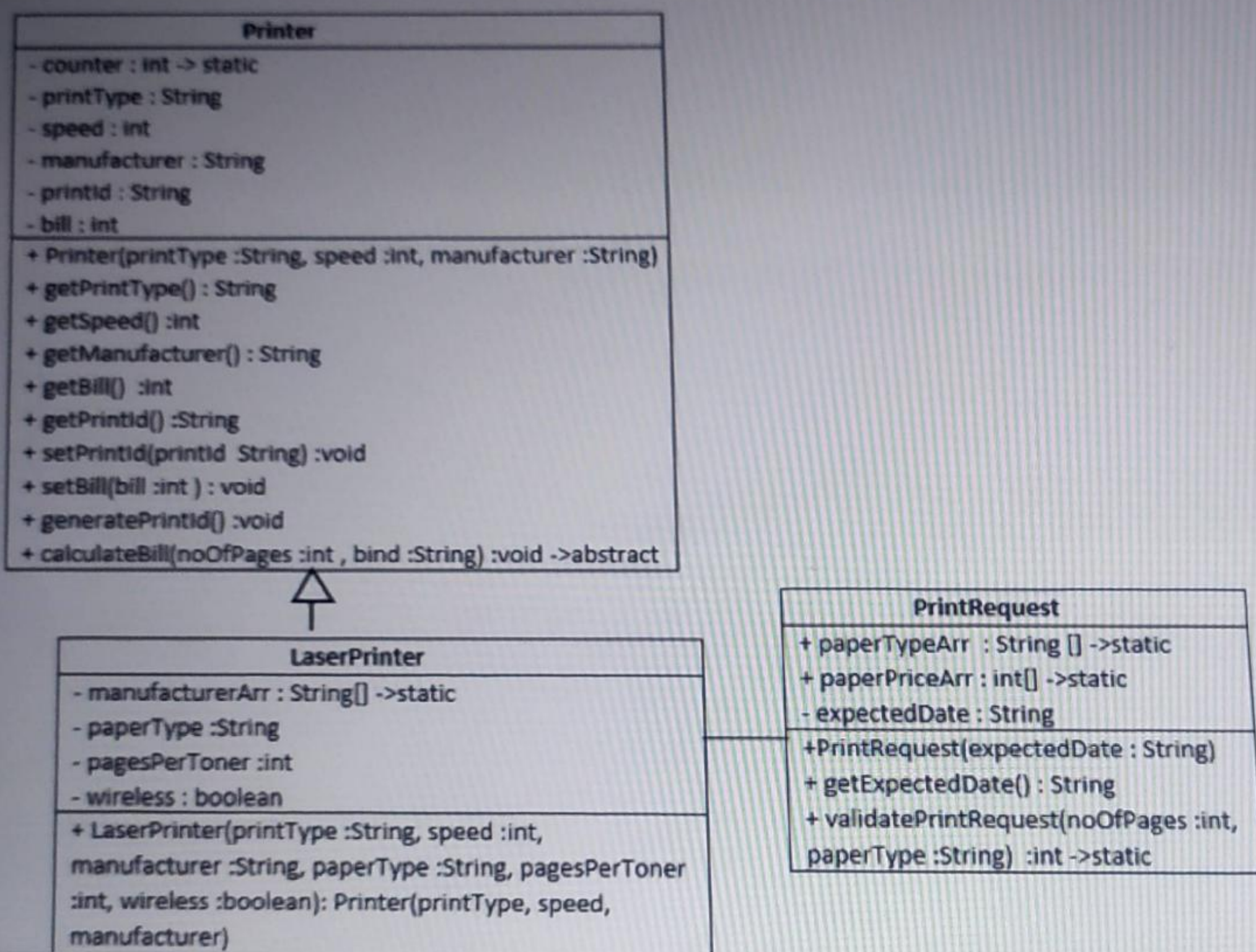


"InkRun", a Printing company wants to automate its bulk orders of printing requests. Implement the below class diagram to achieve the same.

Class diagram:



Note:

- Do not include any extra instance/static variables and instance/static methods in the given classes
- Case in-sensitive comparison is required to be done unless until specified explicitly
- Do not change any value or case of the given variables
- Read notes and examples for better understanding of the logic
- In the derived classes, the order of passing arguments to the constructor would be- base class variables followed by derived class variables

Implementation Details:

Class Name	Implementation Details
PrintRequest	Partially Implemented
Printer	Partially Implemented
LaserPrinter	Partially Implemented

PrintRequest class:

paperTypeArr:

This is a static array (String[]) which contains *paperType* (String) as its elements

paperTypeArr:

- This is a static array (String[]) which contains *paperType* (String) as its elements
- The initial values of the **paperTypeArr** are given below -

paperTypeArr	{"thicksheet", "glossysheet", "A4sheet", "A3sheet"}
---------------------	---

Note:

- This array is supplied. Hence no need to code
- Do not change the **CASE** of elements in the array

paperPriceArr:

- This is a static array (int[]) which contains *price* (int) as its elements
- This array has one-to-one correspondence with **paperTypeArr**
- The initial values of the **paperPriceArr** are given below -

paperPriceArr	{40, 10, 3, 5}
----------------------	----------------

Note:

Note:

- This array is supplied. Hence no need to code

validatePrintRequest(*noOfPages*, *paperType*):

This is a static method which accepts *noOfPages* (int) and *paperType* (String) as its parameters and returns *price* (int) for the given *paperType*

If the *noOfPages* is between 90(exclusive) and 500(inclusive), then

- If the *paperType* is present in the **paperTypeArr** then, identify the *price* of the *paperType* from corresponding value in **paperPriceArr**
- If *paperType* is not found in **paperTypeArr** then, *price* would be -1

Otherwise, return *price* as -1

Note: Perform case in-sensitive comparison

Example: If the *noOfPages* is 150 and *paperType* is "THICKSHEET", then the above method must return *price* as 40

Printer class:

generatePrintId():

This method auto-generates and sets the **printId** (String)

The **printId** must be prefixed with first letter of **printType** followed by auto-generated value starting from 101

The auto-generated value should be incremented by 1 for the next **printId**

Exam Data Submit Help

This method auto-generates and sets the **printId** (String)

The **printId** must be prefixed with first letter of **printType** followed by auto-generated value starting from 101

The auto-generated value should be incremented by 1 for the next **printId**

Use the static variable **counter** appropriately to implement auto-generation logic

Note: Perform case sensitive comparison

Example: If the **printType** is "Color", then the first **printId** would be "C101", second **printId** generated would be "M102" if the **printType** is "Monochromatic" and so on

LaserPrinter class:

manufacturerArr:

- This is a static array (String[]) which contains details of *manufacturer*(String) as its elements
- The initial values of the **manufacturerArr** are given below -

manufacturerArr	{"Hp", "Dell", "Epson"}
------------------------	-------------------------

Note:

- This array is supplied. Hence no need to code
- Do not change the **CASE** of elements in the array

checkPerformance():

- This method checks the performance of the printer and returns true/false based on the below logic
- Check if the **printType** is "Monochromatic", if yes then,
 - Check if **pagesPerToner** is greater than or equal to 1700 and **speed** is greater than or equal to 150, if yes, then return true
 - Otherwise, return false
- Otherwise, check if **printType** is "Color", if yes then,
 - Check if **pagesPerToner** is greater than or equal to 1500 and **speed** is greater than or equal to 100, if yes, then return true
 - Otherwise, return false
- Otherwise, return false

Note: Perform case sensitive comparison

Example: If the **printType** is "Color" and **pagesPerToner** is 150 and **speed** is 2000, then the above method returns **true**

calculateBill(*noOfPages*, *bind*):

- This method takes *noOfPages* (int) and *bind* (String) as its parameters
- This method calculates and sets the **bill** and generates the **printId** based on the below logic

Example: If the **printType** is "Color" and **pagesPerToner** is 150 and **speed** is 2000, then the above method returns **true**

calculateBill(*noOfPages*, *bind*):

- This method takes *noOfPages* (int) and *bind* (String) as its parameters
- This method calculates and sets the **bill** and generates the **printId** based on the below logic
- Identify *price* by invoking **validatePrintRequest()** method of **PrintRequest** class by passing *noOfPages* and **paperType** as parameter
- If the *price* is not -1 and **validateManufacturer()** method of **LaserPrinter** class returns true and attribute **wireless** is true, then -
 - If **checkPerformance()** method returns true, then -
 - Calculate *billAmount* as product of *price* and *noOfPages*
 - If *bind* is "spiral", add additional currency of 35 to the *billAmount*. Otherwise, if *bind* is "lamination", add additional currency of 20 to the *billAmount*. And if the *bind* is null, add additional currency of 10 to the *billAmount*
 - If the **printType** is "Color", then double the *billAmount*
 - Generate **printId** by invoking **generatePrintId()** method of **Printer** class
 - Set the **bill** with obtained *billAmount*
 - Otherwise, set the **printId** as "NA" (uppercase) and **bill** as -1
- Otherwise, set the **printId** as "NA" (uppercase) and **bill** as -1

Note:

- The parameter *bind* can take valid values such as null, "spiral" or "lamination". Only valid values would be passed
- Perform case sensitive comparison

Example: If *printType* is "Color", *speed* is 150, *manufacturer* is "hP", *paperType* is "THICKSHEET", *pagesPerToner* is 2000, *wireless* is true, *noOfPages* is 150 and *bind* is null, then *printid* would be 12000 currency.

Question 2: Data Structures

[5 Marks]

Problem Statement:

Description: Write a Java program that accepts non-empty *inStrStack* (String Stack) as input parameter and returns an *outStrStack* (String Stack) based on the logic given below:

- Consider the elements of the *inStrStack*, where each *stackElement* is in the format given below:

"employeeName:departmentNo" (where *departmentNo* can be "D1", "D2" or "D3")
- The *outStrStack* would be obtained by rearranging all the *stackElement* of *inStrStack* in ascending order of *departmentNo* such that employees belonging to department "D1" would be at the top of the *outStrStack* followed by employees belonging to department "D2" followed by employees belonging to department "D3" at the bottom of the *outStrStack*
- The employees belonging to the same *departmentNo* would be added as elements to *outStrStack* (Top → Bottom) in the same order as they appear in the *inStrStack* (Top → Bottom)
- If no employees belong to department "D1", obtain *outStrStack* by arranging all the *stackElement* of *inStrStack* in ascending order of *departmentNo* such that employees belonging to department "D2" would be at the top of the *outStrStack* followed by employees belonging to department "D3" at the bottom of the *outStrStack* (also applicable to "D2" and "D3")

Description: Write a Java program that accepts non-empty **inStrStack** (String Stack) as input parameter and returns an **outStrStack** (String Stack) based on the logic given below:

- Consider the elements of the **inStrStack**, where each *stackElement* is in the format given below:
"employeeName:departmentNo" (where *departmentNo* can be "D1", "D2" or "D3")
- The **outStrStack** would be obtained by rearranging all the *stackElement* of **inStrStack** in ascending order of *departmentNo* such that employees belonging to department "D1" followed by employees belonging to department "D2" followed by employees belonging to department "D3" at the bottom of the **outStrStack**
- The employees belonging to the same *departmentNo* would be added as elements to **outStrStack** (Top → Bottom) in the same order as they appear in the **inStrStack** (Top → Bottom)
- If no employees belong to department "D1", obtain **outStrStack** by arranging all the *stackElement* of **inStrStack** in ascending order of *departmentNo* such that employees belonging to department "D2" followed by employees belonging to department "D3" at the bottom of the **outStrStack** (also applicable to "D2" and "D3")
- If all employees belong to the same *departmentNo*, obtain the **outStrStack** by adding all the *stackElement* in the same order (Top → Bottom) as they appear in the **inStrStack**
- Return **outStrStack**

Assumptions:

- **inStrStack** would not be empty
- The employees can belong to only three departments i.e. D1, D2 and D3

outStrStack (Top → Bottom): {"John:D1", "Karen:D2", "Ken:D2", "Collin:D2", "Lily:D3", "Mary:D3"}

Explanation:

- From the above example, in the **InStrStack**, the employee belonging to *departmentNo* "D1" is "John".

Hence, add "John:D1" to **outStrStack**.

Now the **outStrStack** (Top → Bottom) would be {"John:D1"}

- The employees belonging to *departmentNo* "D2" are "Karen", "Ken", and "Collin". Hence add "Karen:D2", "Ken:D2" and "Collin:D2" to **outStrStack** (Top → Bottom) in the same order as they appear in the **InStrStack** (Top → Bottom).

Now the **outStrStack** (Top → Bottom) would be {"John:D1", "Karen:D2", "Ken:D2", "Collin:D2"}

- The employees belonging to *departmentNo* "D3" are "Lily" and "Mary". Hence add "Lily:D3" and "Mary:D3" to **outStrStack** (Top → Bottom) in the same order as they appear in the **InStrStack** (Top → Bottom).

The final **outStrStack** (Top → Bottom) would be {"John:D1", "Karen:D2", "Ken:D2", "Collin:D2", "Lily:D3", "Mary:D3"}

Sample Input and Output:

InStrStack (Top → Bottom)	outStrStack (Top → Bottom)
{"Simon:D3", "Kylie:D1", "Kim:D3", "Carrie:D1", "Mary:D1", "Will:D2"}	{"Kylie:D1", "Carrie:D1", "Mary:D1", "Will:D2", "Simon:D3", "Kim:D3"}
{"Matt:D3", "Sara:D2", "Tom:D3", "George:D3", "Wendy:D2"}	{"Sara:D2", "Wendy:D2", "Matt:D3", "Tom:D3", "George:D3"}
{"Liam:D2", "Sharon:D2", "Amelia:D2", "Jack:D2"}	{"Liam:D2", "Sharon:D2", "Amelia:D2", "Jack:D2"}


```
public class Solution{  
    public Stack sortDepartment(Stack inStrStack){  
        Stack outStrStack = new Stack(inStrStack.getMaxSize());  
        while(!inStrStack.isEmpty()){  
            String curr = inStrStack.pop();  
            int dept = Integer.parseInt(""+curr.charAt(curr.length()-1));  
            helper(outStrStack, dept, curr);  
        }  
  
        return outStrStack;  
    }  
    public void helper(Stack outStrStack, int dept, String str){  
        if(outStrStack.isEmpty()){  
            outStrStack.push(str);  
            return;  
        }  
        String temp = outStrStack.peek();  
        int d = Integer.parseInt(""+temp.charAt(temp.length()-1));  
        if(d <= dept){  
            outStrStack.pop();  
            helper(outStrStack, dept, str);  
            outStrStack.push(temp);  
        }else outStrStack.push(str);  
        return;  
    }  
}
```


//To trainees

```
public static int validatePrintRequest(int noOfPages, String paperType) {  
    int price=0;  
    if(noOfPages>90 && noOfPages<=500){  
        for(int i=0; i<PrintRequest.paperTypeArr.length; i++){  
            if(PrintRequest.paperTypeArr[i].equalsIgnoreCase(paperType))  
                price= PrintRequest.paperPriceArr[i];  
        }  
    }  
    else price = -1;  
    //Implement your logic here  
    return price;  
}
```



```
//To trainees
```

```
public void generatePrintId() {  
    char fc = this.printType.charAt(0);  
    String id = Character.toString(fc);  
    String oid = id + ++Printer.counter;  
    this.setPrintId(oid);
```

```
    //Implement your logic here
```

```
}
```

```
public abstract void calculateBill(int noOfPages, S
```



```
44
45 //To trainees
46 public void generatePrintId() {
47
48     //Implement your logic here
49     char firstChar = this.printType.charAt(0);
50     String id = Character.toString(firstChar);
51     String id1 = id + ++Printer.counter;
52     this.setPrintId(id1);
53 }
54
55 public abstract void calculateBill(int noOfPages, String bind);
56 }
57
```


Solution.java

LaserPrinter.java ✕

Printer.java

PrintRequest.java

```
8 //To trainees
9 @Override
0 public void calculateBill(int noOfPages, String bind){
1     int billAmount = -1;
2     int v= PrintRequest.validatePrintRequest(noOfPages, paperType);
3     boolean valmanf = validateManufacturer();
4     boolean ch = checkPerformance();
5     if(v!= -1 && valmanf==(true) && wireless==true){
6
7         if(ch==true){
8             billAmount= v*noOfPages;
9             if(bind=="spiral"){
10                 billAmount += 35;
11             }
12             else if(bind=="lamination"){
13                 billAmount += 20;
14             }
15             if(getPrintType()=="Color")
16                 billAmount = 2*billAmount;
17             super.generatePrintId();
18             this.setBill(billAmount);
19         }
20         else {setPrintId("NA");
21             setBill(-1);}
22     }
23     else {setPrintId("NA");
24         setBill(-1);}
25     //Implement your logic here
```



```
27 //To trainees
28 public Boolean checkPerformance() {
29     //Implement your logic here
30     if(super.getPrintType().equalsIgnoreCase("Monochromatic"))
31     {
32         if(this.pagesPerToner>=1700 && super.getSpeed()>=150)
33         {
34             return true;
35         }
36         else
37         {
38             return false;
39         }
40     }
41 }
42 else if(super.getPrintType().equalsIgnoreCase("Color"))
43 {
44     if(this.pagesPerToner>=1500 && super.getSpeed()>=100)
45     {
46         return true;
47     }
48     else
49     {
50         return false;
51     }
}
```



```

59 //To trainees
60 //To PrintRequest.java
61 //Tester.java
62 @Override
63 public void calculateBill(int noOfPages, String bind){
64     //Implement your logic here
65     int price = PrintRequest.validatePrintRequest(noOfPages, paperType);
66     boolean manu = this.validateManufacturer();
67     int billAmount = 0;
68
69     if(price!=-1 && manu==true && this.wireless ){
70         if(this.checkPerformance()==true){
71             billAmount = price*noOfPages;
72             if(bind.equals("spiral")){
73                 billAmount+=35;
74             }
75             else if(bind.equals("lamination")){
76                 billAmount+=20;
77             }
78             else if(bind==null){
79                 billAmount+=0;
80             }
81         }
82
83         if(this.getPrintType().equals("Color")){
84             billAmount = billAmount*2;
85         }
86
87         this.generatePrintId();
88
89     }
90 }




```

sole Servers ☐ Properties
 ted> Tester [Java Application] C:\Program Files\AdoptOpenJDK\jdk-8.0.202.08\bin\javaw.exe (Feb 9, 2021, 10:51:11 AM)
 unt : -1
 : NA


```

80      billAmount+=0;
81  }
82
83      if(this.getPrintType().equals("Color")){
84          billAmount = billAmount*2;
85      }
86
87      this.generatePrintId();
88      this.setBill(billAmount);
89
90  }
91  else{
92      this.setPrintId("NA");
93      this.setBill(-1);
94  }
95
96  }
97
98  else{
99      this.setPrintId("NA");
100     this.setBill(-1);
101 }
102
103 }
104 }
105

```

 Console
  Servers
  Properties

<terminated> Tester [Java Application] C:\Program Files\AdoptOpenJDK\jdk-8.0.202.08\bin\javaw.exe (Feb 9, 2021)

Project Explorer

- Assessment
- IRE System Library [JavaSE-1.8]
- src
 - dsausingjava
 - progusingjava
 - LaserPrinter.java
 - LaserPrinter
 - Printer.java
 - PrintRequest.java
 - Tester.java

45 }
46
47 //To trainees
48 @Override
49 public void calculateBill(int noOfPages, String bind){
50
51 //Implement your logic here
52 int billAmount = -1;
53 int price= PrintRequest.validatePrintRequest(noOfPages, this.paperType);
54 if(price != -1 && validateManufacturer()==true && this.wireless == true){
55 if(this.checkPerformance()==true){
56 billAmount = price * noOfPages;
57 if(bind!=null){
58 if(bind.equals("spiral"))
59 billAmount = billAmount+35;
60 else if(bind.equals("lamination"))
61 billAmount = billAmount +20;
62 }
63 if(this.getPrintType().equals("Color"))
64 billAmount = 2*billAmount;
65 super.generatePrintId();
66 this.setBill(billAmount);
67 }
68 }
69 else{
70 this.setPrintId("NA");
71 this.setBill(-1);
72 }
73 }
74 else{
75 this.setPrintId("NA");
76 this.setBill(-1);
77 }
78 }

Test Results

- 10/10 Structural Test Cases - Non Assessed
 - Looks for structural errors in the code.
 - All test cases must pass. These are not Assessed
- 13/13 Logical Test Cases - Non Assessed
 - Looks for logical errors in the code
 - Logical test cases will be shown only if all Structural test cases pass
 - Completing these will help debug your code
- 29/29 Logical Test Cases - Assessed
 - Test details are not shown.
- Good Code Quality Check
 - Looks for quality of the code
 - Review your code and improve the quality

Console Servers Properties

<terminated> Tester [Java Application] C:\Program Files\AdoptOpen\DK\jdk-8.0.202.08\bin\javaw.exe (Feb 9, 2021, 1:10:10 PM)
Displaying Stack elements:

Writable

```
}  
}
```

```
//To trainees
```

```
public Boolean checkPerformance() {  
    if(super.getPrintType().equalsIgnoreCase("Monochromatic")){  
        if(this.pagesPerToner>=1700 && super.getSpeed()>=150){  
            return true;  
        }  
        else return false;  
    }  
    else if(super.getPrintType().equalsIgnoreCase("Color")){  
        if(this.pagesPerToner>=1500 && super.getSpeed()>=100){  
            return true;  
        }  
        else return false;  
    }  
    //Implement your logic here
```