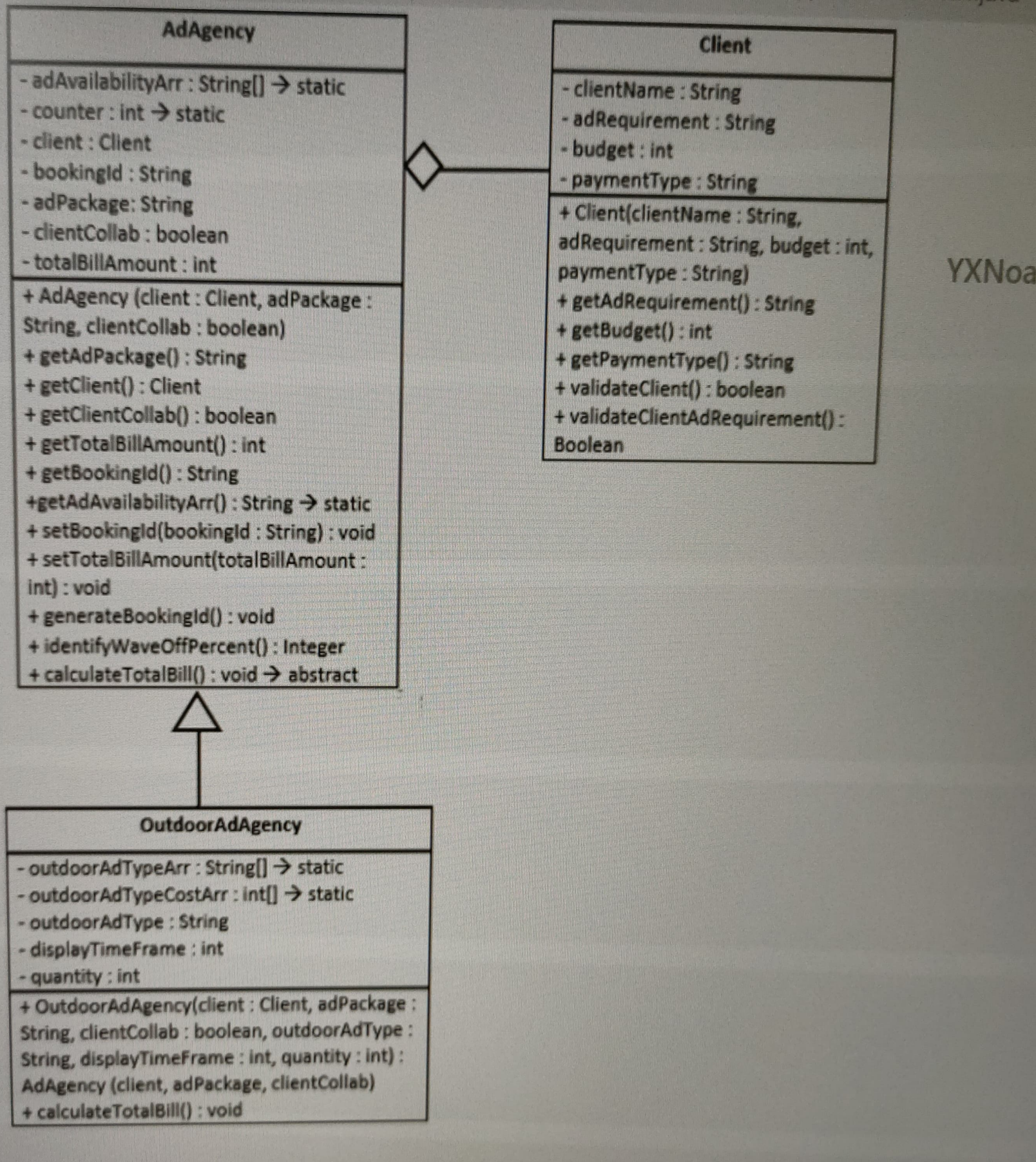

AdAgency
- adAvailabilityArr : String[] → static
- counter : int → static
- client : Client
- bookingId : String
- adPackage: String
- clientCollab : boolean
- totalBillAmount : int
+ AdAgency (client : Client, adPackage : String, clientCollab : boolean)
+ getAdPackage() : String
+ getClient() : Client
+ getClientCollab() : boolean
+ getTotalBillAmount() : int
+ getBookingId() : String
+getAdAvailabilityArr() : String → static
+ setBookingId(bookingId : String) : void
+ setTotalBillAmount(totalBillAmount : int) : void
+ generateBookingId() : void
+ identifyWaveOffPercent() : Integer
+ calculateTotalBill() : void → abstract
Client
- clientName : String
- adRequirement : String
- budget : int
- paymentType : String
+ Client(clientName : String, adRequirement : String, budget : int, paymentType : String)
+ getAdRequirement() : String
+ getBudget() : int
+ getPaymentType() : String
+ validateClient() : boolean
+ validateClientAdRequirement() : Boolean
OutdoorAdAgency
- outdoorAdTypeArr : String[] → static
- outdoorAdTypeCostArr : int[] → static
- outdoorAdType : String
- displayTimeFrame : int
- quantity : int
+ OutdoorAdAgency(client : Client, adPackage : String, clientCollab : boolean, outdoorAdType : String, displayTimeFrame : int, quantity : int) : AdAgency (client, adPackage, clientCollab)
+ calculateTotalBill() : void

- Do not include any extra instance/static variables and instance/static methods in the given classes
- **Case-insensitive** comparison is to be done if not explicitly mentioned
- 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

YXNoaWxhLm1hdGhld3M=

Implementation Details:

Class Name	Implementation Details
AdAgency	Partially implemented
OutdoorAdAgency	Partially implemented
Client	Partially implemented

AdAgency class:

adAvailabilityArr:

- This is a static array (String[]) which contains *availableServiceArea* (String) as its elements
- The initial value of the **adAvailabilityArr** is as below:

adAvailabilityArr	{"Product", "Service", "Marketing", "Brand"}

Note:

- This array is supplied and hence, no need to code
- Do not change the CASE of the elements in the array

generateBookingId():

- This method auto-generates and sets **bookingId** (String)
- The **bookingId** must be prefixed by first character of **adRequirement** in uppercase followed by the auto-generated value starting from
- The auto-generated value would be incremented by one for the next **bookingId**

generateBookingId():

- This method auto-generates and sets **bookingId** (String)
- The **bookingId** must be prefixed by first character of **adRequirement** in uppercase followed by the auto-generated value starting from 101
- The auto-generated value would be incremented by one for the next **bookingId**
- Use static variable **counter** appropriately to implement the auto-generation logic

Example: The first **bookingId** would be **S101** if the **adRequirement** is service, the second would be **B102** if the **adRequirement** is Brand and so on

identifyWaveOffPercent():

- This method finds and returns the *waveOffPercent* (Integer) based on the **budget** (int) provided by the client
- Refer the below table to identify the *waveOffPercent*

budget	*waveOffPercent*
Less than or equal to 200000	0
Between 200000 (excluded) and 500000 (included)	5
Greater than 500000	10

Example: If the **budget** is 500000, then the *waveOffPercent* would be **5**

Client class:

validateClientAdRequirement():

- This method checks if the **adRequirement** (String) is present in the **adAvailabilityArr** of **AdAgency** class and returns a Boolean value
- If it is present then, return true
- Otherwise, return false

Note: Perform **case-insensitive** string comparison

Example: If **adRequirement** is ﷺserviceﷺ then, above method returns true

02:03:14

OutdoorAdAgency class:

outdoorAdTypeArr:

- This is a static array (String[]) which has *adType* (String) as its elements
- The initial value of the **outdoorAdTypeArr** is as below:

outdoorAdTypeArr	{"Billboard", "Transit", "Banner"}

YXNoaWxhLm1hdGhld3M=

Note:

- This array is supplied and hence, no need to code
- Do not change the CASE of the elements in the array

outdoorAdTypeCostArr:

- This is a static array (int[]) which has *costPerAdType*(int) as its elements
- This array has one to one correspondence with **outdoorAdTypeArr**
- The initial value of the **outdoorAdTypeCostArr** is as below:

outdoorAdTypeCostArr	{250, 170, 150}

Note:

- This array is supplied and hence, no need to code

calculateTotalBill():

- This method generates and sets the **bookingId** (String) and calculates and sets the **totalBillAmount** (int) to be paid by the client based on the following logic:
- Invoke **validateClient()** and **validateClientAdRequirement()** methods of **Client** class
- If both the methods return true,
  - If **outdoorAdType** is present as one of the elements in **outdoorAdTypeArr**,
    - Identify the corresponding *costPerAdType* (int) for the **outdoorAdType** from **outdoorAdTypeCostArr**

calculateTotalBill():

- This method generates and sets the **bookingId** (String) and calculates and sets the **totalBillAmount** (int) to be paid by the client based logic:
- Invoke **validateClient()** and **validateClientAdRequirement()** methods of **Client** class
- If both the methods return true,
  - If **outdoorAdType** is present as one of the elements in **outdoorAdTypeArr**,
    - Identify the corresponding *costPerAdType* (int) for the **outdoorAdType** from **outdoorAdTypeCostArr**
    - Invoke **generateBookingId()** method
    - Invoke the **identifyWaveOffPercent()** method to find the *waveoffPercent*
    - Calculate the *adCost* (int) as a product of *costPerAdType*, **quantity** ordered and **displayTimeFrame** (int) required (in months)
    - If the **adPackage** (String) is **Digital**, then add additional currency 5000 to *adCost*, otherwise if the **adPackage** is **Traditional** then add additional currency 2000 to *adCost*
    - In addition, if the **clientCollab** (boolean) is true and the **paymentType** (String) of client is **Prepaid**, then apply the *waveOffPercent* on the obtained *adCost*

      Note:
      - Perform **case-sensitive** comparison for **outdoorAdType**
      - The valid values of **paymentType** are either **Prepaid** or **Postpaid**. Perform **case-sensitive** comparison for **paymentType**
      - The valid values of **adPackage** are either **Digital** or **Traditional**. Perform **case-insensitive** comparison for **adPackage**
    - Set the **totalBillAmount** to be paid with the obtained *adCost*
    - If the **totalBillAmount** is exceeding the **budget** provided, set the **bookingId** to **NA** and **totalBillAmount** to -1
  - Otherwise, set the **bookingId** to**NA**and **totalBillAmount** to -1
- Otherwise, set the **bookingId** to **NA** and **totalBillAmount** to -1

Assumption: The valid values would be passed for **adPackage** and **paymentType**

Note: No need to validate the assumption

Example:

If the **clientName** is **GoIndia**, **adRequirement** is **service**, **budget** is currency 500000, **paymentType** is **Prepaid**, **adPackage** is **Digital**, **clientColla** is true, **outdoorAdType** is **Banner**, **quantity** required is 500 for a **displayTimeFrame** of 6 months, then the **bookingId** for the client would be **S101** (assuming first client) and the **totalBillAmount** to be paid would be currency **432250.**

Question 3: Data Structures

(assuming first client) and the **totalBillAmount** to be paid would be currency **432250**.

Question 2: Data Structures:
[5 Marks]

02:02:44

Problem Statement:

Description: Consider an **inStrQueue** (String Queue) containing non-empty strings as its elements.

YXNoaWxhLm1hdGhld3M=

Write a Java function that accepts above **inStrQueue**as input parameterand returns a non-empty **outStrStack** (String Stack)based on the logic given below:

- Consider the elements of **inStrQueue** from front to rear. Each element of **inStrQueue** is a string that represents a *card* among the deck of 52 cards.
- The *card* ends with **♦C♦**, **♦S♦**, **♦D♦** or **♦H♦** which denote the suits- Clubs, Spades, Diamonds and Hearts respectively, preceded by **♦Q♦**, **♦A♦**, **♦K♦** or **♦J♦** denoting Queen, Ace, King and Jack respectively or ranks ranging from **2** to **10.** Apart from these, the *card* can be **♦joker♦** element.
  Example: **♦2H♦** denotes **♦**Two of Hearts**♦**, **♦QS♦** denotes **♦**Queen of Spades**♦**
- Obtain an **outStrStack** (Top -> Bottom) such that:
  - All the *card* elements having **♦Q♦** **♦A♦** **♦K♦** or **♦J♦** are added in the reverse order as in the **inStrQueue** (Front -> Rear) at the top of **outStrStack** followed by all the *card* elements containing ranks **2** to **10** addedin the same order as in **inStrQueue** (Front -> Rear), followed by all the **♦joker♦** *card* elements added at the bottom of the **outStrStack** in the same order as in **inStrQueue** (Front -> Rear)

Note: Perform **case-insensitive** comparison for **♦joker♦** elements and **case-sensitive** comparison for *card* elements with **♦Q♦** **♦A♦** **♦K♦** or **♦J♦**

Assumptions:

- The size of **outStrStack** is always equal to the size of **inStrQueue**
- The occurrence of any of the 3 kinds of *card* mentioned above is not mandatory
- **inStrQueue** would not be empty
- Only valid values would be passed to **inStrQueue**

Note: No need to validate the assumptions

Example:

inStrQueue (Front -> Rear): {**♦**2C**♦**, **♦**3S**♦**, **♦**AC**♦**, **♦**9D**♦**, **♦**Joker**♦**, **♦**8S**♦**, **♦**4D**♦**, **♦**QH**♦**, **♦**JOKER**♦**, **♦**joker**♦**}

NjA2NDM5MTLwM2VmNDMzZTY5YTZiZmNiIGEzaGlsYS5tYXRoZXdz 02:02:47

- Only valid values would be passed to **inStrQueue**

Note: No need to validate the assumptions

Example:

inStrQueue (Front -> Rear): {**?**2C**?**, **?**3S**?**, **?**AC**?**, **?**9D**?**, **?**Joker**?**, **?**8S**?**, **?**4D**?**, **?**QH**?**, **?**JOKER**?**, **?**joker**?**}

YXNoaWghLm1hdGhld3M=

outStrStack (Top -> Bottom): {**?**AC**?**, **?**QH**?**, **?**4D**?**, **?**8S**?**, **?**9D**?**, **?**3S**?**, **?**2C**?**, **?**joker**?**, **?**JOKER**?**, **?**Joker**?**}

- In the above example, all the *card* elements that contain **?**joker**?** in **inStrQueue** are **?**Joker**?**, **?**JOKER**?** and **?**joker**?** (in the order they appear from Front -> Rear). Hence, add the *card* to **outStrStack** in the same order as in the **inStrQueue** (Front -> Rear).
  Now, the **outStrStack** (Top -> Bottom) would be {**?joker?**, **?JOKER?**, **?Joker?**}
- The *card* elements that contain ranks 2 to 10 in the **inStrQueue** are **?**2C**?**, **?**3S**?**, **?**9D**?**, **?**8S**?**, **?**4D**?** (in the order they appear). Hence, add the *card* to **outStrStack** in the same order as in the **inStrQueue** (Front -> Rear).
  Now, the **outStrStack** (Top -> Bottom) would be {**?4D?**, **?8S?**, **?9D?**, **?3S?**, **?2C?**, **?**joker**?**, **?**JOKER**?**, **?**Joker**?**}
- Next, the *card* elements that contain **?Q?**, **?A?**, **?K?** or **?J?** in the **inStrQueue** are **?**AC**?** and **?**QH**?**. Hence, add the *card* to **outStrStack** in the reverse order as in the **inStrQueue** (Front -> Rear).
  Now, the **outStrStack** (Top -> Bottom) would be {**?AC?**, **?QH?**, **?**4D**?**, **?**8S**?**, **?**9D**?**, **?**3S**?**, **?**2C**?**, **?**joker**?**, **?**JOKER**?**, **?**Joker**?**}

Sample Input and Output:

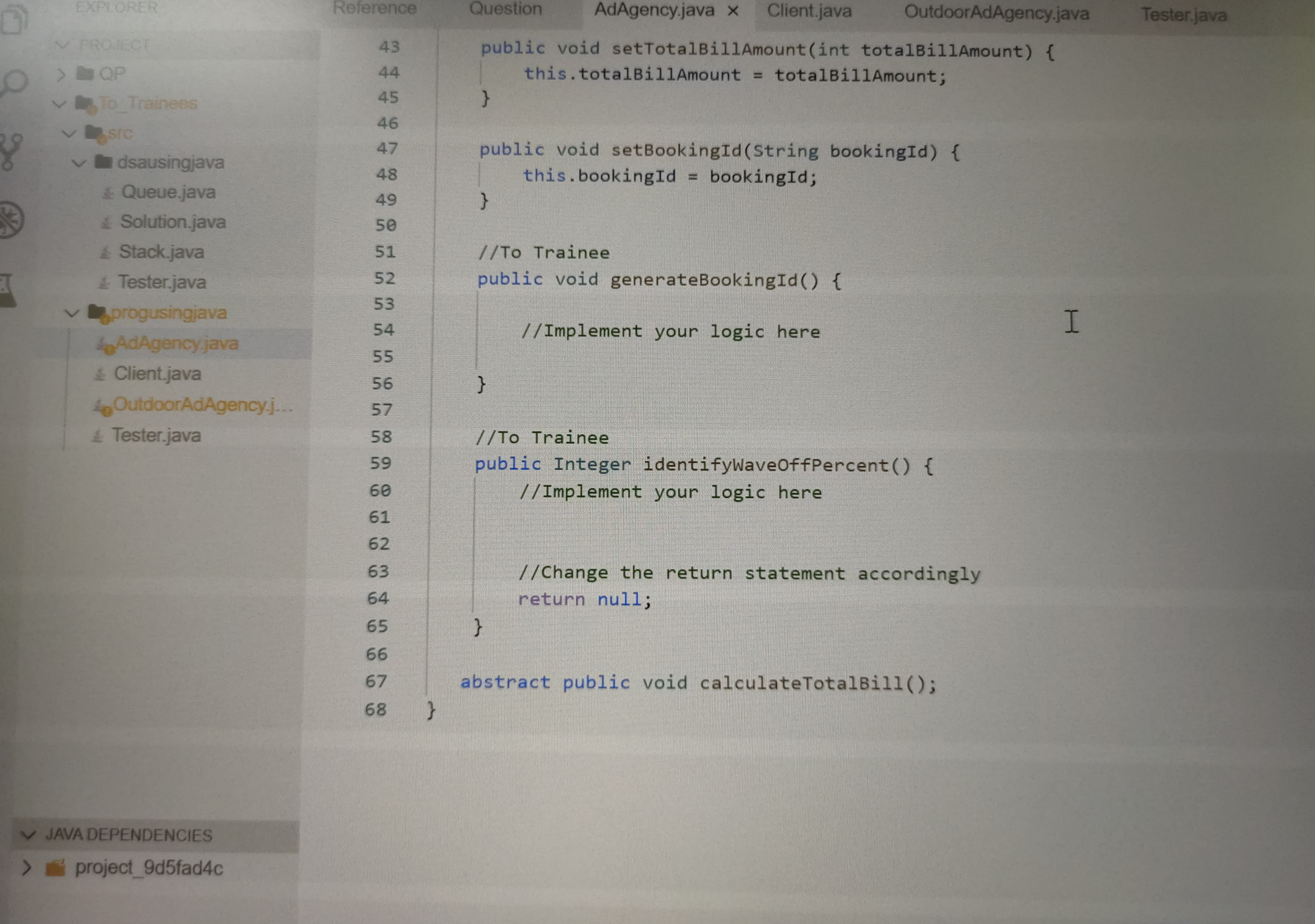
inStrQueue (Front → Rear)	outStrStack (Top → Bottom)
{"8C", "5S", "JOKER", "3H"}	{"3H", "5S", "8C", "JOKER"}
{"AS", "AC", "10H", "QD"}	{"AS", "AC", "QD", "10H"}
{"Joker", "QC", "KC"}	{"QC", "KC", "Joker"}
{"QD", "KH", "2D", "10S", "joker", "JC", "joker"}	{"QD", "KH", "JC", "10S", "2D", "joker", "joker"}

WISH YOU ALL THE BEST

Reference Question AdAgency.java × Client.java OutdoorAdAgency.java Tester.java

```java
package progusingjava;

abstract class AdAgency {
    private static int counter = 100;
    private static String[] adAvailabilityArr = {"Product", "Service", "Marketing", "Brand"};

    private Client client;
    private String adPackage;
    private boolean clientCollab;
    private int totalBillAmount;
    private String bookingId;

    public AdAgency(Client client, String adPackage, boolean clientCollab) {
        this.client = client;
        this.adPackage = adPackage;
        this.clientCollab = clientCollab;
    }

    public Client getClient() {
        return this.client;
    }

    public String getAdPackage() {
        return this.adPackage;
    }

    public boolean getClientCollab() {
        return this.clientCollab;
    }

    public int getTotalBillAmount() {
        return this.totalBillAmount;
    }
```

```
35      public String getBookingId() {
36          return this.bookingId;
37      }
38
39      public static String[] getAdAvailabilityArr(){
40          return AdAgency.adAvailabilityArr;
41      }
42
43      public void setTotalBillAmount(int totalBillAmount) {
44          this.totalBillAmount = totalBillAmount;
45      }
46
47      public void setBookingId(String bookingId) {
48          this.bookingId = bookingId;
49      }
50
51      //To Trainee
52      public void generateBookingId() {
53
54          //Implement your logic here
55
56      }
57
58      //To Trainee
59      public Integer identifyWaveOffPercent() {
60          //Implement your logic here
61
62
63          //Change the return statement accordingly
64          return null;
65      }
66
```


EXPLORER
Reference
Question
AdAgency.java
Client.java
OutdoorAdAgency.java
Tester.java
PROJECT
QP
To_Trainees
src
dsausingjava
Queue.java
Solution.java
Stack.java
Tester.java
progusingjava
AdAgency.java
Client.java
OutdoorAdAgency.j...
Tester.java
JAVA DEPENDENCIES
project_9d5fad4c
43 public void setTotalBillAmount(int totalBillAmount) {
44 this.totalBillAmount = totalBillAmount;
45 }
46
47 public void setBookingId(String bookingId) {
48 this.bookingId = bookingId;
49 }
50
51 //To Trainee
52 public void generateBookingId() {
53
54 //Implement your logic here
55
56 }
57
58 //To Trainee
59 public Integer identifyWaveOffPercent() {
60 //Implement your logic here
61
62
63 //Change the return statement accordingly
64 return null;
65 }
66
67 abstract public void calculateTotalBill();
68 }

```java
public class Client {
    private String clientName;
    private String adRequirement;
    private int budget;
    private String paymentType;

    public Client(String clientName, String adRequirement, int budget, String paymentType) {
        this.clientName = clientName;
        this.adRequirement = adRequirement;
        this.budget = budget;
        this.paymentType = paymentType;
    }

    public int getBudget() {
        return this.budget;
    }

    public String getPaymentType() {
        return this.paymentType;
    }

    public String getAdRequirement() {
        return this.adRequirement;
    }

    public boolean validateClient() {
        if (this.clientName.length()>=3) {
            return true;
        }
        return false;
    }
```

```
22      }
23
24      public String getAdRequirement() {
25          return this.adRequirement;
26      }
27
28      public boolean validateClient() {
29          if (this.clientName.length()>=3) {
30              return true;
31          }
32          return false;
33      }
34
35      //To Trainee
36      public Boolean validateClientAdRequirement() {
37          //Implement your logic here
38
39
40          //Change the return statement accordingly
41          return null;
42      }
43  }
```

Reference Question AdAgency.java Client.java × OutdoorAdAgency.java × Tester.java

```
22      }
23
24      public String getAdRequirement() {
25          return this.adRequirement;
26      }
27
28      public boolean validateClient() {
29          if (this.clientName.length()>=3) {
30              return true;
31          }
32          return false;
33      }
34
35      //To Trainee
36      public Boolean validateClientAdRequirement() {
37          //Implement your logic here
38
39
40          //Change the return statement accordingly
41          return null;
42      }
43  }
```

Run Terminal Exam Help

ference Question AdAgency.java Client.java OutdoorAdAgency.java Tester.java ×

```java
package progusingjava;

public class Tester {
    Run | Debug
    public static void main(String[] args) {
        Client clientOb1 = new Client("GoIndia", "service", 500000, "Prepaid");
        OutdoorAdAgency adOb1 = new OutdoorAdAgency(clientOb1, "Digital", true, "Banner", 6, 500
        adOb1.calculateTotalBill();
        System.out.println("Booking Id: "+adOb1.getBookingId());
        System.out.println("Bill amount: "+adOb1.getTotalBillAmount());
    }
}
```

```java
package dsausingjava;

//DO NOT MODIFY THE CODE PROVIDED TO YOU

public class Queue {
    private int front, rear, maxSize;
    private String[] arr;

    Queue(int maxSize) {
        this.maxSize = maxSize;
        this.front = 0;
        this.rear = -1;
        this.arr = new String[maxSize];
    }

    public int getMaxSize() {
        return this.maxSize;
    }

    public boolean isFull() {
        if (rear == maxSize-1)
            return true;
        return false;
    }

    public boolean enqueue(String data) {
        if(isFull())
            return false;
        else {
            arr[++rear] = data;
            return true;
        }
    }
```

```
28              return false;
29          else {
30              arr[++rear] = data;
31              return true;
32          }
33      }
34
35      public boolean isEmpty() {
36          if (front > rear)
37              return true;
38          return false;
39      }
40
41      public String dequeue() {
42          if (isEmpty())
43              return null;
44          else
45              return arr[front++];
46      }
47
48      public void display() {
49          if (isEmpty())
50              System.out.println("Queue is empty!");
51          else {
52              System.out.println("Displaying Queue elements:");
53              for (int ind = front; ind <= rear; ind++)
54                  System.out.println(arr[ind]);
55          }
56      }
57  }
```

```java
package dsausingjava;

//DO NOT MODIFY THE CODE PROVIDED TO YOU

Verify
public class Solution {
    public Stack rearrangeDeck(Queue inStrQueue) {
        Stack outStrStack = new Stack(inStrQueue.getMaxSize());

        //Implement your logic here

        return outStrStack;
    }
}
```

```java
package dsausingjava;

//DO NOT MODIFY THE CODE PROVIDED TO YOU

public class Stack {
    private int top;
    private int maxSize;
    private String[] arr;

    public Stack(int maxSize) {
        this.top = -1;
        this.maxSize = maxSize;
        this.arr = new String[maxSize];
    }

    public int getMaxSize() {
        return this.maxSize;
    }

    public boolean isFull() {
        if (top >= maxSize-1)
            return true;
        return false;
    }

    public boolean push(String data) {
        if(isFull())
            return false;
        else {
            arr[++top] = data;
            return true;
        }
    }
```

```
34
35      public boolean isEmpty() {
36          if (top < 0)
37              return true;
38          return false;
39      }
40
41      public String peek() {
42          if (isEmpty())
43              return null;
44          return arr[top];
45      }
46
47      public String pop() {
48          if (isEmpty())
49              return null;
50          return arr[top--];
51      }
52
53      public void display() {
54          if (isEmpty())
55              System.out.println("Stack is empty!");
56          else {
57              System.out.println("Displaying Stack elements:");
58              for (int ind = top; ind >=0; ind--)
59                  System.out.println(arr[ind]);
60          }
61      }
62  }
63
```

Client.java | OutdoorAdAgency.java | Tester.java .../progusingjava | Queue.java | Solution.java | Stack.java | Tester.java .../dsausingjava

```java
package dsausingjava;

public class Tester {
    Run | Debug
    public static void main(String[] args) {
        Solution ob = new Solution();
        Queue inStrQueue = new Queue(10);
        inStrQueue.enqueue("2C");
        inStrQueue.enqueue("3S");
        inStrQueue.enqueue("AC");
        inStrQueue.enqueue("9D");
        inStrQueue.enqueue("Joker");
        inStrQueue.enqueue("8S");
        inStrQueue.enqueue("4D");
        inStrQueue.enqueue("QH");
        inStrQueue.enqueue("JOKER");
        inStrQueue.enqueue("joker");

        Stack outStrStack = ob.rearrangeDeck(inStrQueue);
        outStrStack.display();
    }
}
```

NjA2NDM5MTUwM2VmNDMzZTY5YTZjZmNjIGFzaGlsYS5tYXRoZXdz