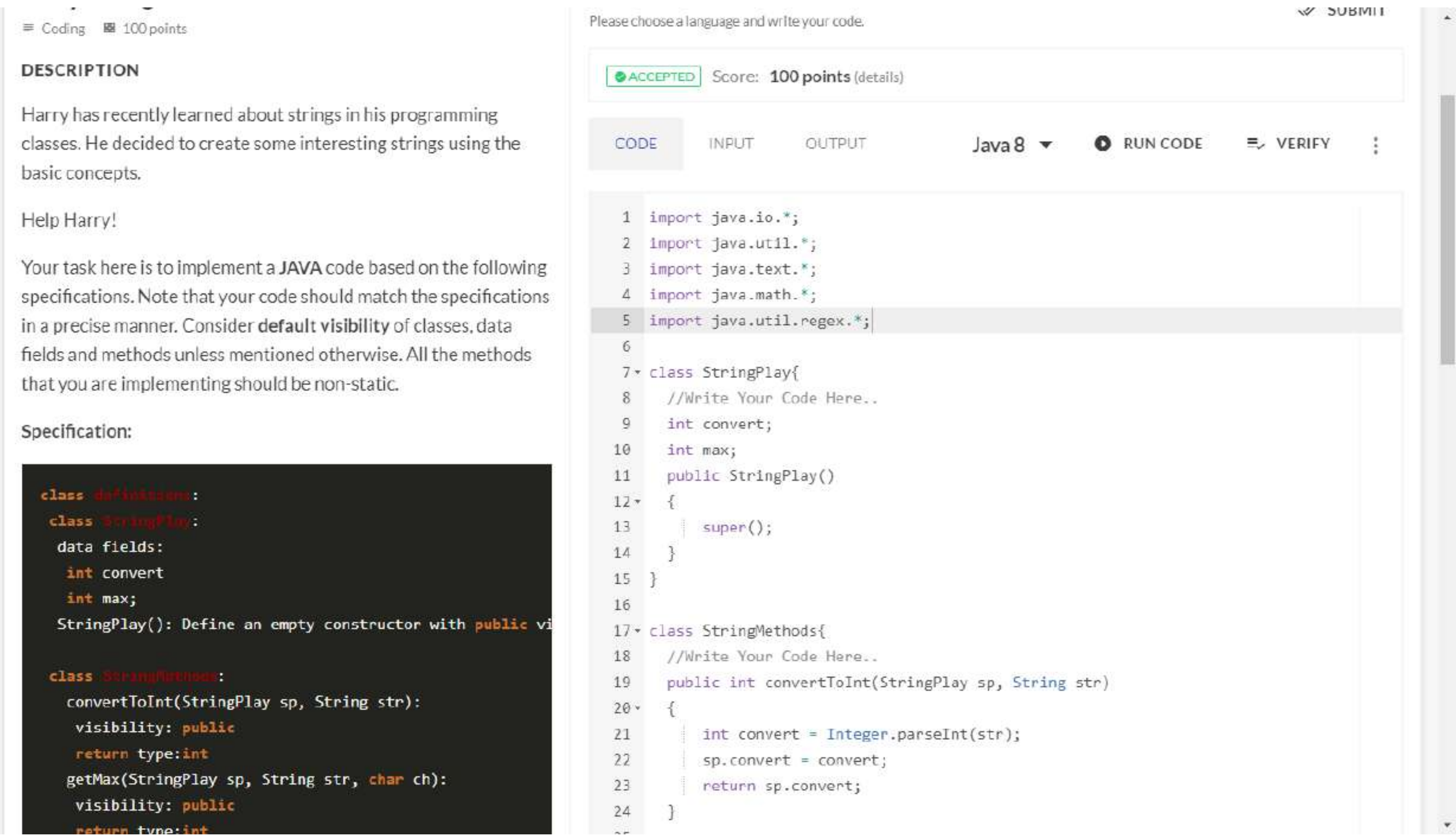


DESCRIPTION

Harry has recently learned about strings in his programming classes. He decided to create some interesting strings using the basic concepts.

Help Harry!

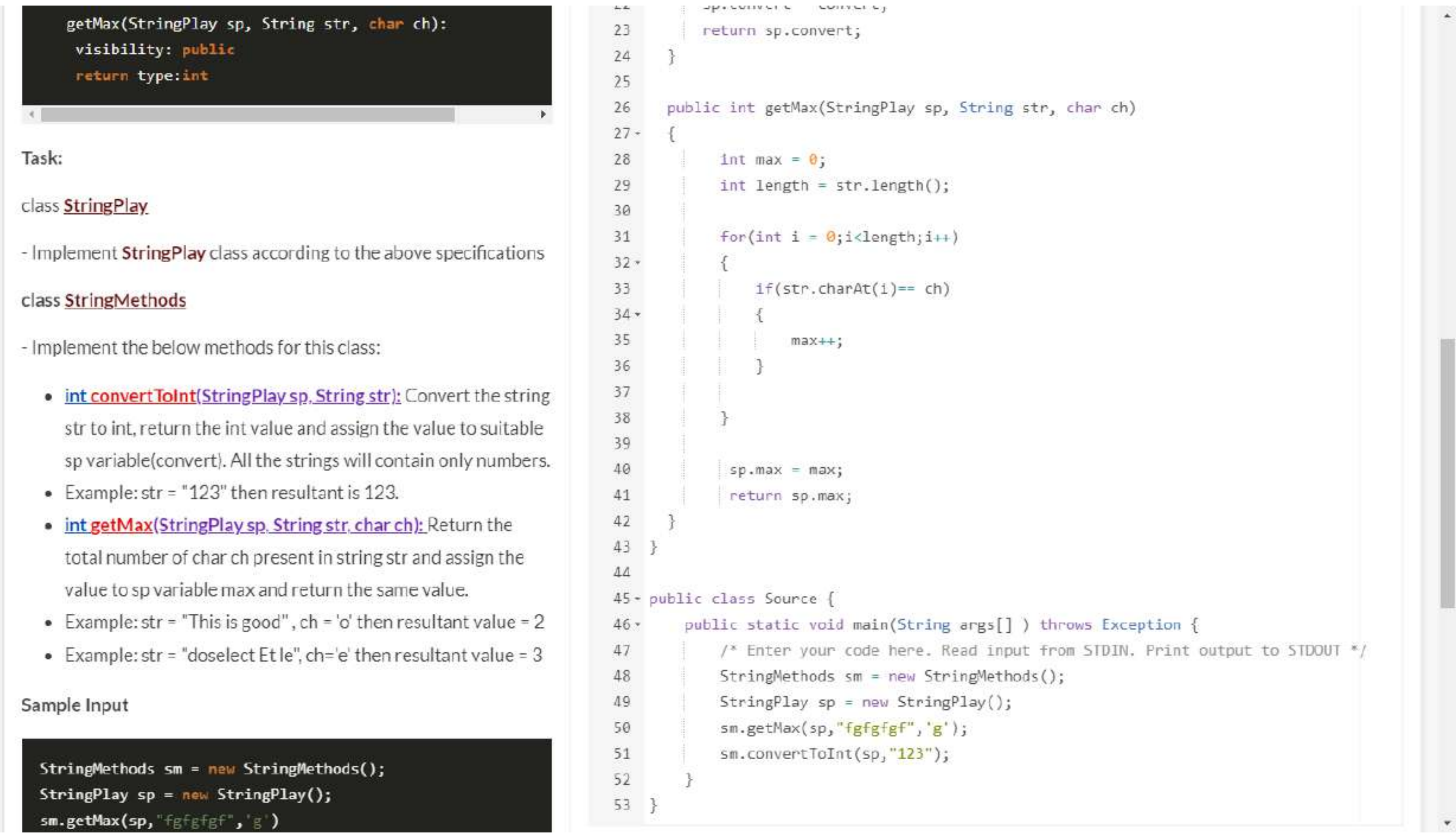
Your task here is to implement a **JAVA** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider **default visibility** of classes, data fields and methods unless mentioned otherwise. All the methods that you are implementing should be non-static.

Specification:

```
class definitions:
 class StringPlay:
  data fields:
   int convert
   int max;
  StringPlay(): Define an empty constructor with public vi

 class StringMethods:
   convertToInt(StringPlay sp, String str):
    visibility: public
    return type:int
   getMax(StringPlay sp, String str, char ch):
    visibility: public
    return type:int
```

Please choose a language and write your code.

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class StringPlay{
  //Write Your Code Here..
  int convert;
  int max;
  public StringPlay()
  {
    super();
  }
}

class StringMethods{
  //Write Your Code Here..
  public int convertToInt(StringPlay sp, String str)
  {
    int convert = Integer.parseInt(str);
    sp.convert = convert;
    return sp.convert;
  }
```



```
    getMax(StringPlay sp, String str, char ch):
     visibility: public
     return type:int
```

Task:

class StringPlay

- Implement **StringPlay** class according to the above specifications

class StringMethods

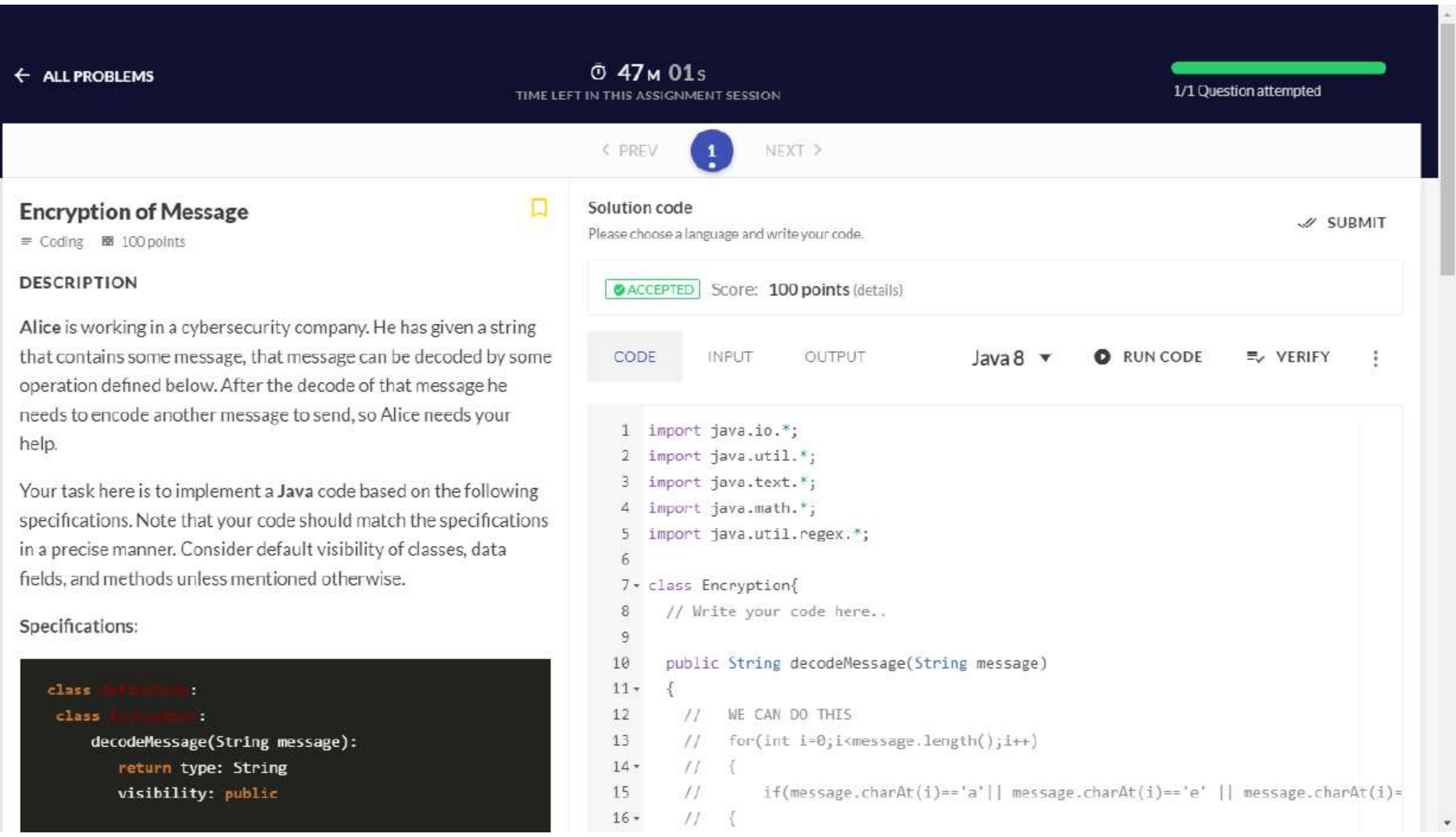
- Implement the below methods for this class:

- int convertToInt(StringPlay sp, String str): Convert the string str to int, return the int value and assign the value to suitable sp variable(convert). All the strings will contain only numbers.
- Example: str = "123" then resultant is 123.
- int getMax(StringPlay sp, String str, char ch): Return the total number of char ch present in string str and assign the value to sp variable max and return the same value.
- Example: str = "This is good" , ch = 'o' then resultant value = 2
- Example: str = "doselect Et le", ch='e' then resultant value = 3

Sample Input

```
StringMethods sm = new StringMethods();
StringPlay sp = new StringPlay();
sm.getMax(sp,"fgfgfgf",'g')
```

```
        sp.convert = convert;
        return sp.convert;
    }

    public int getMax(StringPlay sp, String str, char ch)
    {
        int max = 0;
        int length = str.length();

        for(int i = 0;i<length;i++)
        {
            if(str.charAt(i)== ch)
            {
                max++;
            }

        }

         sp.max = max;
         return sp.max;
    }
}

public class Source {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        StringMethods sm = new StringMethods();
        StringPlay sp = new StringPlay();
        sm.getMax(sp,"fgfgfgf",'g');
        sm.convertToInt(sp,"123");
    }
}
```



Encryption of Message

DESCRIPTION

Alice is working in a cybersecurity company. He has given a string that contains some message, that message can be decoded by some operation defined below. After the decode of that message he needs to encode another message to send, so Alice needs your help.

Your task here is to implement a **Java** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of classes, data fields, and methods unless mentioned otherwise.

Specifications:

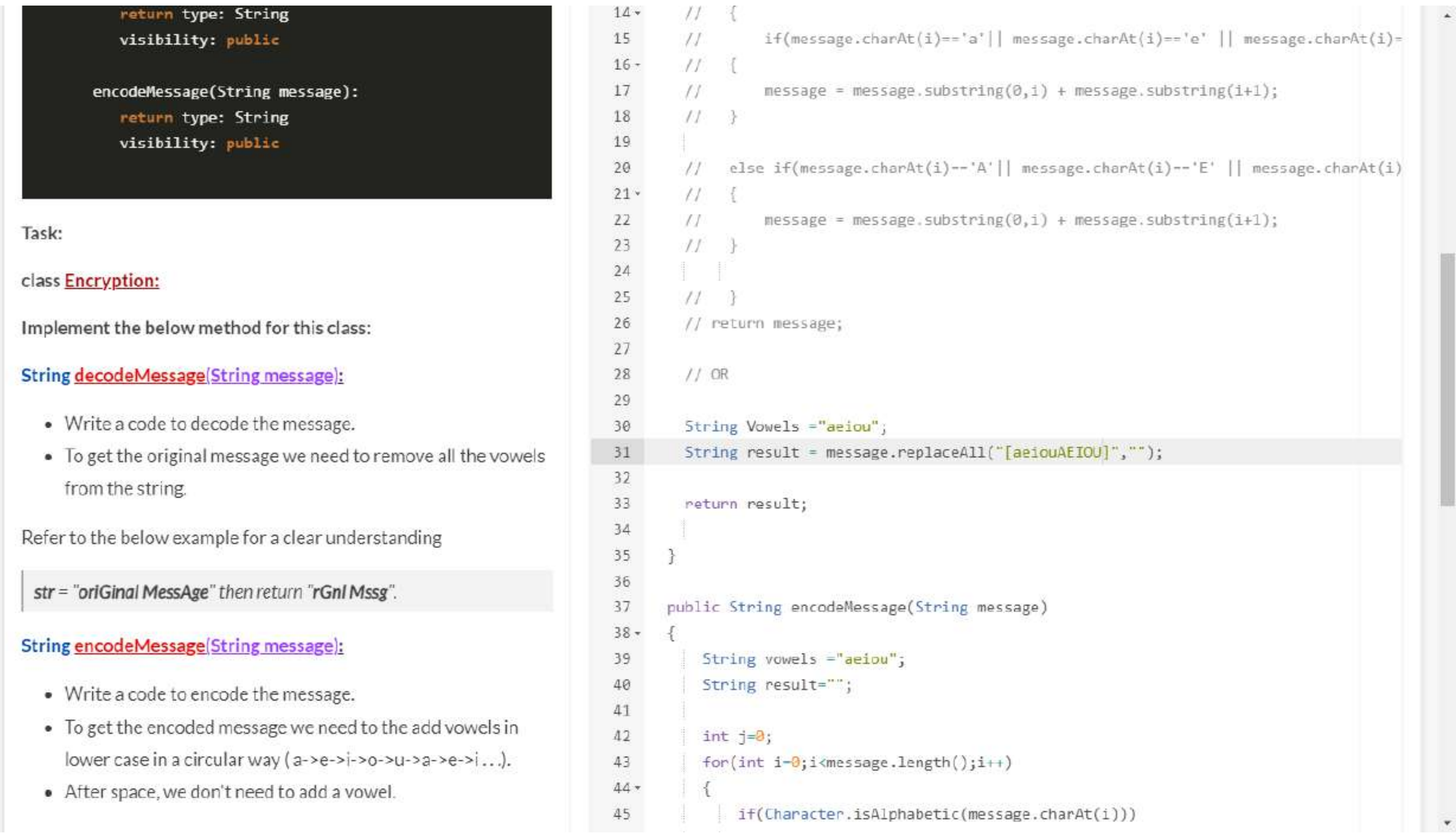
```
class Definition:
 class Encryption:
    decodeMessage(String message):
       return type: String
       visibility: public
```

Solution code

Please choose a language and write your code.

Score: **100 points** (details)

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class Encryption{
  // Write your code here..

  public String decodeMessage(String message)
  {
    //   WE CAN DO THIS
    //   for(int i=0;i<message.length();i++)
    //   {
    //       if(message.charAt(i)=='a'|| message.charAt(i)=='e'  || message.charAt(i)=
    //   {
```



```
        return type: String
        visibility: public

    encodeMessage(String message):
        return type: String
        visibility: public
```

Task:

class Encryption:

Implement the below method for this class:

String decodeMessage(String message):

- Write a code to decode the message.
- To get the original message we need to remove all the vowels from the string.

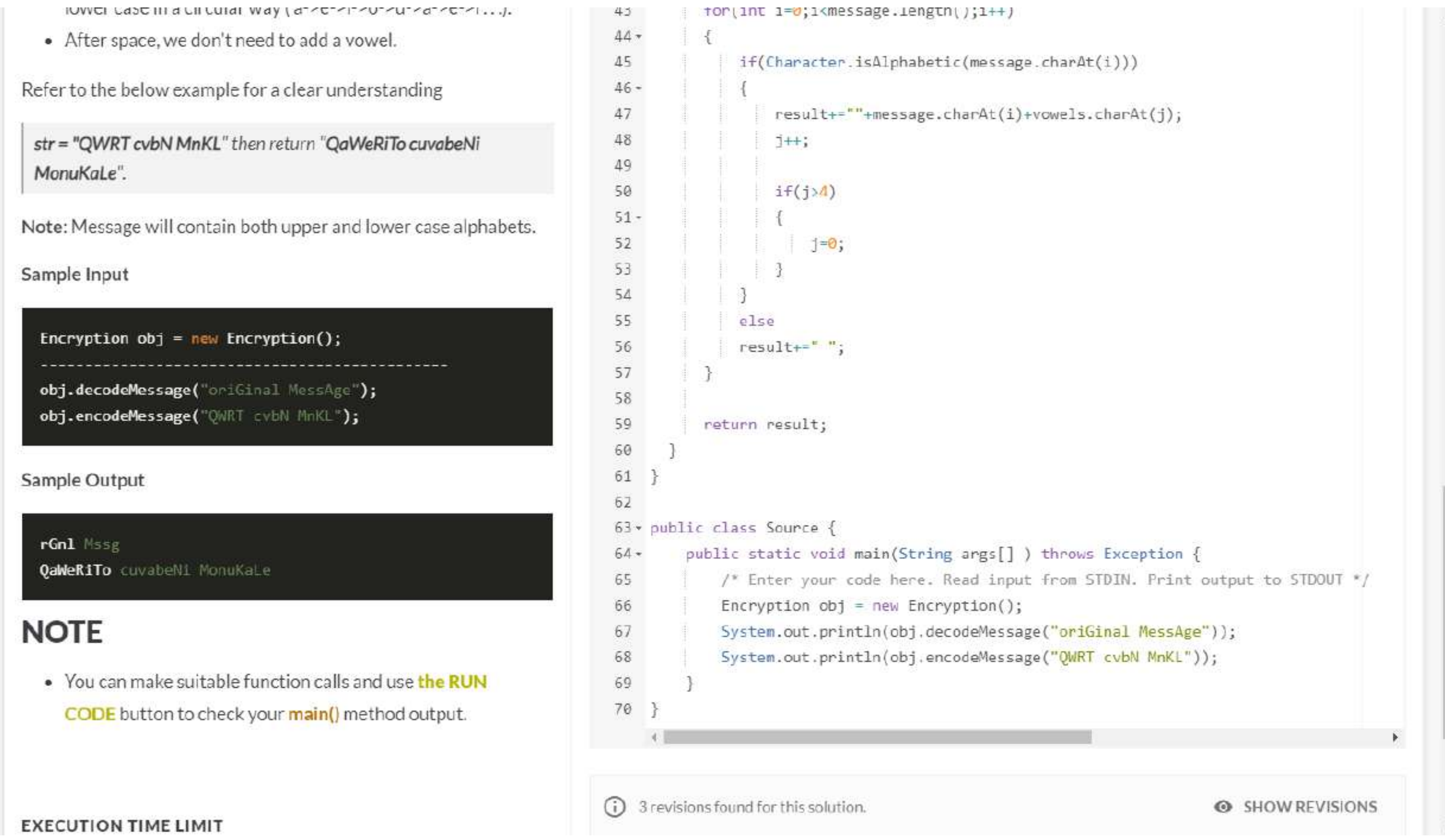
Refer to the below example for a clear understanding

> ***str** = "**oriGinal MessAge**" then return "**rGnl Mssg**".*

String encodeMessage(String message):

- Write a code to encode the message.
- To get the encoded message we need to the add vowels in lower case in a circular way (a->e->i->o->u->a->e->i...).
- After space, we don't need to add a vowel.

```
    //    {
    //        if(message.charAt(i)=='a'|| message.charAt(i)=='e'  || message.charAt(i)=
    //    {
    //        message = message.substring(0,i) + message.substring(i+1);
    //    }

    //    else if(message.charAt(i)=='A'|| message.charAt(i)=='E'  || message.charAt(i)
    //    {
    //        message = message.substring(0,i) + message.substring(i+1);
    //    }

    //    }
    // return message;

    // OR

    String Vowels ="aeiou";
    String result = message.replaceAll("[aeiouAEIOU]","");

    return result;

  }

  public String encodeMessage(String message)
  {
    String vowels ="aeiou";
    String result="";

    int j=0;
    for(int i=0;i<message.length();i++)
    {
      if(Character.isAlphabetic(message.charAt(i)))
```


lower case in a circular way (a->e->i->o->u->a->e->i...).

- After space, we don't need to add a vowel.

Refer to the below example for a clear understanding

> *str* = ***"QWRT cvbN MnKL"*** *then return "**QaWeRiTo cuvabeNi MonuKaLe**".*

Note: Message will contain both upper and lower case alphabets.

Sample Input

```
Encryption obj = new Encryption();
----------------------------------------------
obj.decodeMessage("oriGinal MessAge");
obj.encodeMessage("QWRT cvbN MnKL");
```

Sample Output

```
rGnl Mssg
QaWeRiTo cuvabeNi MonuKaLe
```

NOTE

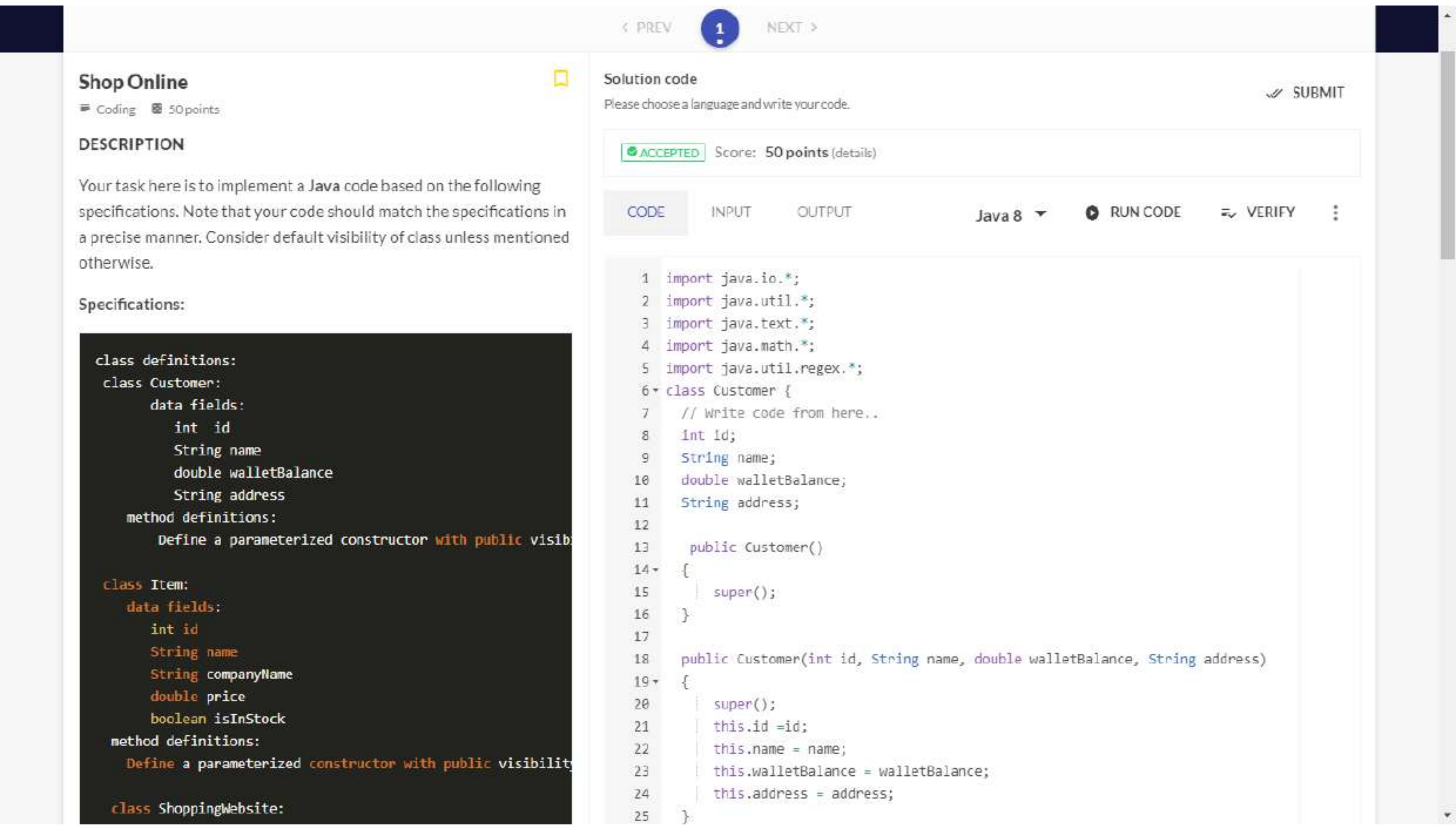
- You can make suitable function calls and use **the RUN CODE** button to check your **main()** method output.

EXECUTION TIME LIMIT

```
        for(int i=0;i<message.length();i++)
        {
            if(Character.isAlphabetic(message.charAt(i)))
            {
                result+=""+message.charAt(i)+vowels.charAt(j);
                j++;

                if(j>4)
                {
                    j=0;
                }
            }
            else
            result+=" ";
        }

        return result;
    }
}

public class Source {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        Encryption obj = new Encryption();
        System.out.println(obj.decodeMessage("oriGinal MessAge"));
        System.out.println(obj.encodeMessage("QWRT cvbN MnKL"));
    }
}
```

3 revisions found for this solution.

SHOW REVISIONS



Shop Online

Coding 50 points

DESCRIPTION

Your task here is to implement a **Java** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider default visibility of class unless mentioned otherwise.

Specifications:

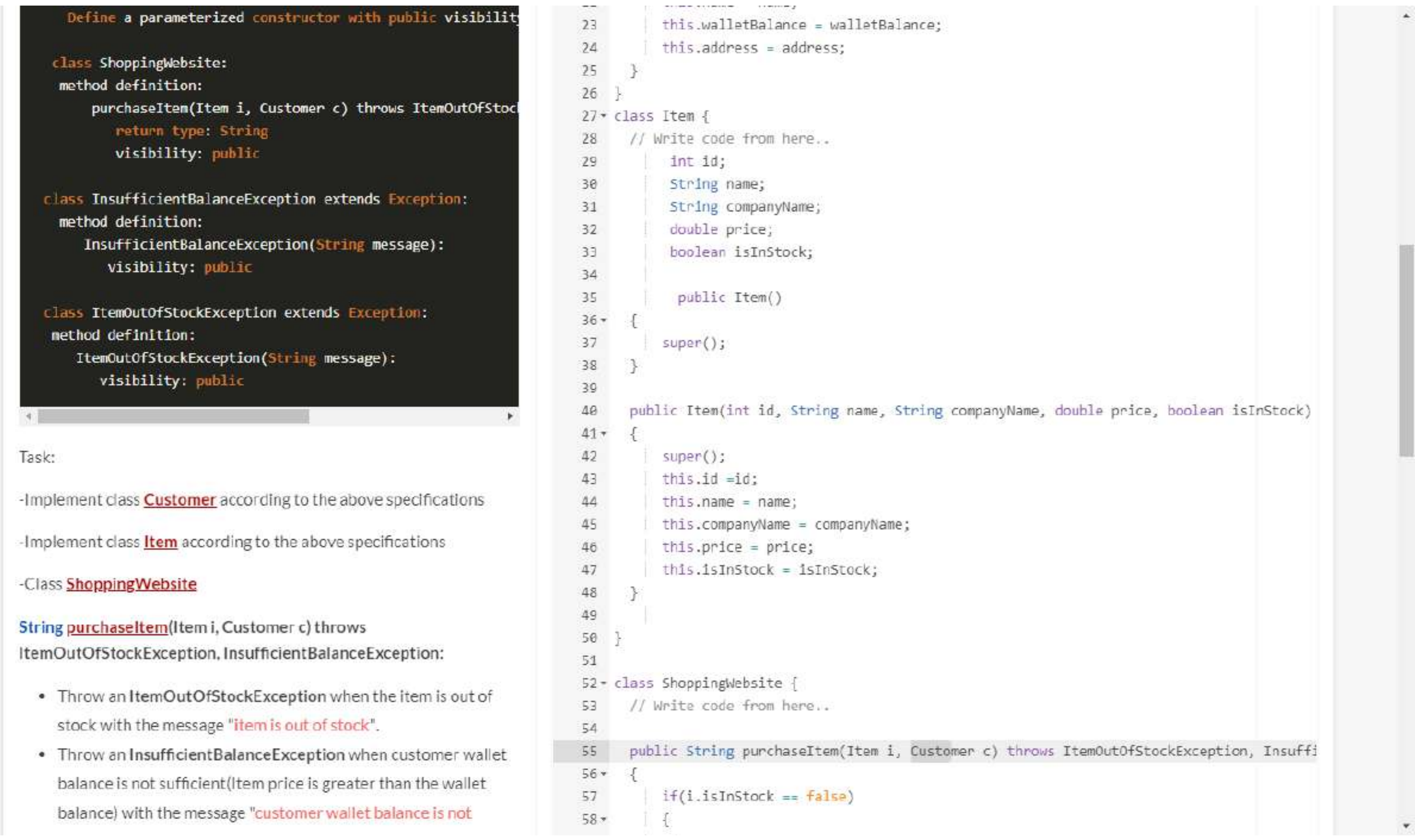
```
class definitions:
 class Customer:
       data fields:
          int  id
          String name
          double walletBalance
          String address
    method definitions:
        Define a parameterized constructor with public visib

 class Item:
    data fields:
       int id
       String name
       String companyName
       double price
       boolean isInStock
  method definitions:
    Define a parameterized constructor with public visibility

  class ShoppingWebsite:
```

Solution code

Please choose a language and write your code.

Score: **50 points** (details)

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
class Customer {
  // Write code from here..
  int id;
  String name;
  double walletBalance;
  String address;

   public Customer()
  {
      super();
  }

  public Customer(int id, String name, double walletBalance, String address)
  {
      super();
      this.id =id;
      this.name = name;
      this.walletBalance = walletBalance;
      this.address = address;
  }
```



```
    Define a parameterized constructor with public visibilit

  class ShoppingWebsite:
   method definition:
       purchaseItem(Item i, Customer c) throws ItemOutOfStoc
          return type: String
          visibility: public

 class InsufficientBalanceException extends Exception:
   method definition:
      InsufficientBalanceException(String message):
         visibility: public

 class ItemOutOfStockException extends Exception:
  method definition:
     ItemOutOfStockException(String message):
        visibility: public
```

Task:

-Implement class Customer according to the above specifications

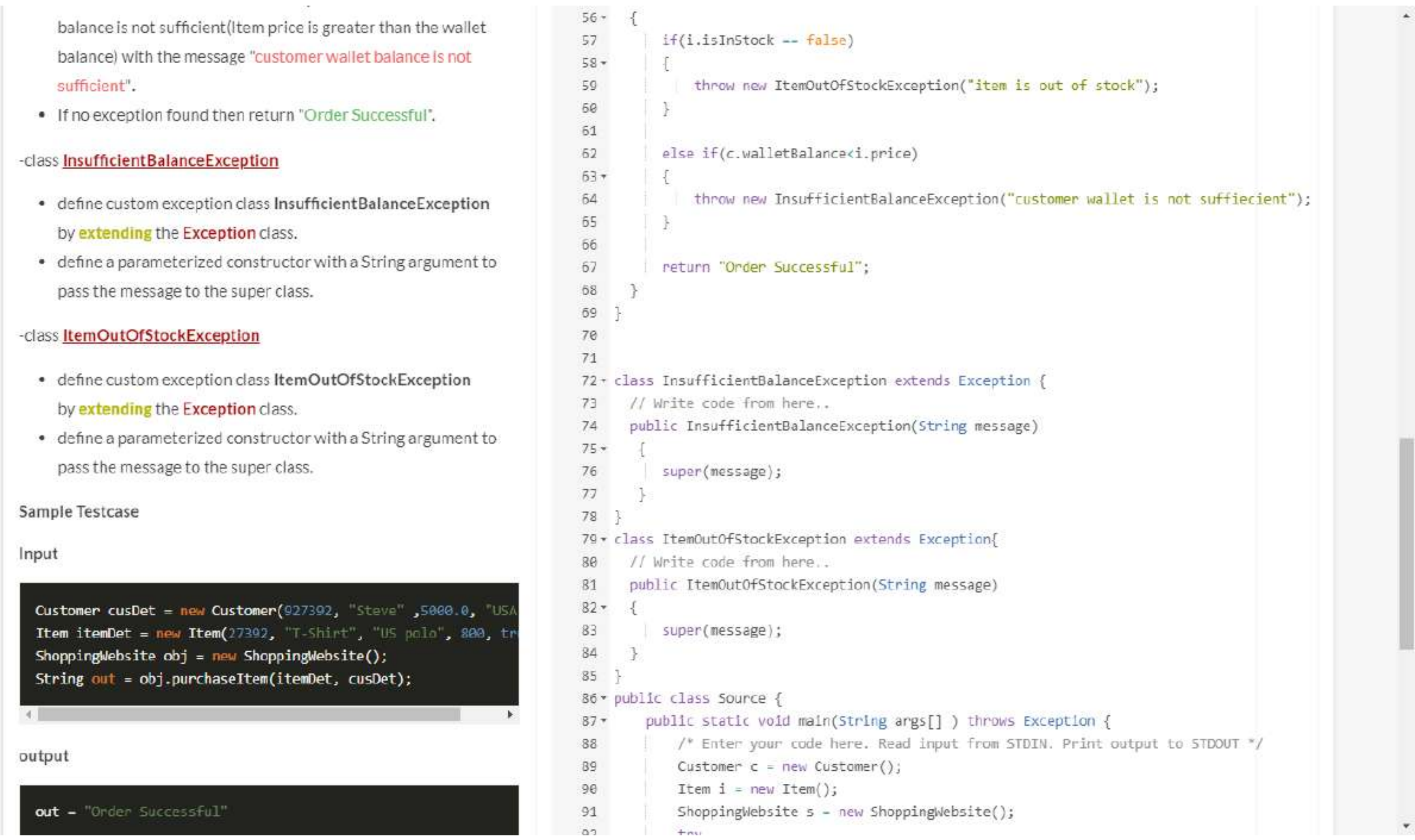
-Implement class Item according to the above specifications

-Class ShoppingWebsite

String purchaseItem(Item i, Customer c) throws ItemOutOfStockException, InsufficientBalanceException:

- Throw an **ItemOutOfStockException** when the item is out of stock with the message "item is out of stock".
- Throw an **InsufficientBalanceException** when customer wallet balance is not sufficient(Item price is greater than the wallet balance) with the message "customer wallet balance is not

```java
        this.walletBalance = walletBalance;
        this.address = address;
    }
}
class Item {
  // Write code from here..
        int id;
        String name;
        String companyName;
        double price;
        boolean isInStock;

         public Item()
    {
        super();
    }

    public Item(int id, String name, String companyName, double price, boolean isInStock)
    {
        super();
        this.id =id;
        this.name = name;
        this.companyName = companyName;
        this.price = price;
        this.isInStock = isInStock;
    }

}

class ShoppingWebsite {
  // Write code from here..

  public String purchaseItem(Item i, Customer c) throws ItemOutOfStockException, Insuffi
    {
        if(i.isInStock == false)
        {
```



balance is not sufficient(Item price is greater than the wallet balance) with the message "customer wallet balance is not sufficient".

- If no exception found then return "Order Successful".

-class **InsufficientBalanceException**

- define custom exception class **InsufficientBalanceException** by **extending** the **Exception** class.
- define a parameterized constructor with a String argument to pass the message to the super class.

-class **ItemOutOfStockException**

- define custom exception class **ItemOutOfStockException** by **extending** the **Exception** class.
- define a parameterized constructor with a String argument to pass the message to the super class.

Sample Testcase

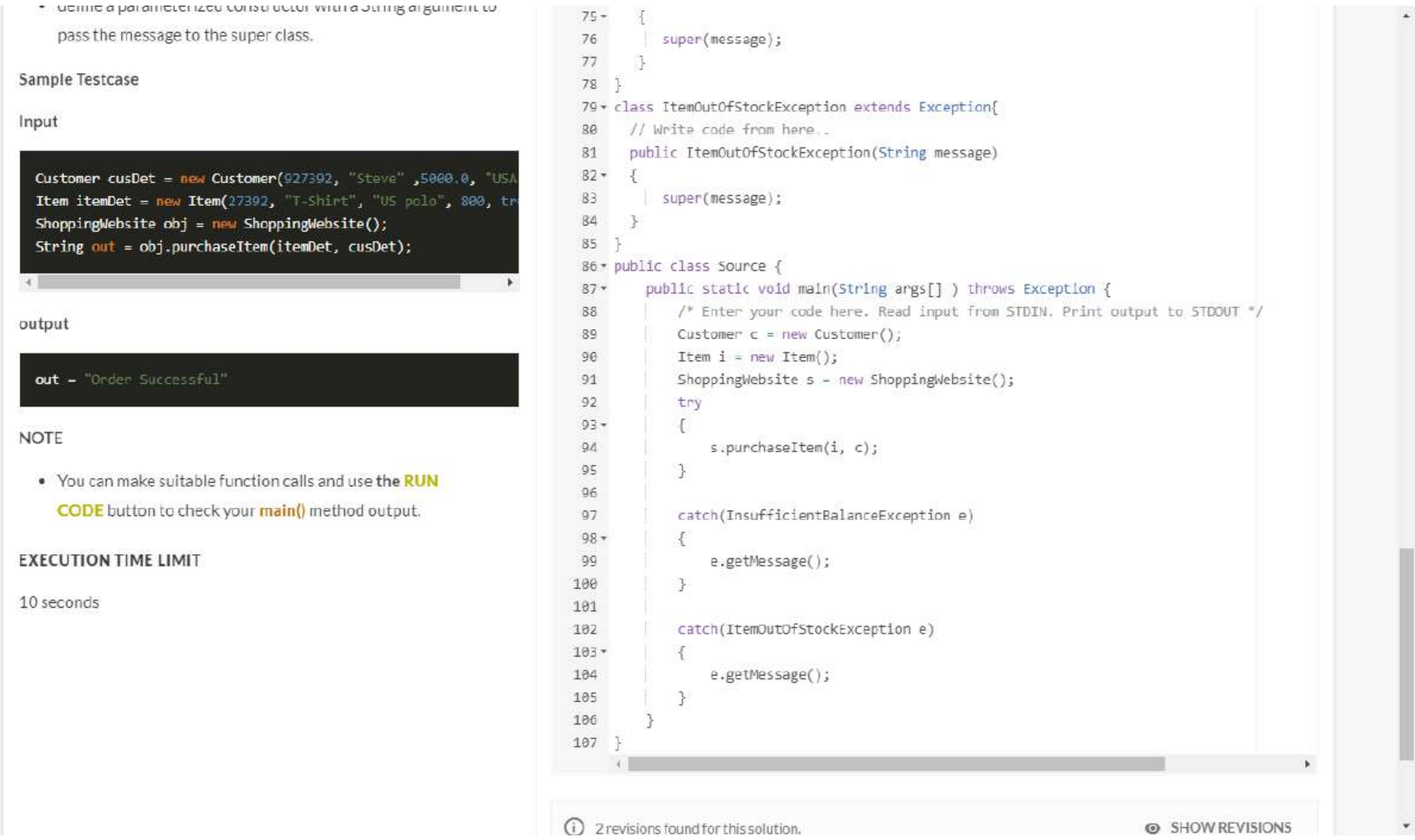
Input

```
Customer cusDet = new Customer(927392, "Steve" ,5000.0, "USA
Item itemDet = new Item(27392, "T-Shirt", "US polo", 800, tr
ShoppingWebsite obj = new ShoppingWebsite();
String out = obj.purchaseItem(itemDet, cusDet);
```

output

```
out = "Order Successful"
```

```java
  {
      if(i.isInStock == false)
      {
          throw new ItemOutOfStockException("item is out of stock");
      }

      else if(c.walletBalance<i.price)
      {
          throw new InsufficientBalanceException("customer wallet is not suffiecient");
      }

      return "Order Successful";
  }
}


class InsufficientBalanceException extends Exception {
  // Write code from here..
  public InsufficientBalanceException(String message)
   {
      super(message);
   }
}
class ItemOutOfStockException extends Exception{
  // Write code from here..
  public ItemOutOfStockException(String message)
  {
      super(message);
  }
}
public class Source {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        Customer c = new Customer();
        Item i = new Item();
        ShoppingWebsite s = new ShoppingWebsite();
        try
```



- define a parameterized constructor with a String argument to pass the message to the super class.

Sample Testcase

Input

```
Customer cusDet = new Customer(927392, "Steve" ,5000.0, "USA
Item itemDet = new Item(27392, "T-Shirt", "US polo", 800, tr
ShoppingWebsite obj = new ShoppingWebsite();
String out = obj.purchaseItem(itemDet, cusDet);
```

output

```
out = "Order Successful"
```

NOTE

- You can make suitable function calls and use **the RUN CODE** button to check your **main()** method output.

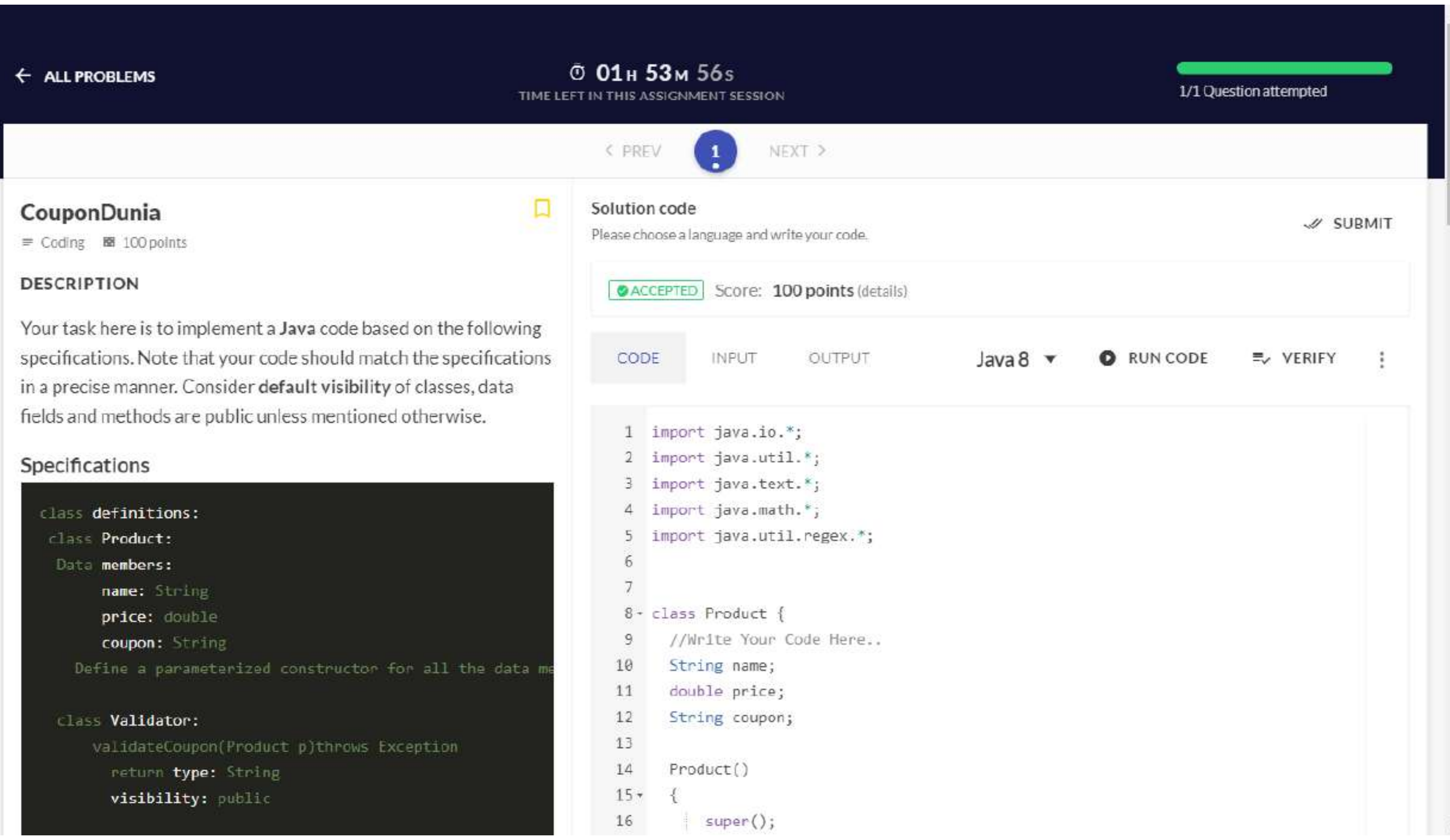
EXECUTION TIME LIMIT

10 seconds

```
    {
        super(message);
    }
}
class ItemOutOfStockException extends Exception{
  // Write code from here..
  public ItemOutOfStockException(String message)
  {
      super(message);
  }
}
public class Source {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        Customer c = new Customer();
        Item i = new Item();
        ShoppingWebsite s = new ShoppingWebsite();
        try
        {
            s.purchaseItem(i, c);
        }

        catch(InsufficientBalanceException e)
        {
            e.getMessage();
        }

        catch(ItemOutOfStockException e)
        {
            e.getMessage();
        }
    }
}
```

2 revisions found for this solution.

SHOW REVISIONS



CouponDunia

Coding 100 points

DESCRIPTION

Your task here is to implement a **Java** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider **default visibility** of classes, data fields and methods are public unless mentioned otherwise.

Specifications

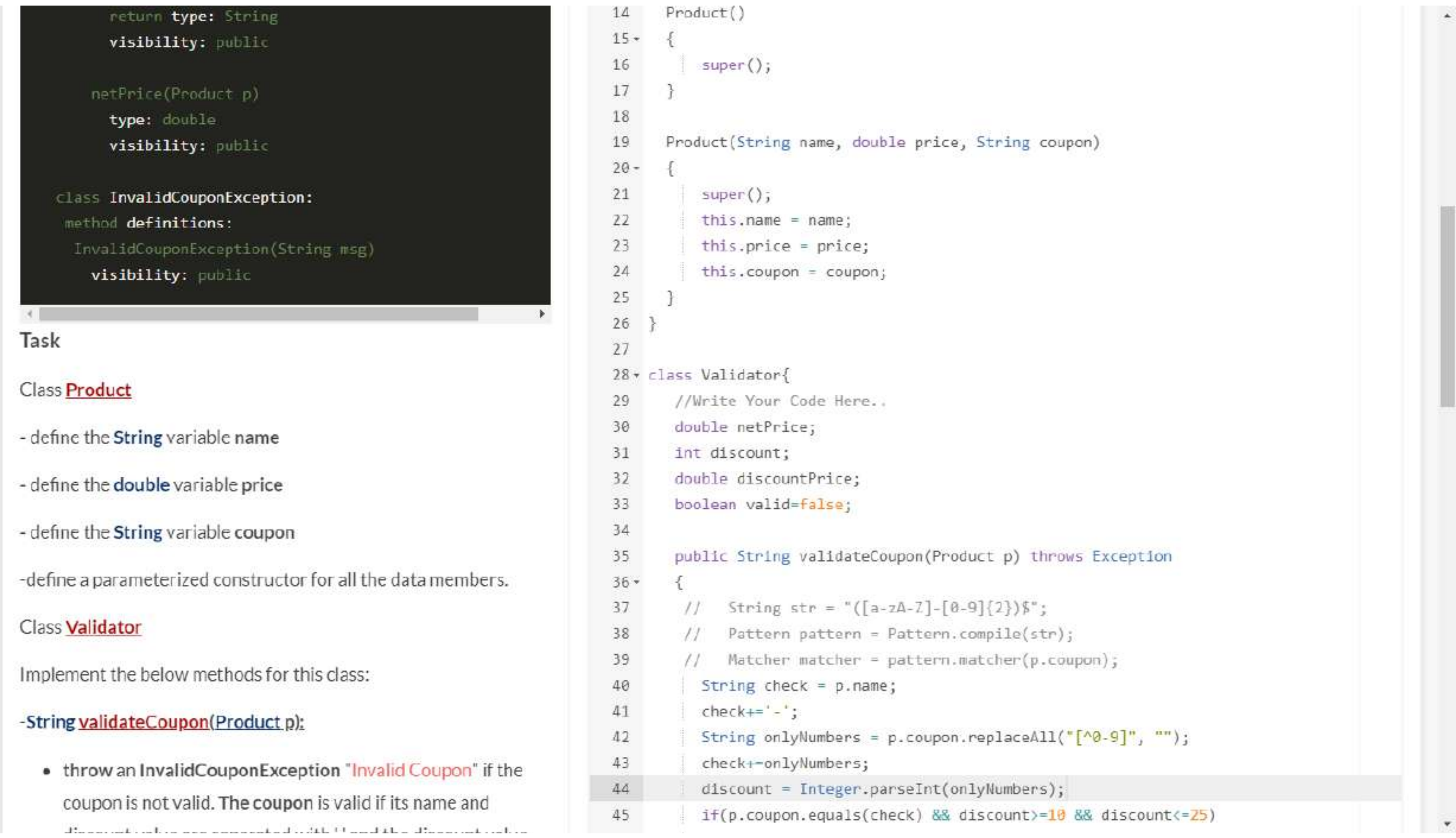
```
class definitions:
 class Product:
  Data members:
       name: String
       price: double
       coupon: String
    Define a parameterized constructor for all the data me

  class Validator:
      validateCoupon(Product p)throws Exception
        return type: String
        visibility: public
```

Solution code

Please choose a language and write your code.

ACCEPTED Score: 100 points (details)

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;


class Product {
  //Write Your Code Here..
  String name;
  double price;
  String coupon;

  Product()
  {
    super();
```



```
        return type: String
        visibility: public

      netPrice(Product p)
        type: double
        visibility: public

  class InvalidCouponException:
   method definitions:
    InvalidCouponException(String msg)
      visibility: public
```

Task

Class **Product**

- define the **String** variable **name**

- define the **double** variable **price**

- define the **String** variable **coupon**

-define a parameterized constructor for all the data members.

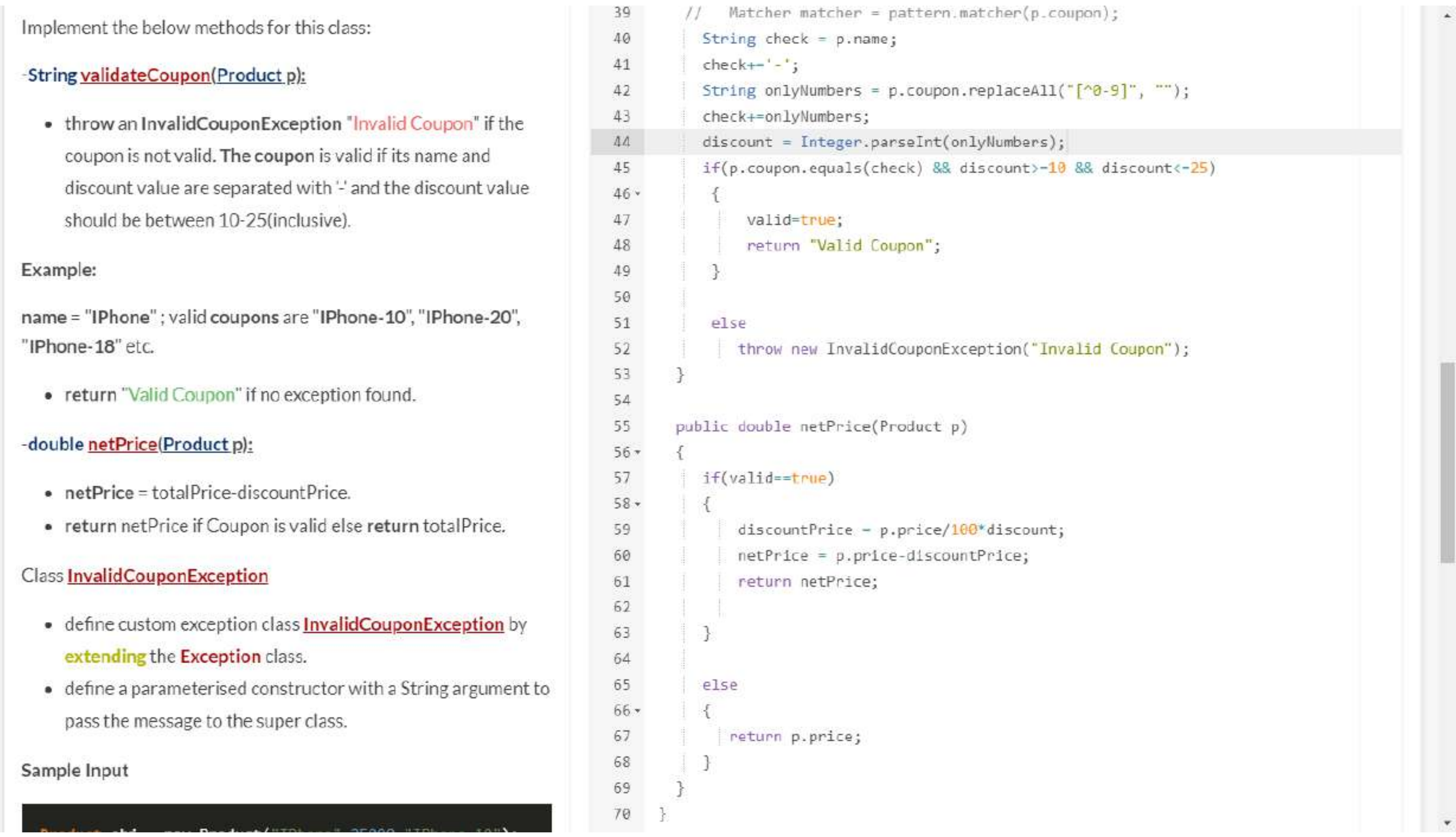
Class **Validator**

Implement the below methods for this class:

-**String validateCoupon(Product p):**

- throw an **InvalidCouponException** "Invalid Coupon" if the coupon is not valid. **The coupon** is valid if its name and

```
  Product()
  {
    super();
  }

  Product(String name, double price, String coupon)
  {
    super();
    this.name = name;
    this.price = price;
    this.coupon = coupon;
  }
}

class Validator{
  //Write Your Code Here..
  double netPrice;
  int discount;
  double discountPrice;
  boolean valid=false;

  public String validateCoupon(Product p) throws Exception
  {
   //   String str = "([a-zA-Z]-[0-9]{2})$";
   //   Pattern pattern = Pattern.compile(str);
   //   Matcher matcher = pattern.matcher(p.coupon);
    String check = p.name;
    check+='-';
    String onlyNumbers = p.coupon.replaceAll("[^0-9]", "");
    check+=onlyNumbers;
    discount = Integer.parseInt(onlyNumbers);
    if(p.coupon.equals(check) && discount>=10 && discount<=25)
```



Implement the below methods for this class:

-**String** validateCoupon(Product p):

- throw an **InvalidCouponException** "Invalid Coupon" if the coupon is not valid. **The coupon** is valid if its name and discount value are separated with '-' and the discount value should be between 10-25(inclusive).

Example:

name = "**IPhone**" ; valid **coupons** are "**IPhone-10**", "**IPhone-20**", "**IPhone-18**" etc.

- **return** "Valid Coupon" if no exception found.

-**double** netPrice(Product p):

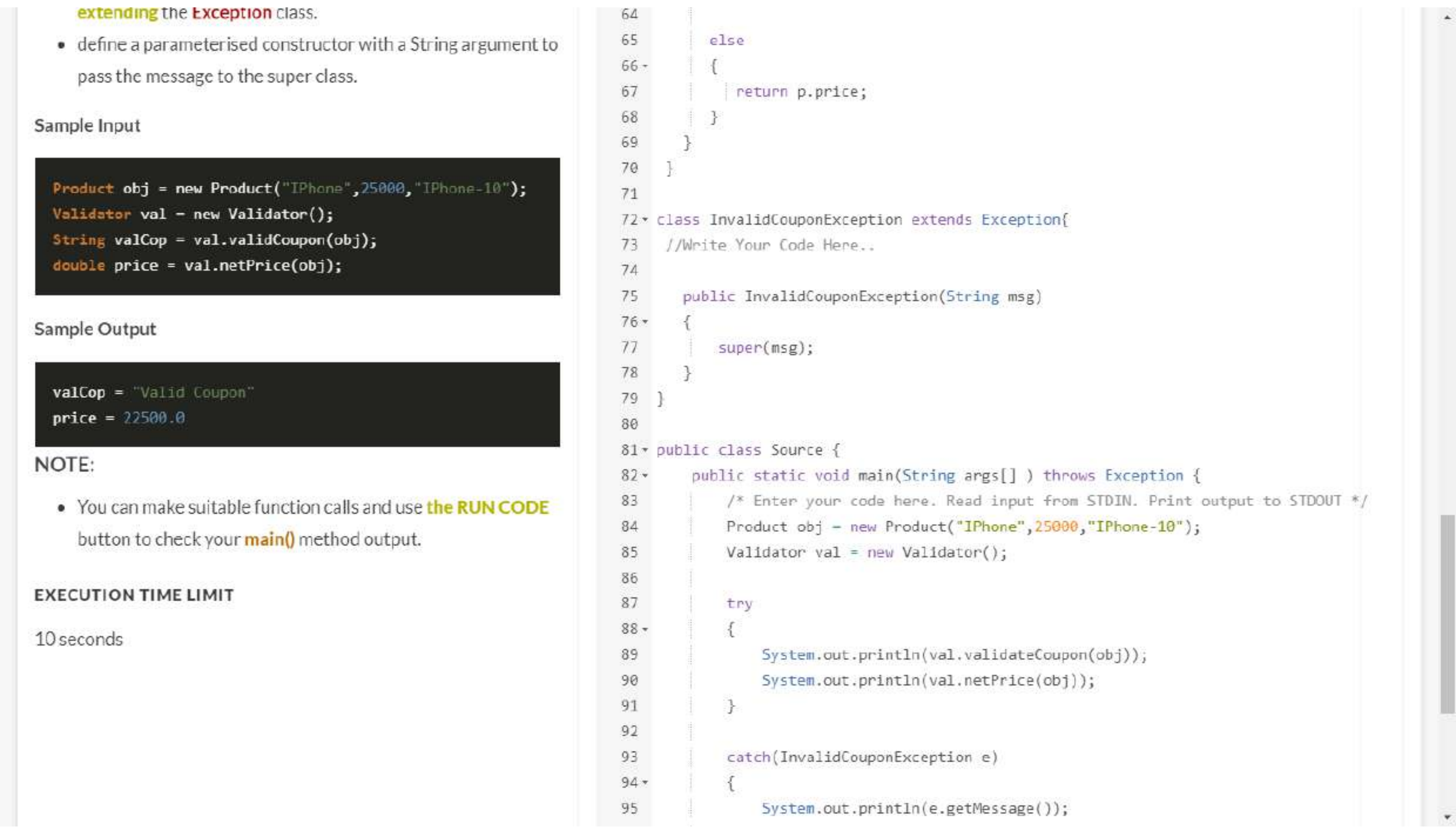
- **netPrice** = totalPrice-discountPrice.
- **return** netPrice if Coupon is valid else **return** totalPrice.

Class InvalidCouponException

- define custom exception class InvalidCouponException by **extending** the **Exception** class.
- define a parameterised constructor with a String argument to pass the message to the super class.

Sample Input

```
   //   Matcher matcher = pattern.matcher(p.coupon);
     String check = p.name;
     check+='-';
     String onlyNumbers = p.coupon.replaceAll("[^0-9]", "");
     check+=onlyNumbers;
     discount = Integer.parseInt(onlyNumbers);
     if(p.coupon.equals(check) && discount>=10 && discount<=25)
      {
          valid=true;
          return "Valid Coupon";
      }

      else
         throw new InvalidCouponException("Invalid Coupon");
  }

  public double netPrice(Product p)
  {
     if(valid==true)
     {
         discountPrice = p.price/100*discount;
         netPrice = p.price-discountPrice;
         return netPrice;

     }

     else
     {
        return p.price;
     }
  }
}
```



extending the Exception class.

- define a parameterised constructor with a String argument to pass the message to the super class.

Sample Input

```
Product obj = new Product("IPhone",25000,"IPhone-10");
Validator val = new Validator();
String valCop = val.validCoupon(obj);
double price = val.netPrice(obj);
```

Sample Output

```
valCop = "Valid Coupon"
price = 22500.0
```

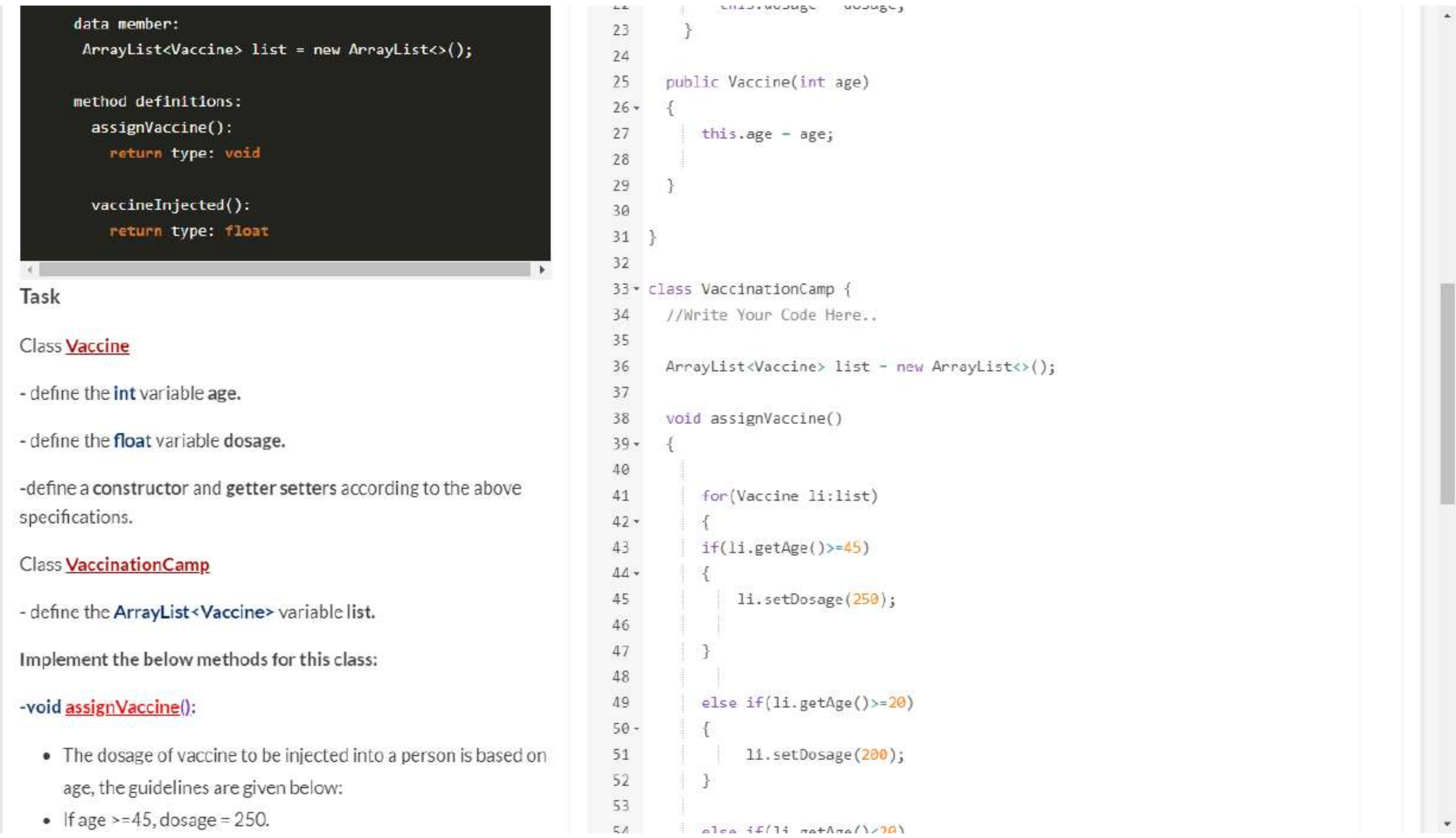
NOTE:

- You can make suitable function calls and use **the RUN CODE** button to check your **main()** method output.

EXECUTION TIME LIMIT

10 seconds

```

        else
        {
            return p.price;
        }
    }
}

class InvalidCouponException extends Exception{
 //Write Your Code Here..

    public InvalidCouponException(String msg)
    {
        super(msg);
    }
}

public class Source {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        Product obj = new Product("IPhone",25000,"IPhone-10");
        Validator val = new Validator();

        try
        {
            System.out.println(val.validateCoupon(obj));
            System.out.println(val.netPrice(obj));
        }

        catch(InvalidCouponException e)
        {
            System.out.println(e.getMessage());
```



```
    data member:
     ArrayList<Vaccine> list = new ArrayList<>();

    method definitions:
      assignVaccine():
        return type: void

      vaccineInjected():
        return type: float
```

Task

Class Vaccine

- define the **int** variable **age.**

- define the **float** variable **dosage.**

-define a **constructor** and **getter setters** according to the above specifications.

Class VaccinationCamp

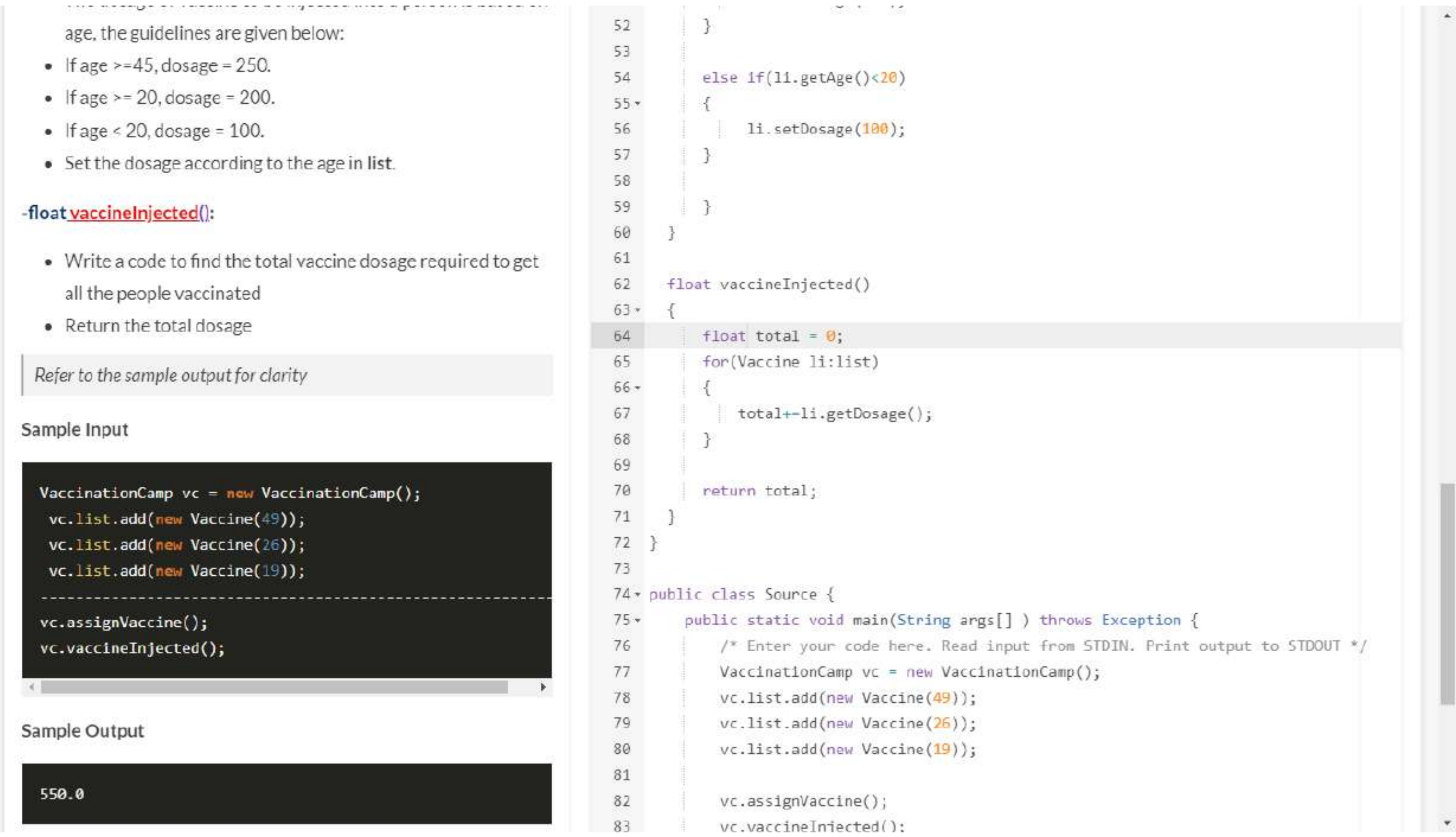
- define the **ArrayList<Vaccine>** variable **list.**

Implement the below methods for this class:

-void assignVaccine():

- The dosage of vaccine to be injected into a person is based on age, the guidelines are given below:
- If age >=45, dosage = 250.

```
        this.dosage = dosage;
    }

  public Vaccine(int age)
  {
      this.age = age;

  }


}

class VaccinationCamp {
  //Write Your Code Here..

  ArrayList<Vaccine> list = new ArrayList<>();

  void assignVaccine()
  {

      for(Vaccine li:list)
      {
      if(li.getAge()>=45)
      {
          li.setDosage(250);

      }

      else if(li.getAge()>=20)
      {
          li.setDosage(200);
      }

      else if(li.getAge()<20)
```



age, the guidelines are given below:

- If age >=45, dosage = 250.
- If age >= 20, dosage = 200.
- If age < 20, dosage = 100.
- Set the dosage according to the age in **list**.

-float vaccineInjected():

- Write a code to find the total vaccine dosage required to get all the people vaccinated
- Return the total dosage

Refer to the sample output for clarity

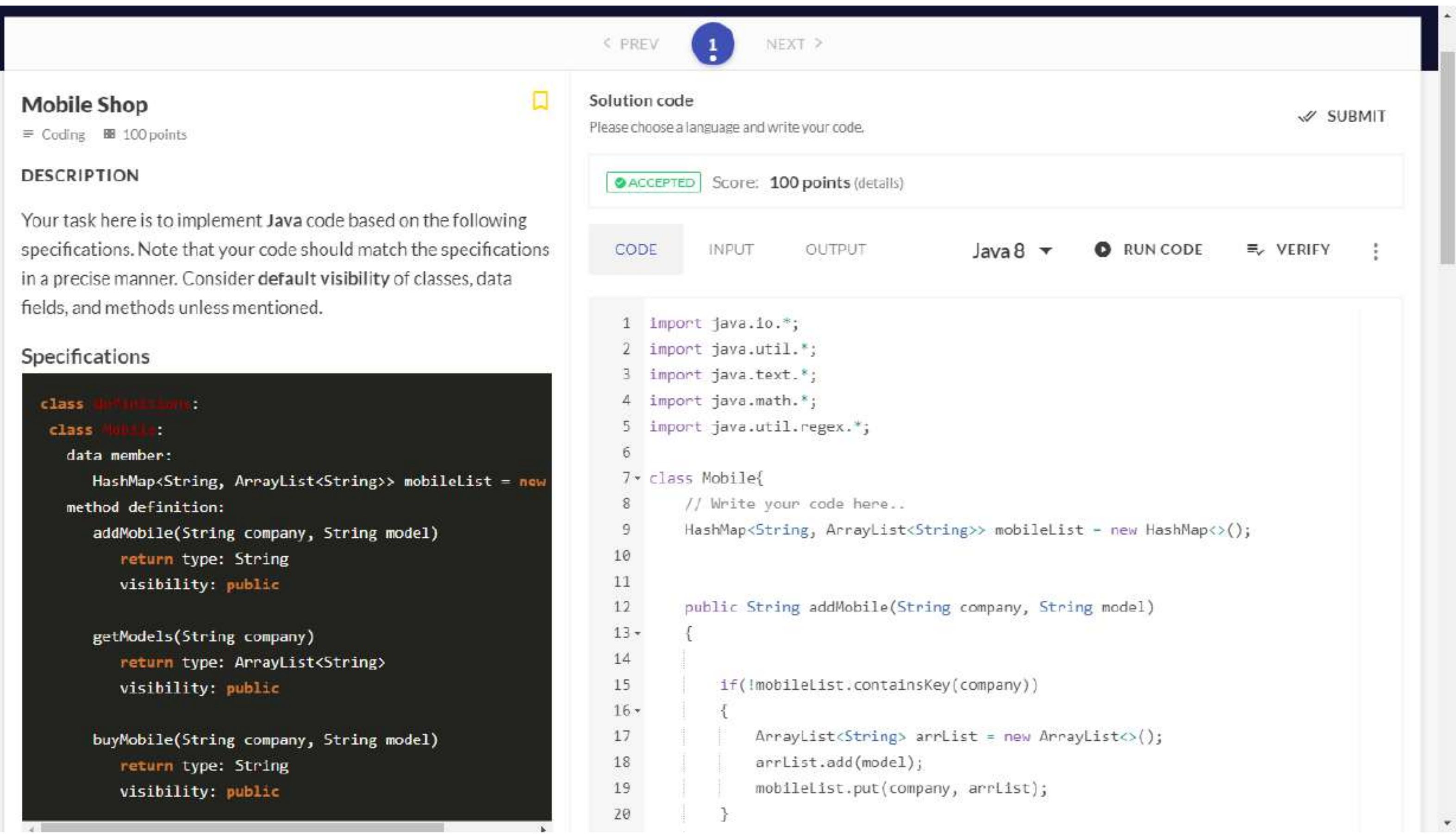
Sample Input

```
VaccinationCamp vc = new VaccinationCamp();
 vc.list.add(new Vaccine(49));
 vc.list.add(new Vaccine(26));
 vc.list.add(new Vaccine(19));
--------------------------------------------------------------------
vc.assignVaccine();
vc.vaccineInjected();
```

Sample Output

```
550.0
```

```java
        }

        else if(li.getAge()<20)
        {
            li.setDosage(100);
        }

        }
    }

    float vaccineInjected()
    {
        float total = 0;
        for(Vaccine li:list)
        {
            total+=li.getDosage();
        }

        return total;
    }
}

public class Source {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        VaccinationCamp vc = new VaccinationCamp();
        vc.list.add(new Vaccine(49));
        vc.list.add(new Vaccine(26));
        vc.list.add(new Vaccine(19));

        vc.assignVaccine();
        vc.vaccineInjected();
```



Mobile Shop

Coding   100 points

DESCRIPTION

Your task here is to implement **Java** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider **default visibility** of classes, data fields, and methods unless mentioned.

Specifications

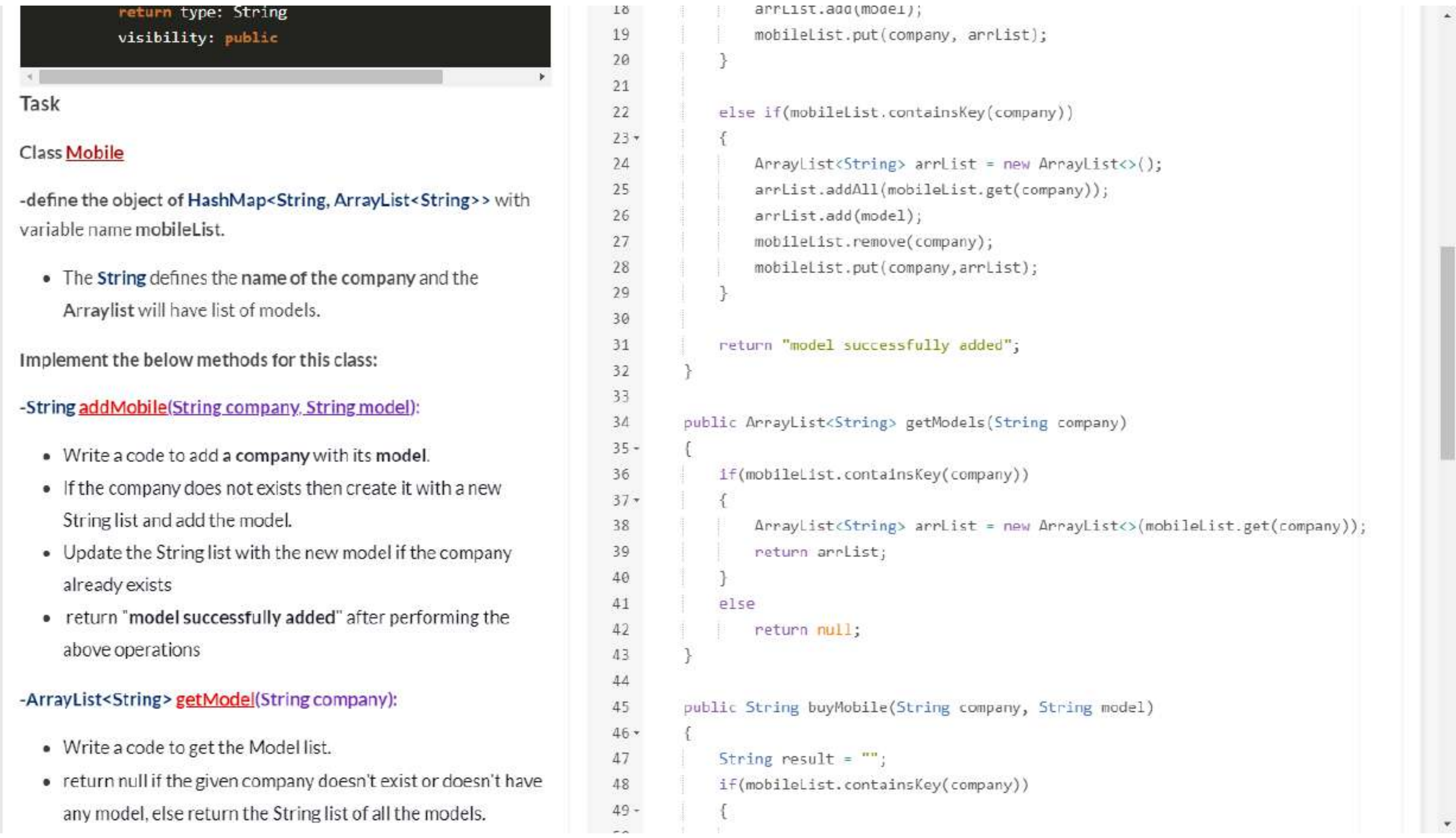
```
class definitions:
 class Mobile:
   data member:
      HashMap<String, ArrayList<String>> mobileList = new
   method definition:
      addMobile(String company, String model)
         return type: String
         visibility: public

      getModels(String company)
         return type: ArrayList<String>
         visibility: public

      buyMobile(String company, String model)
         return type: String
         visibility: public
```

Solution code

Please choose a language and write your code.

Score: **100 points** (details)

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class Mobile{
    // Write your code here..
    HashMap<String, ArrayList<String>> mobileList = new HashMap<>();


    public String addMobile(String company, String model)
    {

        if(!mobileList.containsKey(company))
        {
            ArrayList<String> arrList = new ArrayList<>();
            arrList.add(model);
            mobileList.put(company, arrList);
        }
```



```
return type: String
visibility: public
```

Task

Class Mobile

-define the object of HashMap<String, ArrayList<String>> with variable name **mobileList**.

- The **String** defines the **name of the company** and the **Arraylist** will have list of models.

Implement the below methods for this class:

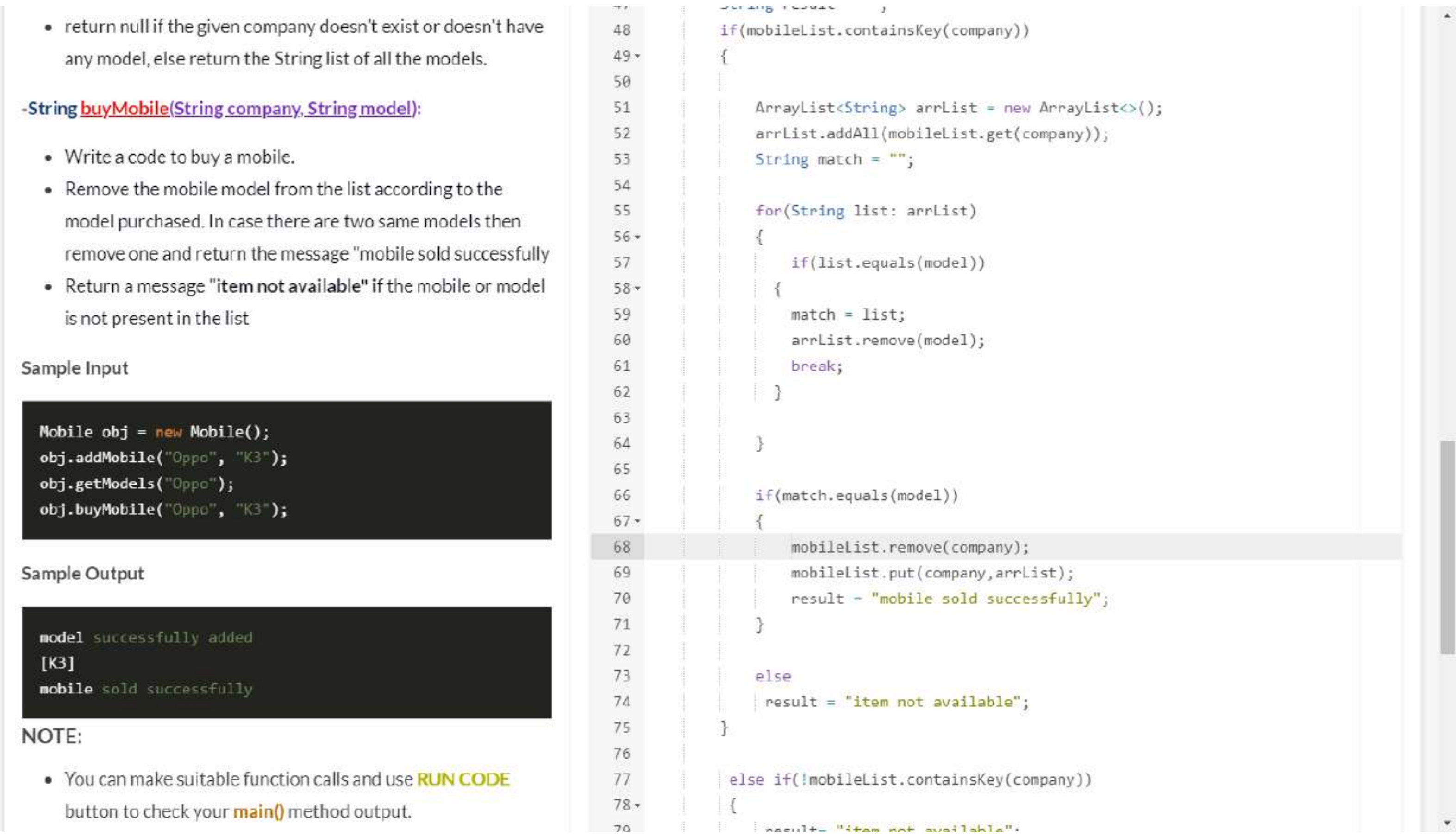
-String addMobile(String company, String model):

- Write a code to add **a company** with its **model**.
- If the company does not exists then create it with a new String list and add the model.
- Update the String list with the new model if the company already exists
- return "**model successfully added**" after performing the above operations

-ArrayList<String> getModel(String company):

- Write a code to get the Model list.
- return null if the given company doesn't exist or doesn't have any model, else return the String list of all the models.

```
            arrList.add(model);
            mobileList.put(company, arrList);
        }

        else if(mobileList.containsKey(company))
        {
            ArrayList<String> arrList = new ArrayList<>();
            arrList.addAll(mobileList.get(company));
            arrList.add(model);
            mobileList.remove(company);
            mobileList.put(company,arrList);
        }

        return "model successfully added";
    }

    public ArrayList<String> getModels(String company)
    {
        if(mobileList.containsKey(company))
        {
            ArrayList<String> arrList = new ArrayList<>(mobileList.get(company));
            return arrList;
        }
        else
            return null;
    }

    public String buyMobile(String company, String model)
    {
        String result = "";
        if(mobileList.containsKey(company))
        {
```



- return null if the given company doesn't exist or doesn't have any model, else return the String list of all the models.

-String buyMobile(String company, String model):

- Write a code to buy a mobile.
- Remove the mobile model from the list according to the model purchased. In case there are two same models then remove one and return the message "mobile sold successfully
- Return a message "**item not available**" if the mobile or model is not present in the list

Sample Input

```
Mobile obj = new Mobile();
obj.addMobile("Oppo", "K3");
obj.getModels("Oppo");
obj.buyMobile("Oppo", "K3");
```

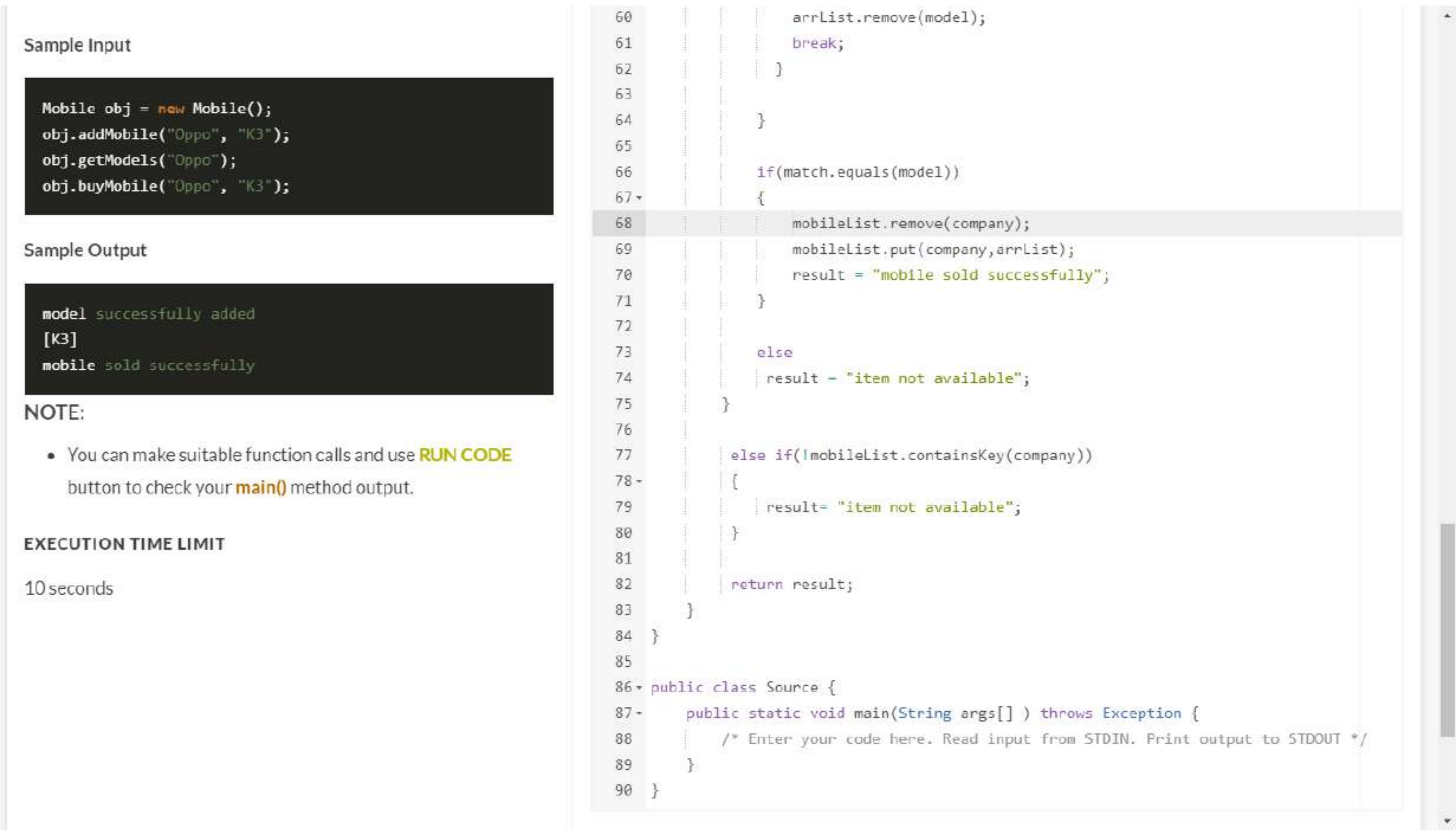
Sample Output

```
model successfully added
[K3]
mobile sold successfully
```

NOTE:

- You can make suitable function calls and use RUN CODE button to check your main() method output.

```
if(mobileList.containsKey(company))
{

    ArrayList<String> arrList = new ArrayList<>();
    arrList.addAll(mobileList.get(company));
    String match = "";

    for(String list: arrList)
    {
        if(list.equals(model))
      {
        match = list;
        arrList.remove(model);
        break;
      }

    }

    if(match.equals(model))
    {
        mobileList.remove(company);
        mobileList.put(company,arrList);
        result = "mobile sold successfully";
    }

    else
     result = "item not available";
}

 else if(!mobileList.containsKey(company))
 {
     result= "item not available";
```



Sample Input

```
Mobile obj = new Mobile();
obj.addMobile("Oppo", "K3");
obj.getModels("Oppo");
obj.buyMobile("Oppo", "K3");
```

Sample Output

```
model successfully added
[K3]
mobile sold successfully
```

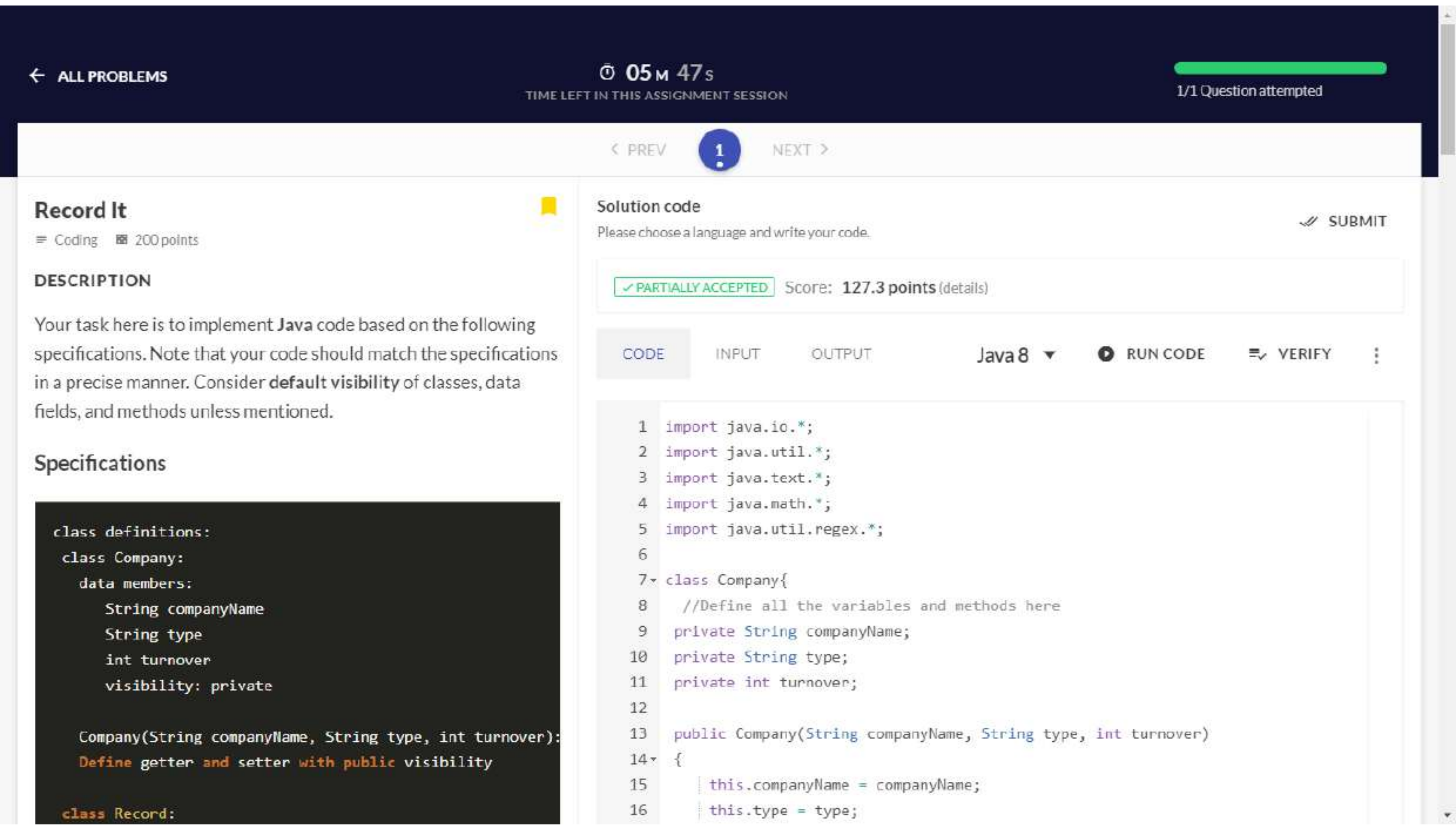
NOTE:

- You can make suitable function calls and use **RUN CODE** button to check your **main()** method output.

EXECUTION TIME LIMIT

10 seconds

```
                arrList.remove(model);
                break;
             }

            }

            if(match.equals(model))
            {
                mobileList.remove(company);
                mobileList.put(company,arrList);
                result = "mobile sold successfully";
            }

            else
             result = "item not available";
        }

        else if(!mobileList.containsKey(company))
        {
             result= "item not available";
        }

        return result;
    }
}

public class Source {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    }
}
```



Record It

Coding 200 points

DESCRIPTION

Your task here is to implement **Java** code based on the following specifications. Note that your code should match the specifications in a precise manner. Consider **default visibility** of classes, data fields, and methods unless mentioned.

Specifications

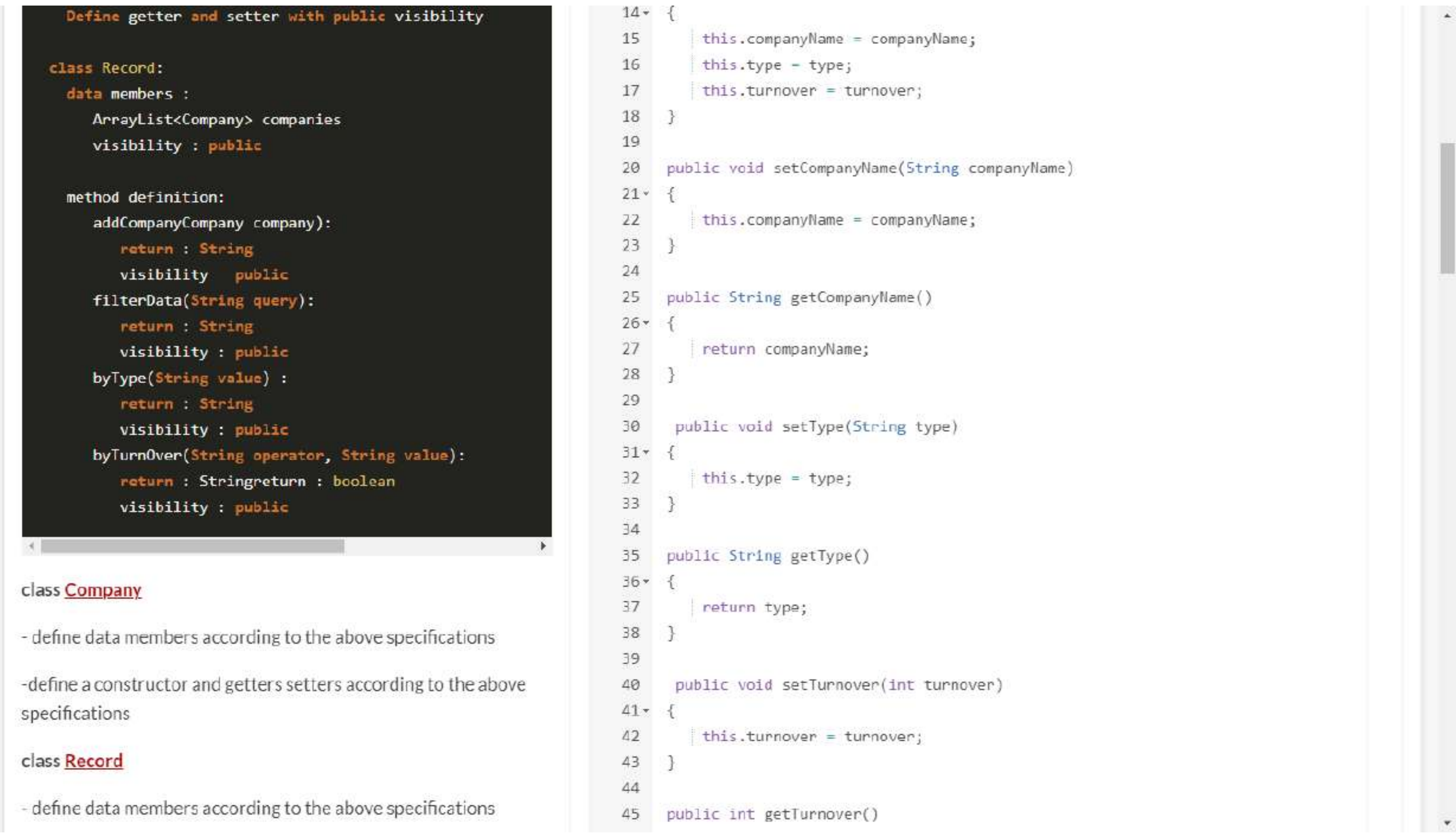
```
class definitions:
 class Company:
   data members:
      String companyName
      String type
      int turnover
      visibility: private

   Company(String companyName, String type, int turnover):
   Define getter and setter with public visibility

 class Record:
```

Solution code

Please choose a language and write your code.

Score: 127.3 points (details)

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class Company{
  //Define all the variables and methods here
 private String companyName;
 private String type;
 private int turnover;

 public Company(String companyName, String type, int turnover)
 {
     this.companyName = companyName;
     this.type = type;
```



```
  Define getter and setter with public visibility

class Record:
  data members :
     ArrayList<Company> companies
     visibility : public

  method definition:
     addCompanyCompany company):
        return : String
        visibility   public
     filterData(String query):
        return : String
        visibility : public
     byType(String value) :
        return : String
        visibility : public
     byTurnOver(String operator, String value):
        return : Stringreturn : boolean
        visibility : public
```

class Company

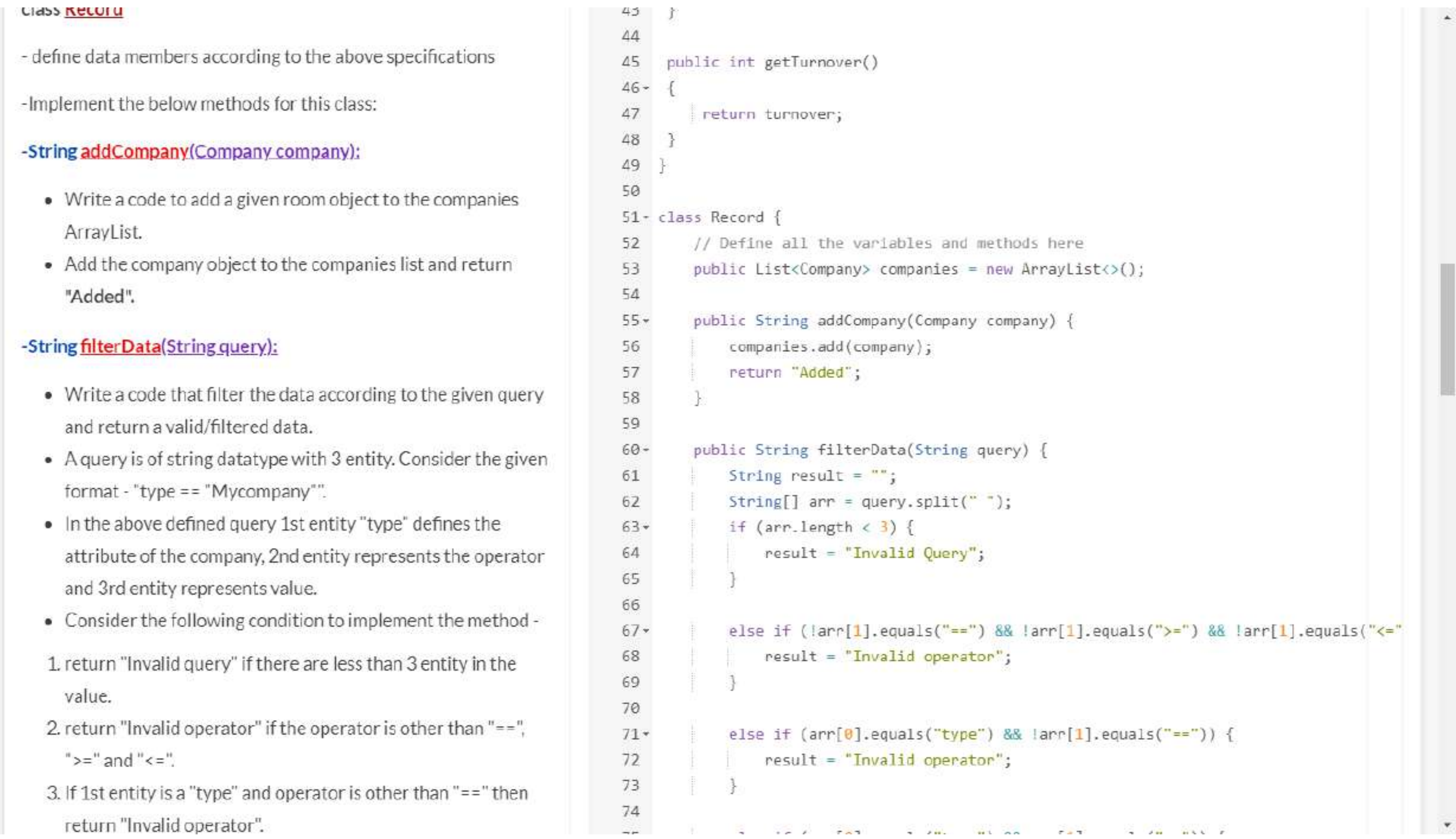
- define data members according to the above specifications

-define a constructor and getters setters according to the above specifications

class Record

- define data members according to the above specifications

```
{
    this.companyName = companyName;
    this.type = type;
    this.turnover = turnover;
}

public void setCompanyName(String companyName)
{
    this.companyName = companyName;
}

public String getCompanyName()
{
    return companyName;
}

 public void setType(String type)
{
    this.type = type;
}

public String getType()
{
    return type;
}

 public void setTurnover(int turnover)
{
    this.turnover = turnover;
}

public int getTurnover()
```



class Record

- define data members according to the above specifications

-Implement the below methods for this class:

-String addCompany(Company company):

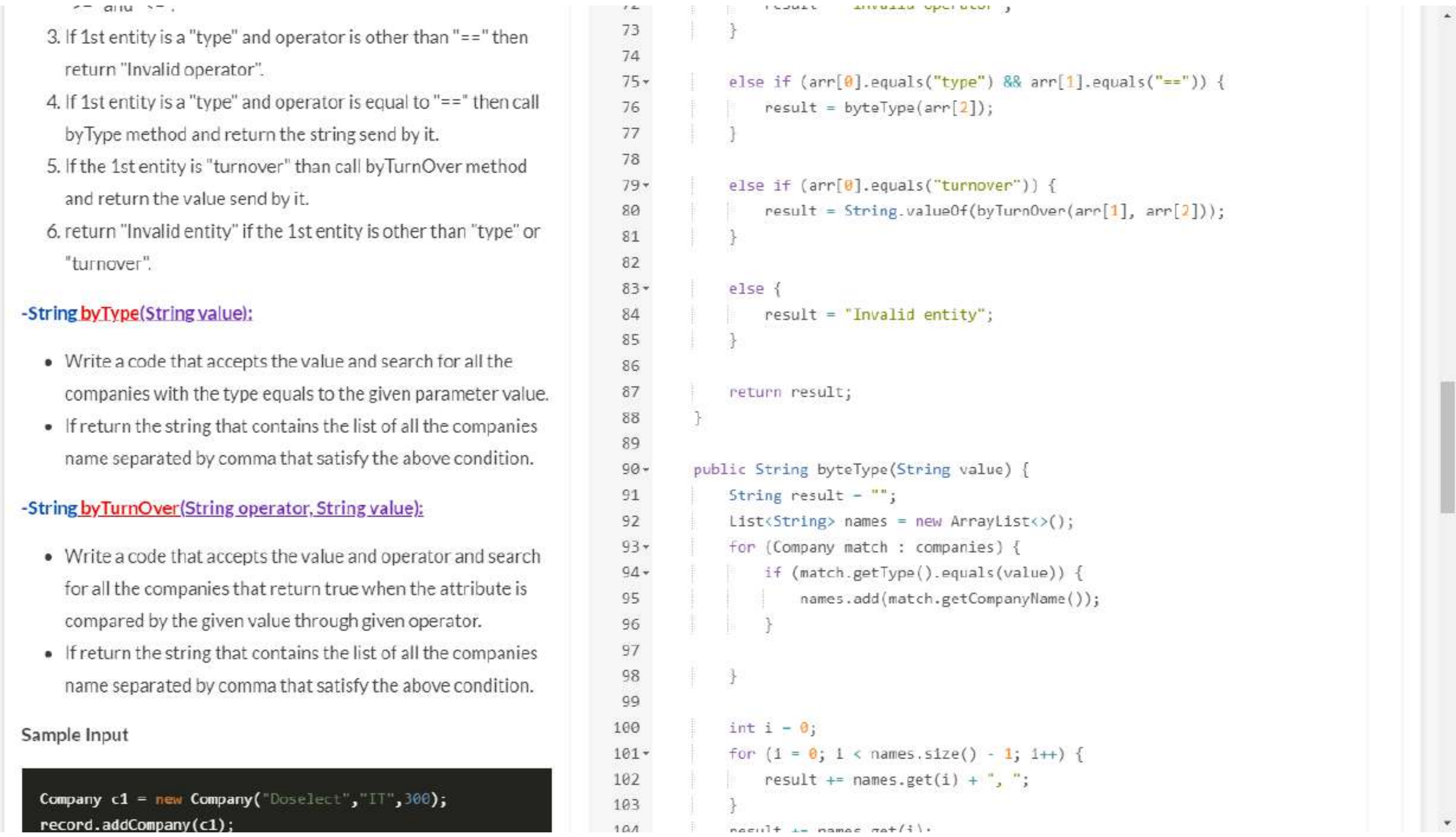
- Write a code to add a given room object to the companies ArrayList.
- Add the company object to the companies list and return **"Added".**

-String filterData(String query):

- Write a code that filter the data according to the given query and return a valid/filtered data.
- A query is of string datatype with 3 entity. Consider the given format - "type == "Mycompany"".
- In the above defined query 1st entity "type" defines the attribute of the company, 2nd entity represents the operator and 3rd entity represents value.
- Consider the following condition to implement the method -

1. return "Invalid query" if there are less than 3 entity in the value.
2. return "Invalid operator" if the operator is other than "==", ">=" and "<=".
3. If 1st entity is a "type" and operator is other than "==" then return "Invalid operator".

```java
 }

 public int getTurnover()
 {
     return turnover;
 }
}

class Record {
    // Define all the variables and methods here
    public List<Company> companies = new ArrayList<>();

    public String addCompany(Company company) {
        companies.add(company);
        return "Added";
    }

    public String filterData(String query) {
        String result = "";
        String[] arr = query.split(" ");
        if (arr.length < 3) {
            result = "Invalid Query";
        }

        else if (!arr[1].equals("==") && !arr[1].equals(">=") && !arr[1].equals("<="
            result = "Invalid operator";
        }

        else if (arr[0].equals("type") && !arr[1].equals("==")) {
            result = "Invalid operator";
        }

```



">=" and "<=".

3. If 1st entity is a "type" and operator is other than "==" then return "Invalid operator".
4. If 1st entity is a "type" and operator is equal to "==" then call byType method and return the string send by it.
5. If the 1st entity is "turnover" than call byTurnOver method and return the value send by it.
6. return "Invalid entity" if the 1st entity is other than "type" or "turnover".

-String byType(String value):

- Write a code that accepts the value and search for all the companies with the type equals to the given parameter value.
- If return the string that contains the list of all the companies name separated by comma that satisfy the above condition.

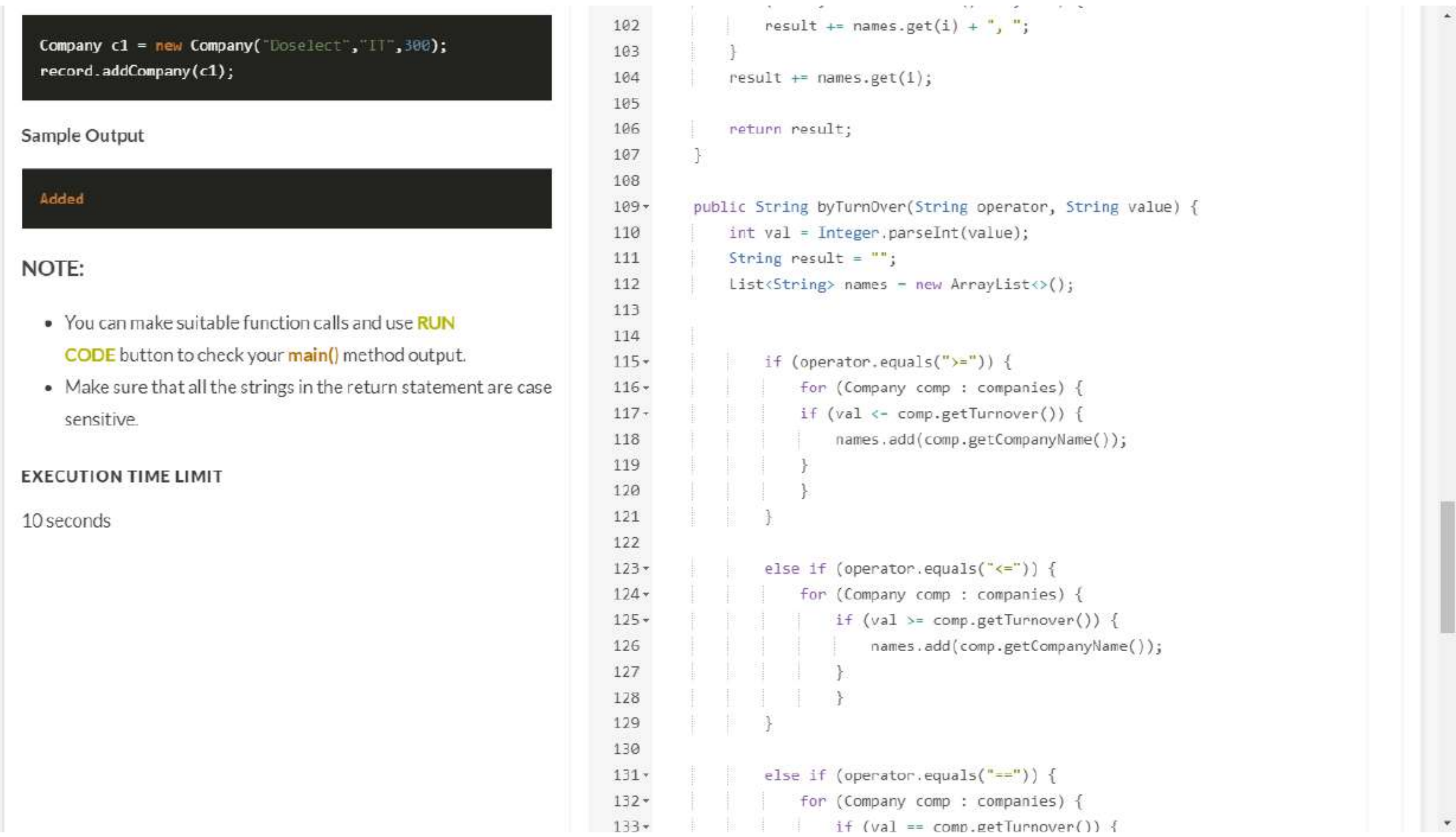
-String byTurnOver(String operator, String value):

- Write a code that accepts the value and operator and search for all the companies that return true when the attribute is compared by the given value through given operator.
- If return the string that contains the list of all the companies name separated by comma that satisfy the above condition.

Sample Input

```
Company c1 = new Company("Doselect","IT",300);
record.addCompany(c1);
```

```
            result = "Invalid operator";
        }

        else if (arr[0].equals("type") && arr[1].equals("==")) {
            result = byteType(arr[2]);
        }

        else if (arr[0].equals("turnover")) {
            result = String.valueOf(byTurnOver(arr[1], arr[2]));
        }

        else {
            result = "Invalid entity";
        }

        return result;
    }

    public String byteType(String value) {
        String result = "";
        List<String> names = new ArrayList<>();
        for (Company match : companies) {
            if (match.getType().equals(value)) {
                names.add(match.getCompanyName());
            }

        }

        int i = 0;
        for (i = 0; i < names.size() - 1; i++) {
            result += names.get(i) + ", ";
        }
        result += names.get(i);
```



```
Company c1 = new Company("Doselect","IT",300);
record.addCompany(c1);
```

Sample Output

```
Added
```

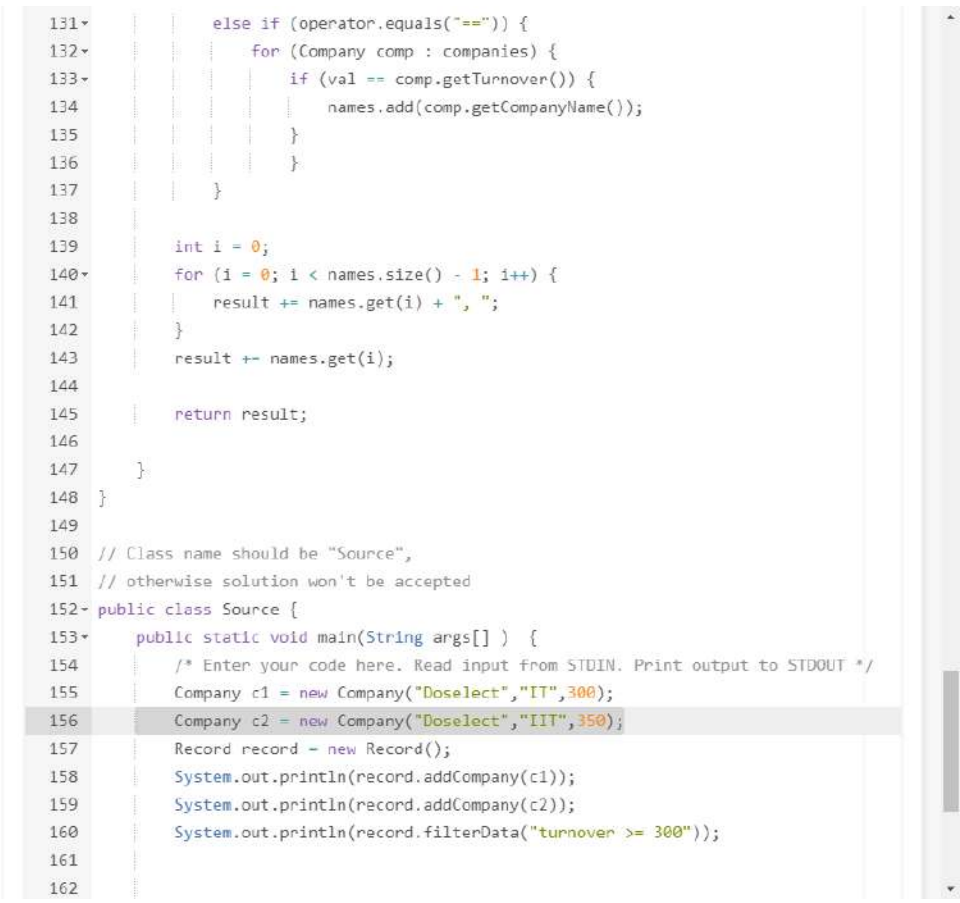
NOTE:

- You can make suitable function calls and use **RUN CODE** button to check your **main()** method output.
- Make sure that all the strings in the return statement are case sensitive.

EXECUTION TIME LIMIT

10 seconds

```
            result += names.get(i) + ", ";
        }
        result += names.get(1);

        return result;
    }

    public String byTurnOver(String operator, String value) {
        int val = Integer.parseInt(value);
        String result = "";
        List<String> names = new ArrayList<>();


            if (operator.equals(">=")) {
                for (Company comp : companies) {
                if (val <= comp.getTurnover()) {
                    names.add(comp.getCompanyName());
                }
                }
            }

            else if (operator.equals("<=")) {
                for (Company comp : companies) {
                    if (val >= comp.getTurnover()) {
                        names.add(comp.getCompanyName());
                    }
                    }
            }

            else if (operator.equals("==")) {
                for (Company comp : companies) {
                    if (val == comp.getTurnover()) {
```



```java
            else if (operator.equals("==")) {
                for (Company comp : companies) {
                    if (val == comp.getTurnover()) {
                        names.add(comp.getCompanyName());
                    }
                    }
            }

        int i = 0;
        for (i = 0; i < names.size() - 1; i++) {
            result += names.get(i) + ", ";
        }
        result += names.get(i);

        return result;

    }
}

// Class name should be "Source",
// otherwise solution won't be accepted
public class Source {
    public static void main(String args[] )  {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        Company c1 = new Company("Doselect","IT",300);
        Company c2 = new Company("Doselect","IIT",350);
        Record record = new Record();
        System.out.println(record.addCompany(c1));
        System.out.println(record.addCompany(c2));
        System.out.println(record.filterData("turnover >= 300"));


```