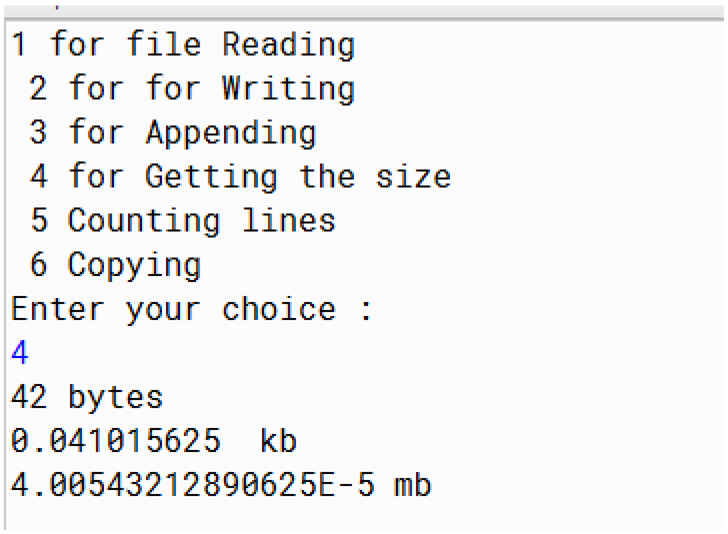


```
}
case 4:{
        File file = new File("testout.txt");
        if (!file.exists() || !file.isFile()) return;
        System.out.println(getFileSizeBytes(file));
        System.out.println(getFileSizeKiloBytes(file));
        System.out.println(getFileSizeMegaBytes(file));
        break;
}
```



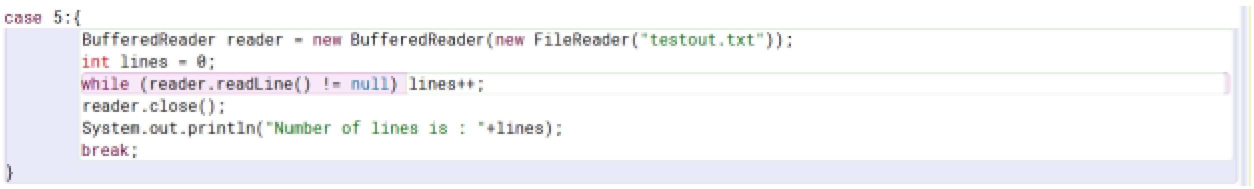
```java
private static String getFileSizeMegaBytes(File file) {
		return (double) file.length() / (1024 * 1024) + " mb";
	}
private static String getFileSizeKiloBytes(File file) {
		return (double) file.length() / 1024 + "  kb";
	}
private static String getFileSizeBytes(File file) {
		return file.length() + " bytes";
	}
```

The Output



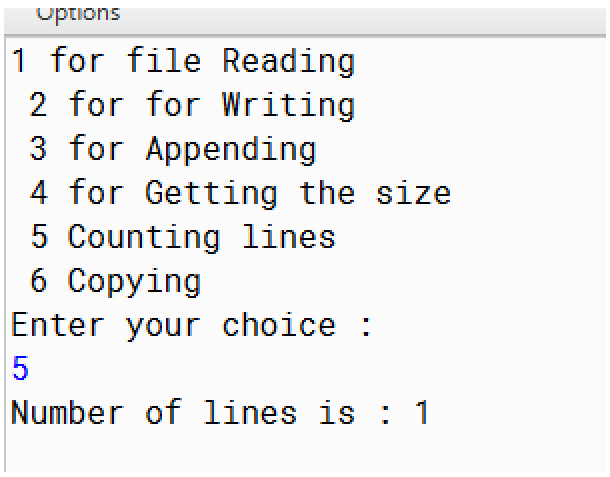
```
1 for file Reading
 2 for for Writing
 3 for Appending
 4 for Getting the size
 5 Counting lines
 6 Copying
Enter your choice :
4
42 bytes
0.041015625  kb
4.00543212890625E-5 mb
```

Counting lines in a particular file:



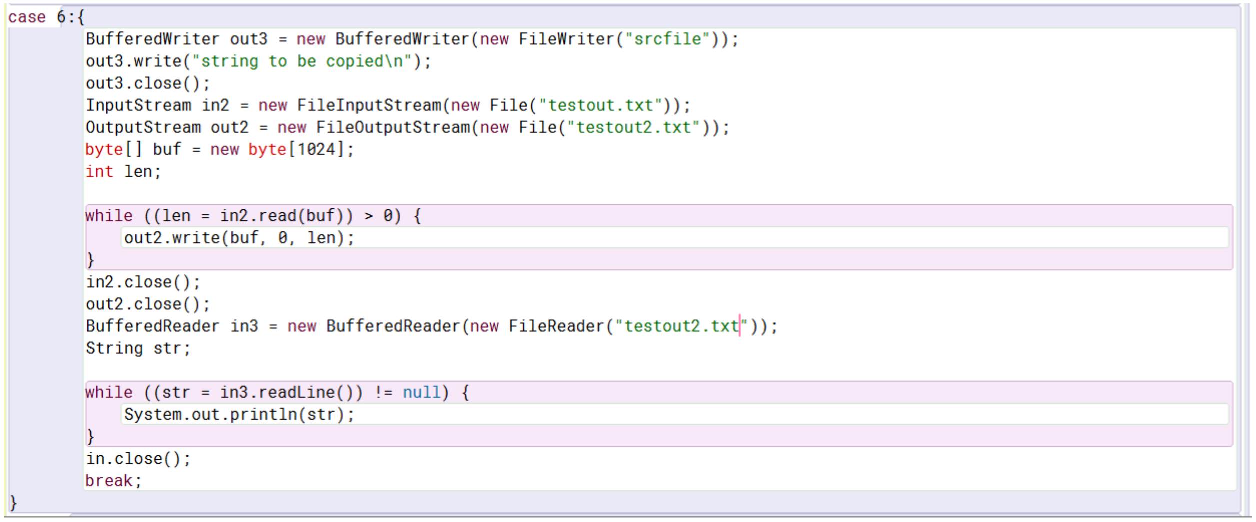
```java
case 5:{
	BufferedReader reader = new BufferedReader(new FileReader("testout.txt"));
	int lines = 0;
	while (reader.readLine() != null) lines++;
	reader.close();
	System.out.println("Number of lines is : "+lines);
	break;
}
```

The Output



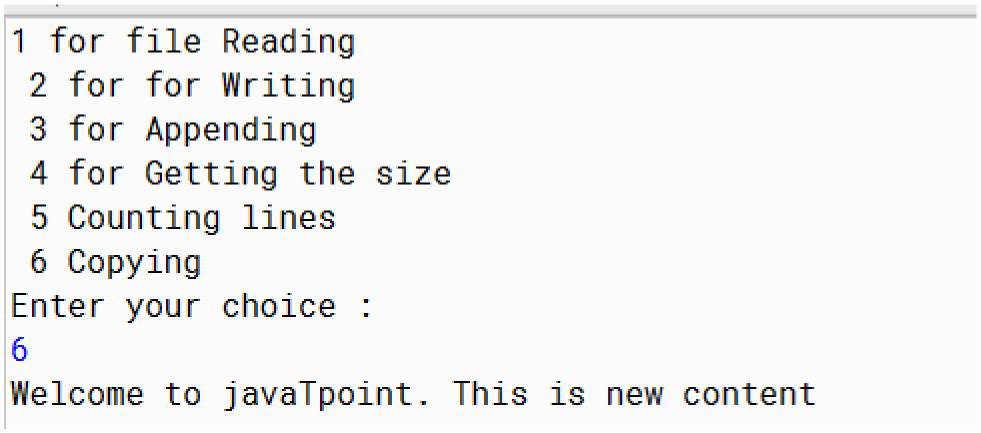
```
1 for file Reading
 2 for for Writing
 3 for Appending
 4 for Getting the size
 5 Counting lines
 6 Copying
Enter your choice :
5
Number of lines is : 1
```

Copying one file to another:



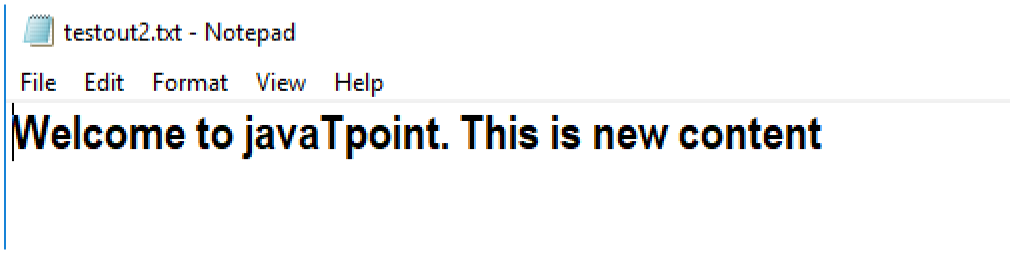
```
case 6:{
        BufferedWriter out3 = new BufferedWriter(new FileWriter("srcfile"));
        out3.write("string to be copied\n");
        out3.close();
        InputStream in2 = new FileInputStream(new File("testout.txt"));
        OutputStream out2 = new FileOutputStream(new File("testout2.txt"));
        byte[] buf = new byte[1024];
        int len;

        while ((len = in2.read(buf)) > 0) {
            out2.write(buf, 0, len);
        }
        in2.close();
        out2.close();
        BufferedReader in3 = new BufferedReader(new FileReader("testout2.txt"));
        String str;

        while ((str = in3.readLine()) != null) {
            System.out.println(str);
        }
        in.close();
        break;
}
```

The Output:



```
1 for file Reading
 2 for for Writing
 3 for Appending
 4 for Getting the size
 5 Counting lines
 6 Copying
Enter your choice :
6
Welcome to javaTpoint. This is new content
```

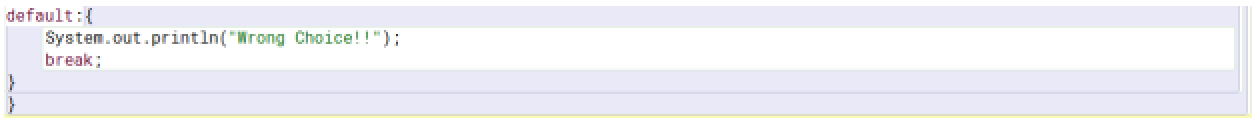
The text is copied to "testout2.txt" file:-



testout2.txt - Notepad

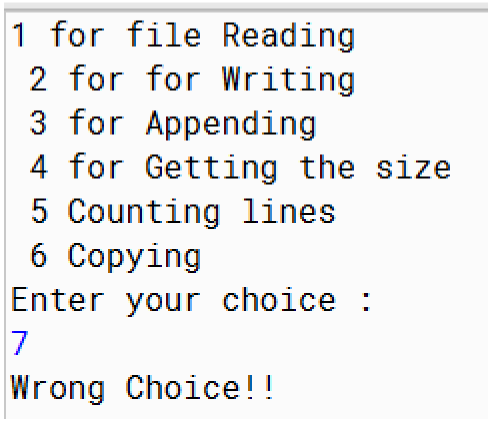
Welcome to javaTpoint. This is new content

For Wrong Choice:



```
default:{
    System.out.println("Wrong Choice!!");
    break;
}
}
```

The Output:



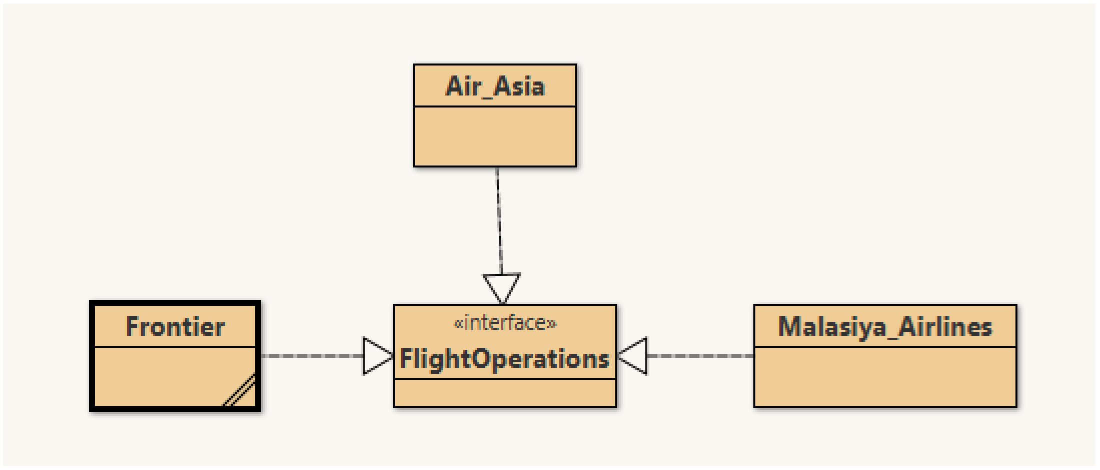
```
1 for file Reading
 2 for for Writing
 3 for Appending
 4 for Getting the size
 5 Counting lines
 6 Copying
Enter your choice :
7
Wrong Choice!!
```

2. Create three classes (Frontier, Air Asia, Malaysia Airlines) implementing the below interface and place an order for seat to respective vendor.

```
interface FlightOpeartions
{
void getAllAvailableFlights();
void booking();
}
```

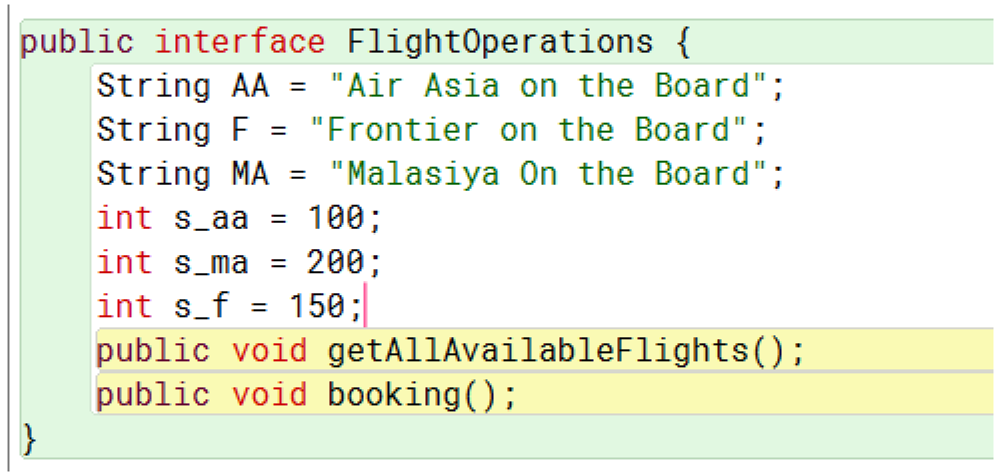
The Answer:

The layout:



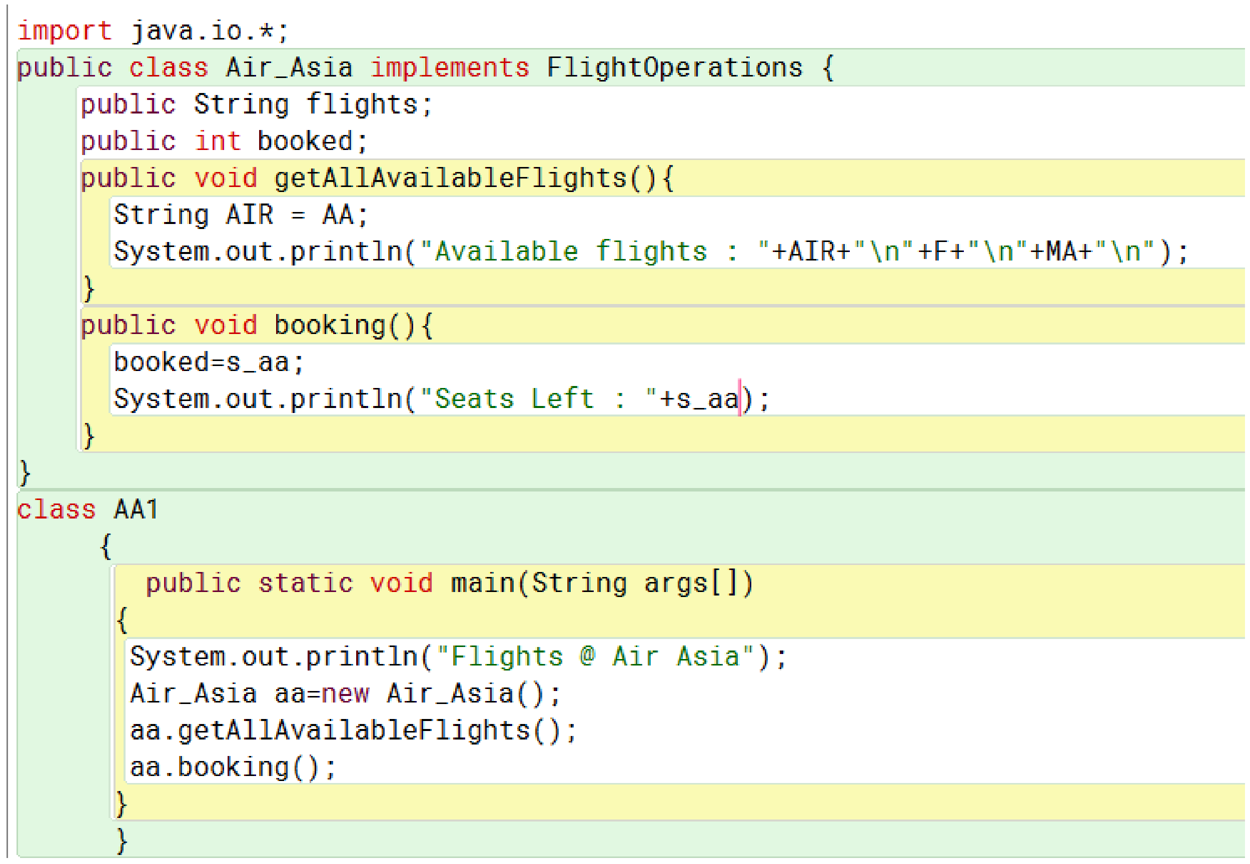
The Code for Each:

For Interface:



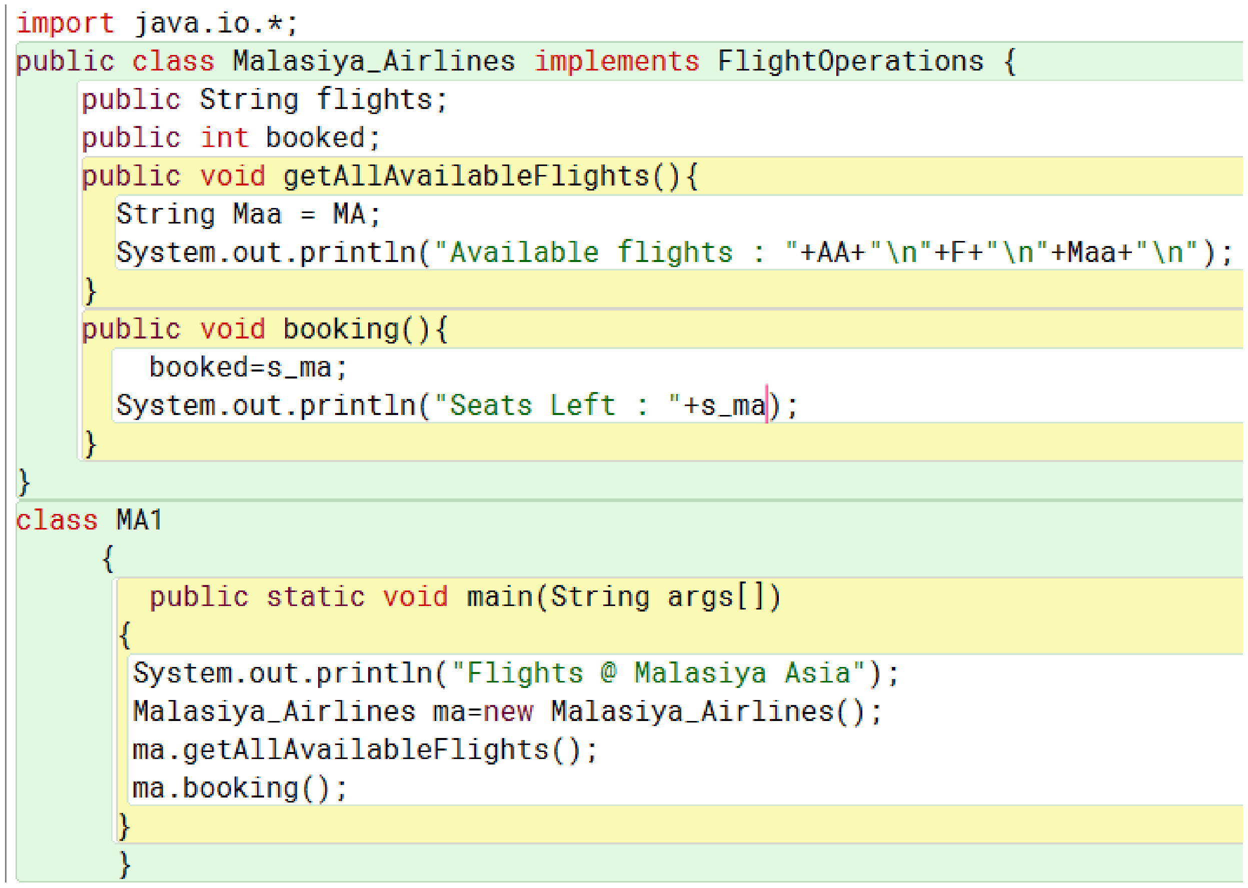
```java
public interface FlightOperations {
    String AA = "Air Asia on the Board";
    String F = "Frontier on the Board";
    String MA = "Malasiya On the Board";
    int s_aa = 100;
    int s_ma = 200;
    int s_f = 150;
    public void getAllAvailableFlights();
    public void booking();
}
```

For Air Asia:



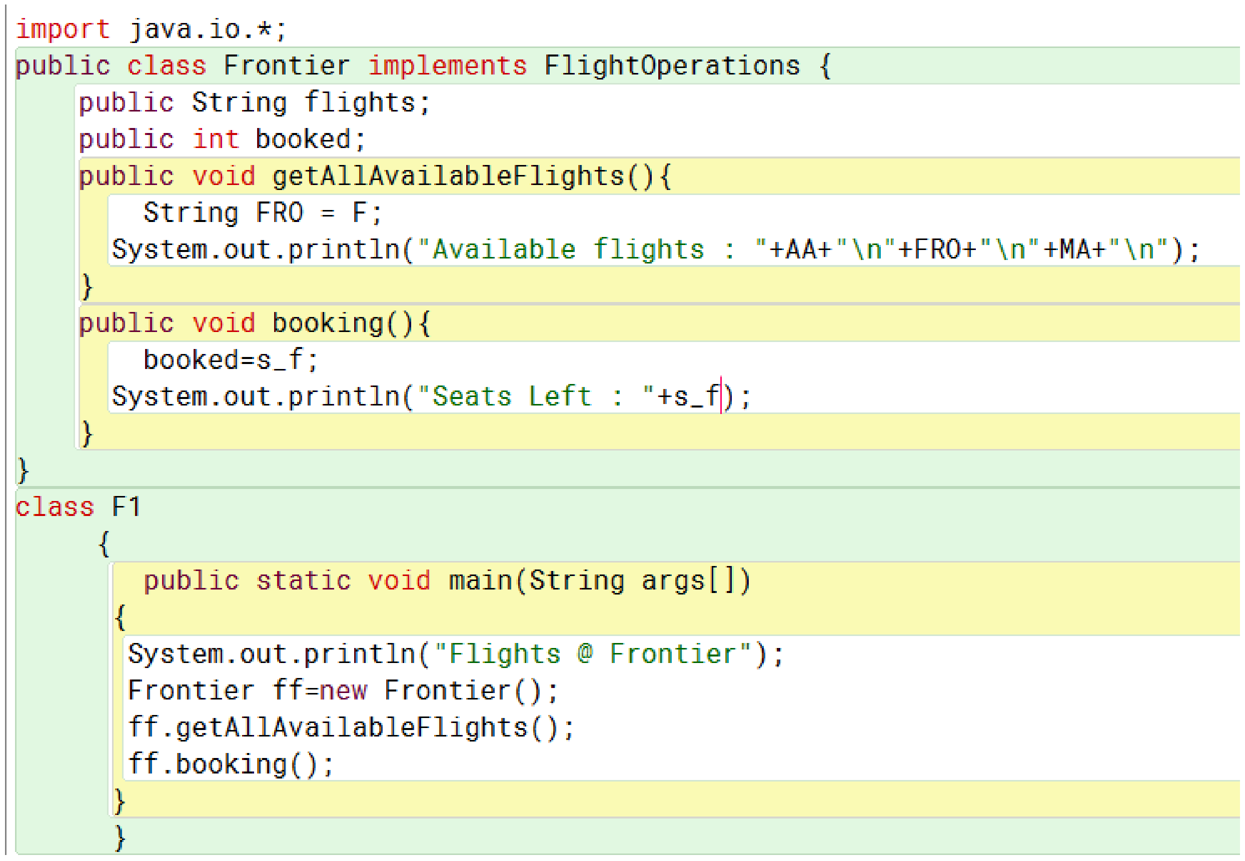
```java
import java.io.*;
public class Air_Asia implements FlightOperations {
    public String flights;
    public int booked;
    public void getAllAvailableFlights(){
        String AIR = AA;
        System.out.println("Available flights : "+AIR+"\n"+F+"\n"+MA+"\n");
    }
    public void booking(){
        booked=s_aa;
        System.out.println("Seats Left : "+s_aa);
    }
}
class AA1
    {
        public static void main(String args[])
        {
         System.out.println("Flights @ Air Asia");
         Air_Asia aa=new Air_Asia();
         aa.getAllAvailableFlights();
         aa.booking();
        }
    }
```

For Malasiya Airlines:



```
import java.io.*;
public class Malasiya_Airlines implements FlightOperations {
    public String flights;
    public int booked;
    public void getAllAvailableFlights(){
      String Maa = MA;
      System.out.println("Available flights : "+AA+"\n"+F+"\n"+Maa+"\n");
    }
    public void booking(){
        booked=s_ma;
      System.out.println("Seats Left : "+s_ma);
    }
}
class MA1
     {
        public static void main(String args[])
       {
        System.out.println("Flights @ Malasiya Asia");
        Malasiya_Airlines ma=new Malasiya_Airlines();
        ma.getAllAvailableFlights();
        ma.booking();
       }
       }
```

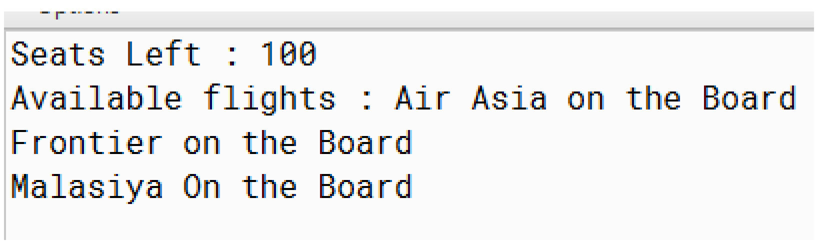
For Frontier:



```java
import java.io.*;
public class Frontier implements FlightOperations {
    public String flights;
    public int booked;
    public void getAllAvailableFlights(){
        String FRO = F;
      System.out.println("Available flights : "+AA+"\n"+FRO+"\n"+MA+"\n");
    }
    public void booking(){
        booked=s_f;
      System.out.println("Seats Left : "+s_f);
    }
}
class F1
     {
        public static void main(String args[])
      {
       System.out.println("Flights @ Frontier");
       Frontier ff=new Frontier();
       ff.getAllAvailableFlights();
       ff.booking();
      }
      }
```

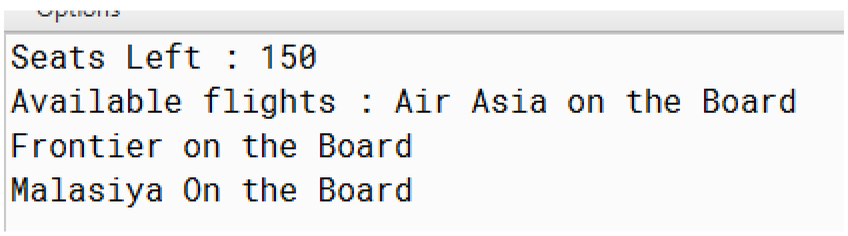
The Output:

For Air Asia:



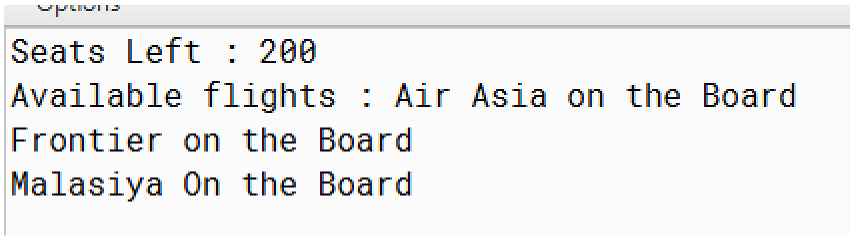
```
Seats Left : 100
Available flights : Air Asia on the Board
Frontier on the Board
Malasiya On the Board
```

For Frontier:



```
Seats Left : 150
Available flights : Air Asia on the Board
Frontier on the Board
Malasiya On the Board
```

Malasiya Airlines:



```
Seats Left : 200
Available flights : Air Asia on the Board
Frontier on the Board
Malasiya On the Board
```

Since in the question it was mentioned to represent all the flight, therefore all the flight have been shown. The seat available varies in all the flights also have been shown.

3.One page Write up on Java Data mining API.

Java Data mining API

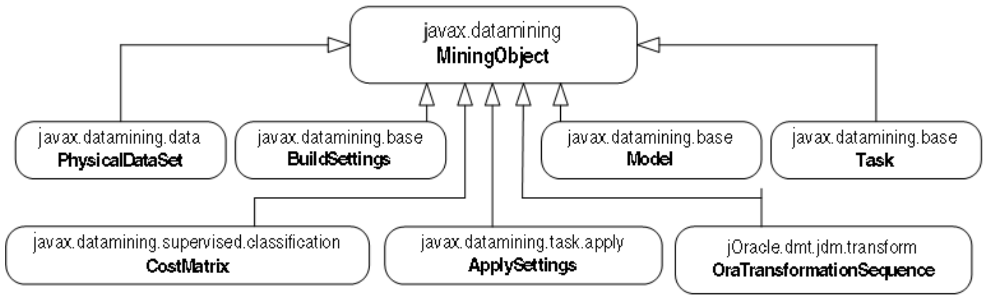
Java Data Mining (JDM) is a standard **Java API** for developing data mining applications and tools. JDM defines an object model and Java API for data mining objects and processes. JDM enables applications to integrate data mining technology for developing predictive analytics applications and tools. The JDM 1.0 standard was developed under the Java Community Process. In 2006, the JDM 2.0 specification was being developed under JSR 247, but has been withdrawn in 2011 without standardization.

Various data mining functions and techniques like statistical classification and association, regression analysis, data clustering, and attribute importance are covered by the 1.0 release of this standard. It never received wide acceptance, and there is no known implementation.

The JDM standard uses the factory method pattern as the core design pattern for the API. User can instantiate a JDM object using its factory. This enables JDM vendors like Oracle to implement a vendor neutral API. **OJDM**(Oracle Java Data Mining API) follows the same factory method pattern for its extensions. **javax.datamining** is the base package for the JDM standard defined classes and **oracle.dmt.jdm** is the base package for the Oracle extensions to the JDM standard.

The JDM standard organizes its packages by the mining functions and mining algorithms. For example, **javax.datamining.supervised** package has all the supervised functions related classes and sub packages.

java.datamining.supervised.classification and **java.datamining.supervised.regression**. Each function sub-package has the classes related to that function. Similarly, **javax.datamining.algorithm** is the base package for all algorithms and each algorithm has its sub-package under this package, for example, **javax.datamining.algorithm.naivebayes** is the sub-package for Naïve Bayes algorithm related classes. OJDM follows a similar package structure for the extensions, for example, feature extraction is a non-JDM standard function supported by the OJDM, here **oracle.dmt.jdm.featureextraction** is the package for this function and **oracle.dmt.jdm.algorithm.nmf** package for the Non-Negative Matrix Factorization algorithm used for feature extraction. The JDM standard has some core packages that define common classes and packages for tasks, model details, rules and statistics.


javax.datamining
MiningObject
javax.datamining.data
PhysicalDataSet
javax.datamining.base
BuildSettings
javax.datamining.base
Model
javax.datamining.base
Task
javax.datamining.supervised.classification
CostMatrix
javax.datamining.task.apply
ApplySettings
jOracle.dmt.jdm.transform
OraTransformationSequence