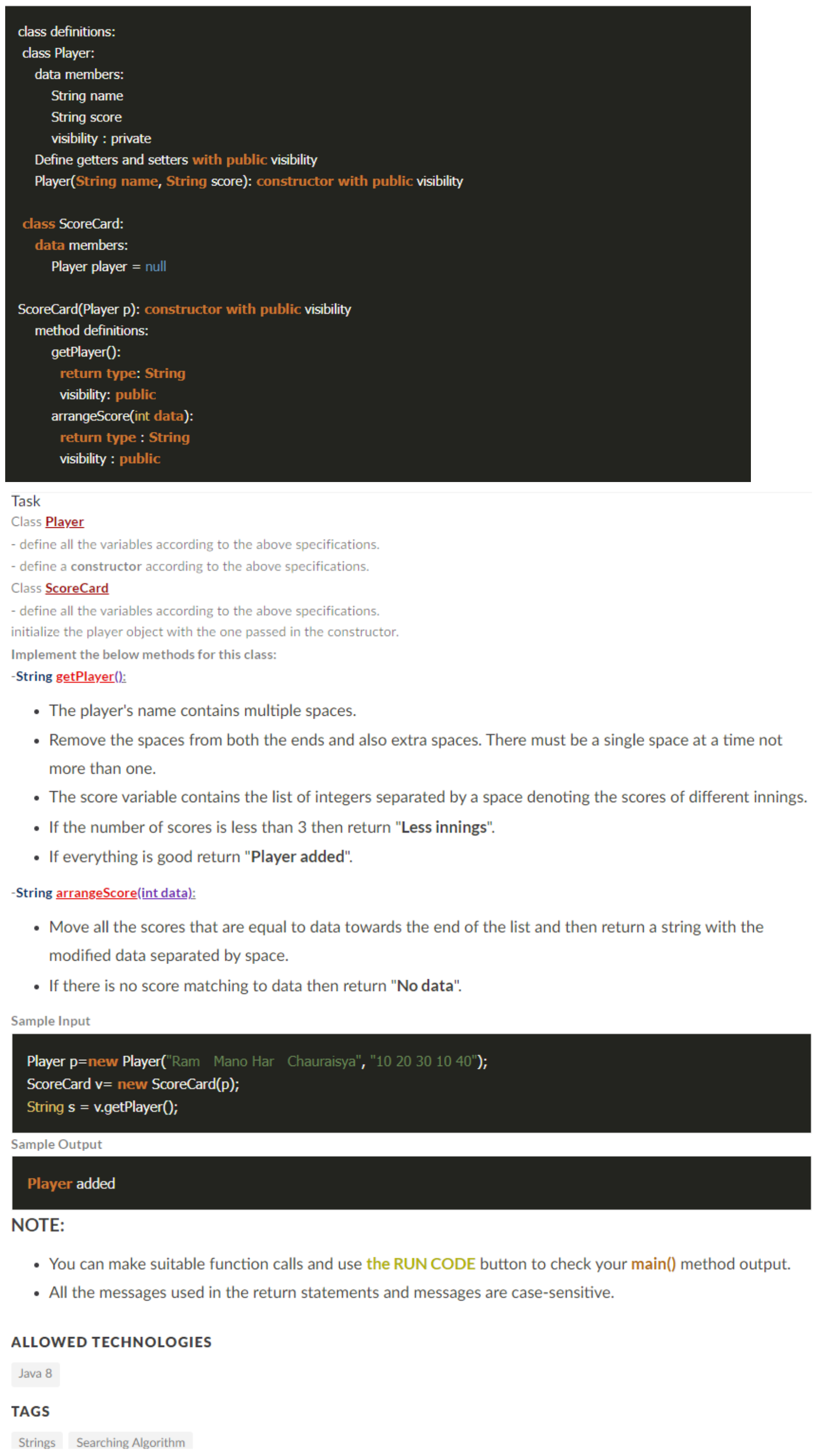
Do select Java questions

Topic: String Manipulation

Question Title: The Player Setting

Description

Complete the classes using the Specifications given below. Consider default visibility of classes, data fields, and methods unless mentioned otherwise.



```
class definitions:
 class Player:
    data members:
        String name
        String score
        visibility : private
    Define getters and setters with public visibility
    Player(String name, String score): constructor with public visibility

 class ScoreCard:
    data members:
        Player player = null

ScoreCard(Player p): constructor with public visibility
    method definitions:
        getPlayer():
          return type: String
          visibility: public
        arrangeScore(int data):
          return type : String
          visibility : public
```

Task

Class Player

- define all the variables according to the above specifications.
- define a **constructor** according to the above specifications.

Class ScoreCard

- define all the variables according to the above specifications.

initialize the player object with the one passed in the constructor.

Implement the below methods for this class:

-**String getPlayer():**

- The player's name contains multiple spaces.
- Remove the spaces from both the ends and also extra spaces. There must be a single space at a time not more than one.
- The score variable contains the list of integers separated by a space denoting the scores of different innings.
- If the number of scores is less than 3 then return "**Less innings**".
- If everything is good return "**Player added**".

-**String arrangeScore(int data):**

- Move all the scores that are equal to data towards the end of the list and then return a string with the modified data separated by space.
- If there is no score matching to data then return "**No data**".

Sample Input

```
Player p=new Player("Ram   Mano Har   Chauraisya", "10 20 30 10 40");
ScoreCard v= new ScoreCard(p);
String s = v.getPlayer();
```

Sample Output

```
Player added
```

NOTE:

- You can make suitable function calls and use the RUN CODE button to check your main() method output.
- All the messages used in the return statements and messages are case-sensitive.

ALLOWED TECHNOLOGIES

Java 8

TAGS

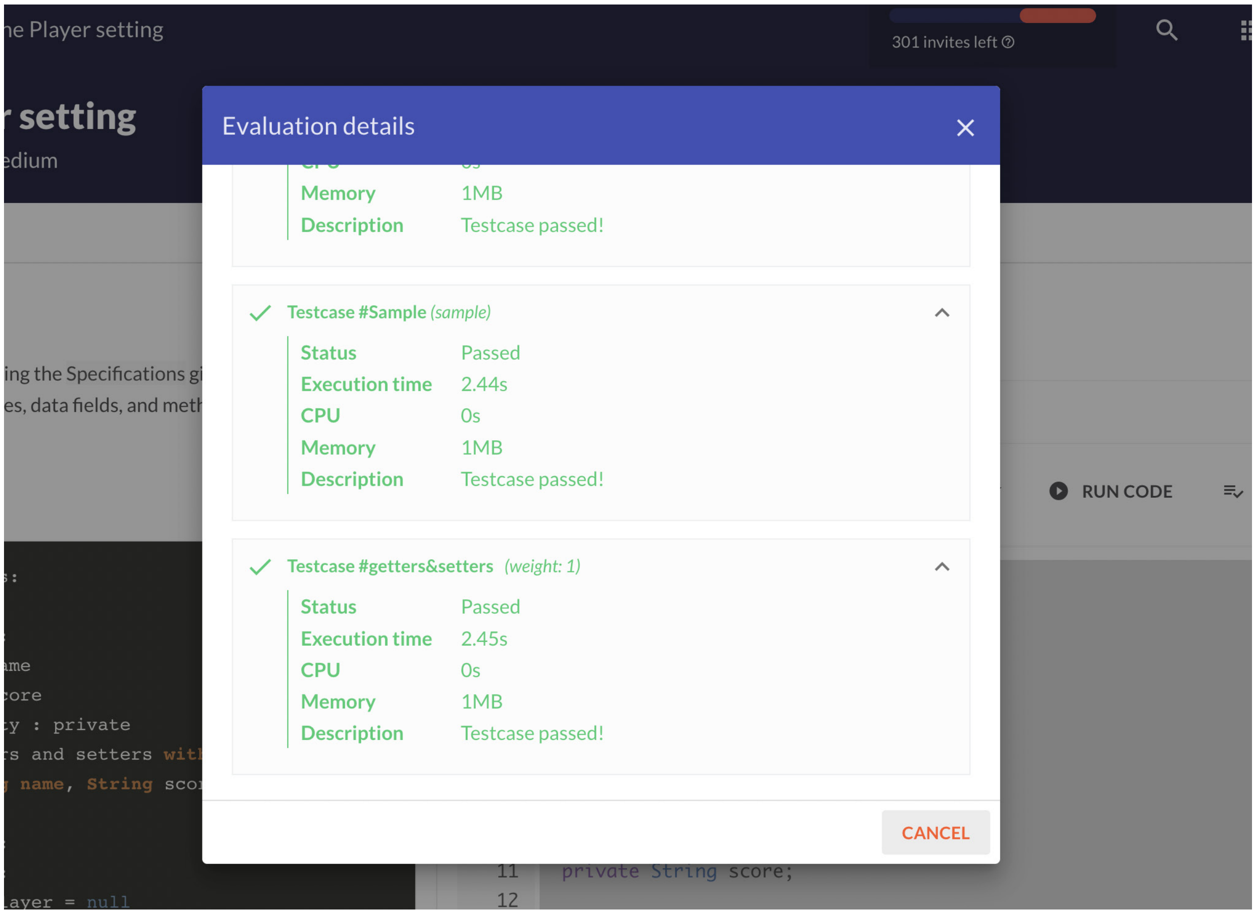
Strings Searching Algorithm

Solution:

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

class Player{
  //Code here..

  private String name;
  private String score;

  public Player(String name, String score){
      this.name=name;
      this.score=score;
  }

  public void setName(String name){
        this.name=name;
  }

  public void setScore(String score){
      this.score=score;
  }

  public String getName(){
      return this.name;
  }

  public String getScore(){
      return this.score;
  }
}

class ScoreCard{
```

```
//Code here..

Player player=null;

public ScoreCard(Player p){
    player = p;
}

public String getPlayer(){

    String name = player.getName().trim();
    String[] names = name.split("\\s+");
    name="";
    for(int i=0;i<names.length;i++)
      name+=(names[i].trim()+" ");
    name = name.trim();

    String score = player.getScore().trim();
    String[] scores = score.split("\\s+");
    if(scores.length<3)
      return "Less innings";
    else
      return "Player added";
}

public String arrangeScore(int data){

    String score = player.getScore().trim();
    String[] scores = score.split("\\s+");

    boolean matching=false;
    for(int i=0;i<scores.length;i++){
        if(Integer.parseInt(scores[i].trim())==data){
            matching=true;
            break;
        }
    }

    if(!matching)
      return "No data";
    else{
        score="";

        for(int i=0;i<scores.length;i++){
            if(Integer.parseInt(scores[i].trim())!=data)
              score+=(scores[i].trim()+" ");
        }
        for(int i=0;i<scores.length;i++){
            if(Integer.parseInt(scores[i].trim())==data)
              score+=(scores[i].trim()+" ");
        }

        score=score.trim();
        return score;
    }

  }
}

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

                        Scanner input = new Scanner(System.in);

                        Player p=new Player("Ram   Mano Har   Chauraisya", "10 20 30 10 40");
        ScoreCard v= new ScoreCard(p);
        String s = v.getPlayer();

        System.out.println(s);
            }
}
```

Verification:


Evaluation details

Memory	1MB
Description	Testcase passed!

Testcase #Sample *(sample)*

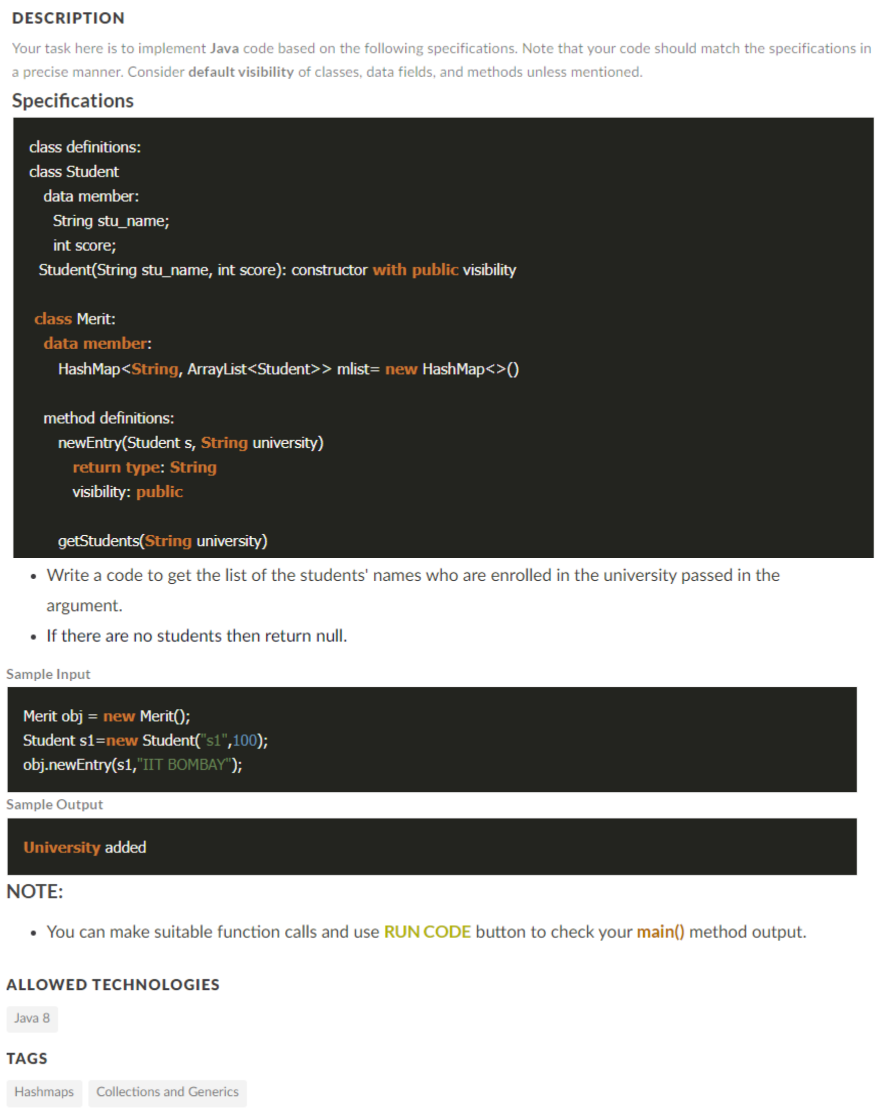
Status	Passed
Execution time	2.44s
CPU	0s
Memory	1MB
Description	Testcase passed!

Testcase #getters&setters *(weight: 1)*

Status	Passed
Execution time	2.45s
CPU	0s
Memory	1MB
Description	Testcase passed!

Topic: Collections

Question Title: The Merit List



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 Student
  data member:
    String stu_name;
    int score;
 Student(String stu_name, int score): constructor with public visibility

 class Merit:
  data member:
    HashMap<String, ArrayList<Student>> mlist= new HashMap<>()

  method definitions:
    newEntry(Student s, String university)
      return type: String
      visibility: public

    getStudents(String university)
```

- Write a code to get the list of the students' names who are enrolled in the university passed in the argument.
- If there are no students then return null.

Sample Input

```
Merit obj = new Merit();
Student s1=new Student("s1",100);
obj.newEntry(s1,"IIT BOMBAY");
```

Sample Output

```
University added
```

NOTE:

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

ALLOWED TECHNOLOGIES

Java 8

TAGS

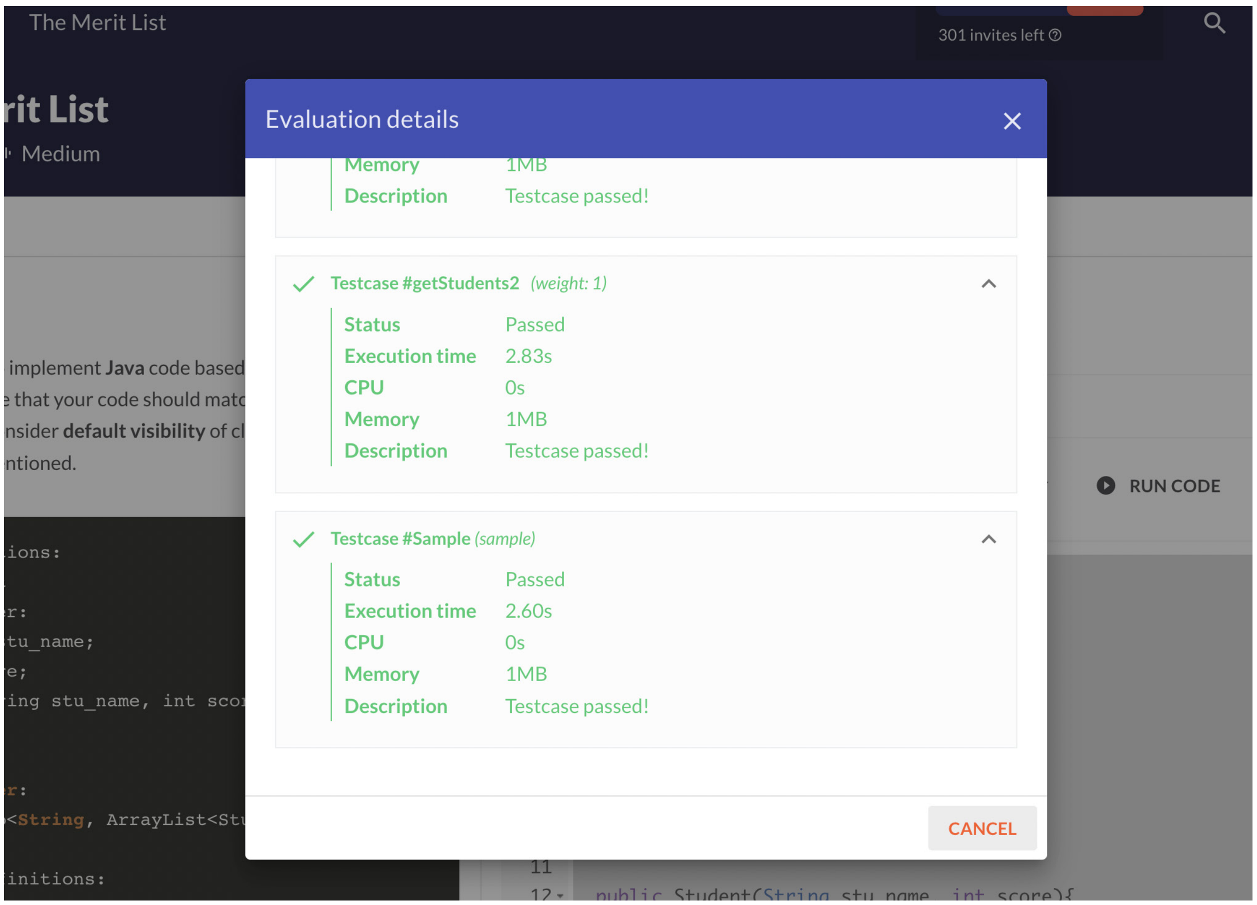
Hashmaps Collections and Generics

Solution:

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

class Student{
  //Your Code Goes Here..
  String stu_name;
  int score;

  public Student(String stu_name, int score){
     this.stu_name = stu_name;
     this.score = score;
  }
}

class Merit{
//Your Code GOes Here..
   HashMap<String, ArrayList<Student>> mlist = new HashMap<>();


   public String newEntry(Student s, String university){

      if(mlist.containsKey(university)){
         ArrayList<Student> list = mlist.get(university);
         list.add(s);
         mlist.put(university, list);
         return "Student added";
      }
      else{
         ArrayList<Student> list = new ArrayList<>();
         list.add(s);
         mlist.put(university, list);
         return "University added";
      }
   }

   public ArrayList<String> getStudents(String university){

      ArrayList<String> list = new ArrayList<>();
      if(mlist.get(university)!=null){
         for(int i=0;i<mlist.get(university).size();i++){
            list.add(mlist.get(university).get(i).stu_name);
         }
         return (list.size()!=0)?list:null;
      }
      else{
         return null;
      }
```

```
    }
}

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



            Merit obj = new Merit();
Student s1=new Student("s1",100);
Student s2=new Student("s2",100);
Student s3=new Student("s3",100);
System.out.println(obj.newEntry(s1,"IIT BOMBAY"));
System.out.println(obj.newEntry(s2,"IIT BOMBAY"));
System.out.println(obj.newEntry(s3,"IIT DELHI"));
System.out.println(obj.getStudents("IIT MUMBAI"));
        }
}
```

Verification:



The Merit List

301 invites left

rit List

Medium

implement **Java** code based
e that your code should matc
nsider **default visibility** of cl
ntioned.

```
ions:
r:
tu_name;
e;
ing stu_name, int scor
r:
<String, ArrayList<Stu
initions:
```

RUN CODE

```
public Student(String stu_name, int score){
```

Evaluation details

Memory	1MB
Description	Testcase passed!

✓ Testcase #getStudents2 *(weight: 1)*

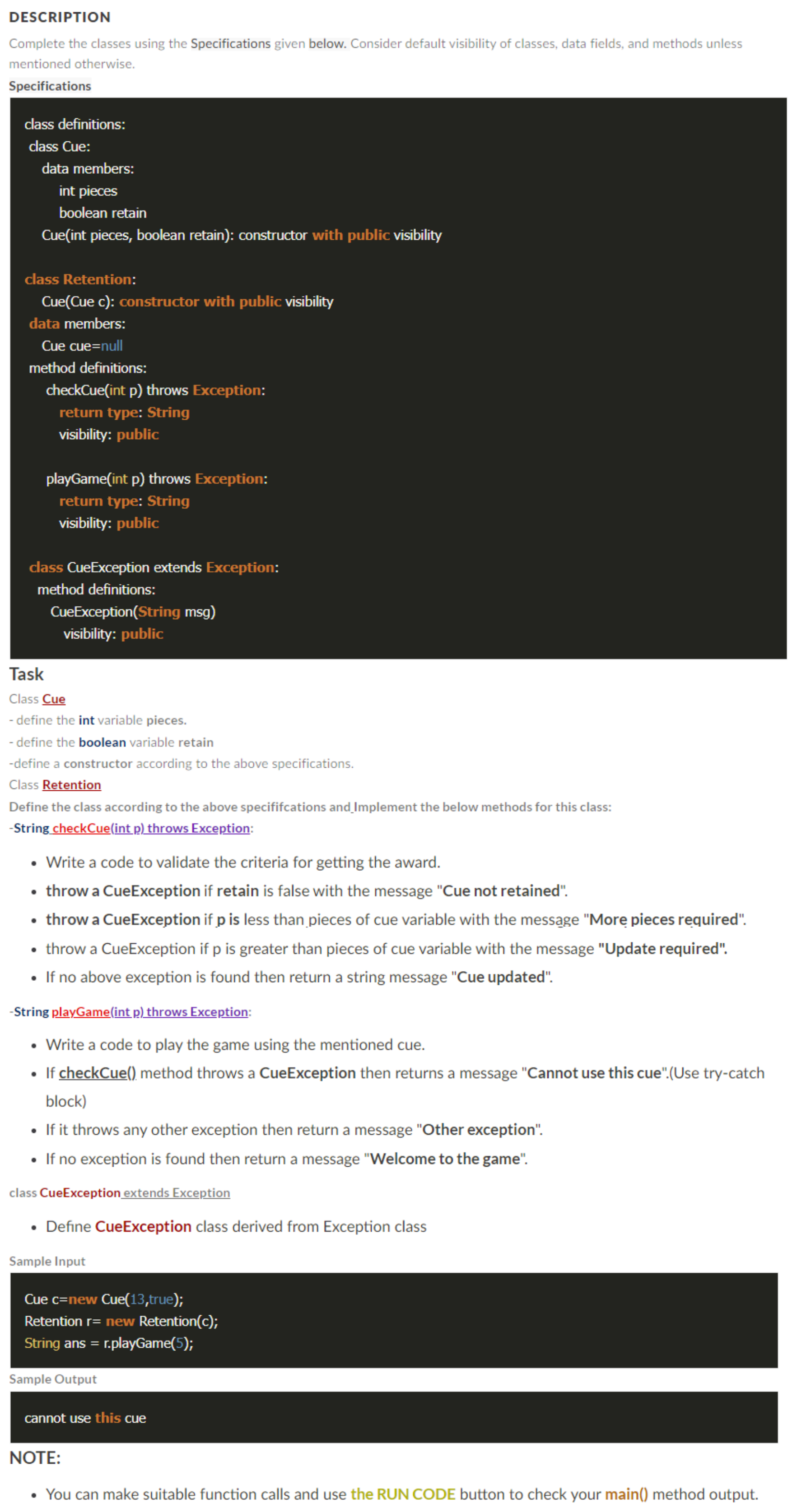
Status	Passed
Execution time	2.83s
CPU	0s
Memory	1MB
Description	Testcase passed!

✓ Testcase #Sample *(sample)*

Status	Passed
Execution time	2.60s
CPU	0s
Memory	1MB
Description	Testcase passed!

CANCEL

Topic: Exception Handling
Question Title: The new cue



DESCRIPTION

Complete the classes using the **Specifications** given **below.** Consider default visibility of classes, data fields, and methods unless mentioned otherwise.

Specifications

```
class definitions:
 class Cue:
   data members:
      int pieces
      boolean retain
   Cue(int pieces, boolean retain): constructor with public visibility

class Retention:
   Cue(Cue c): constructor with public visibility
 data members:
   Cue cue=null
 method definitions:
    checkCue(int p) throws Exception:
      return type: String
      visibility: public

    playGame(int p) throws Exception:
      return type: String
      visibility: public

 class CueException extends Exception:
   method definitions:
     CueException(String msg)
       visibility: public
```

Task

Class Cue
- define the **int** variable **pieces.**
- define the **boolean** variable **retain**
-define a **constructor** according to the above specifications.

Class Retention
Define the class according to the above specififcations and Implement the below methods for this class:
-**String checkCue(int p) throws Exception:**

- Write a code to validate the criteria for getting the award.
- **throw a CueException** if **retain** is false with the message "**Cue not retained**".
- **throw a CueException** if **p is** less than pieces of cue variable with the message "**More pieces required**".
- throw a CueException if p is greater than pieces of cue variable with the message **"Update required".**
- If no above exception is found then return a string message "**Cue updated**".

-**String playGame(int p) throws Exception:**

- Write a code to play the game using the mentioned cue.
- If checkCue() method throws a **CueException** then returns a message "**Cannot use this cue**".(Use try-catch block)
- If it throws any other exception then return a message "**Other exception**".
- If no exception is found then return a message "**Welcome to the game**".

class **CueException** extends Exception

- Define **CueException** class derived from Exception class

Sample Input

```
Cue c=new Cue(13,true);
Retention r= new Retention(c);
String ans = r.playGame(5);
```

Sample Output

```
cannot use this cue
```

NOTE:

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

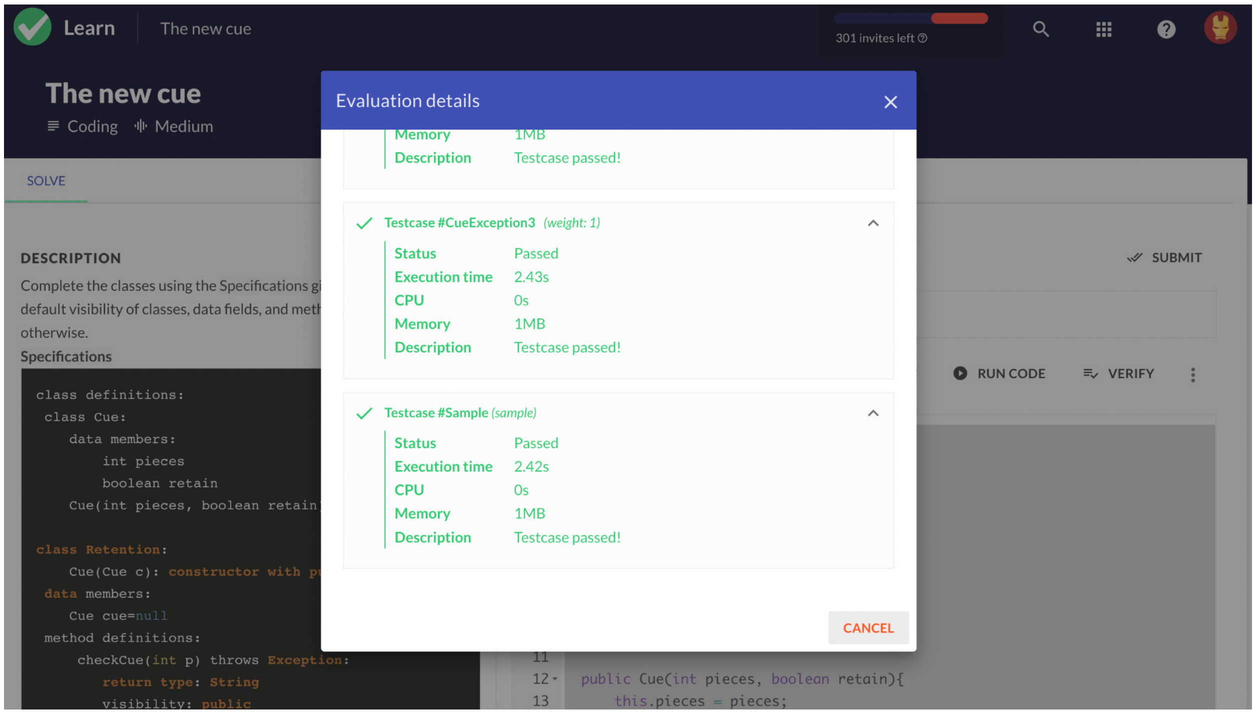
Correction in Question:

```
class Retention:
   Cue(Cue c): constructor with public visibility
Retention(Cue c): constructor with public visibility
```

Solution:

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

class Cue{
  //Your Code Goes Here..
  int pieces;
  boolean retain;

  public Cue(int pieces, boolean retain){
      this.pieces = pieces;
      this.retain = retain;
  }
}

class Retention{
  //Your Code Goes Here..
  Cue cue = null;

  public Retention(Cue c){
      this.cue = c;
  }

  public String checkCue(int p) throws Exception{

      if(!cue.retain)
        throw new CueException("Cue not retained");
      else if(p<cue.pieces)
        throw new CueException("More pieces required");
      else if(p>cue.pieces)
        throw new CueException("Update required");
      else
        return "Cue updated";
  }


  public String playGame(int p) throws Exception{

      try{
          checkCue(p);
          return "Welcome to the game";
      }
      catch(CueException e1){
          return "Cannot use this cue";
      }
      catch(Exception e2){
         return "Other exception";
      }
   }
}

class CueException extends Exception {
   //Your Code Goes Here..
   public CueException(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 */
         }
}
```

Verification:



The new cue

Coding · Medium

DESCRIPTION

Complete the classes using the Specifications gi
default visibility of classes, data fields, and meth
otherwise.

Specifications

```
class definitions:
 class Cue:
    data members:
        int pieces
        boolean retain
    Cue(int pieces, boolean retain

class Retention:
    Cue(Cue c): constructor with pu
 data members:
    Cue cue=null
 method definitions:
     checkCue(int p) throws Exception:
        return type: String
        visibility: public
```

Evaluation details

Memory	1MB
Description	Testcase passed!

Testcase #CueException3 *(weight: 1)*

Status	Passed
Execution time	2.43s
CPU	0s
Memory	1MB
Description	Testcase passed!

Testcase #Sample *(sample)*

Status	Passed
Execution time	2.42s
CPU	0s
Memory	1MB
Description	Testcase passed!

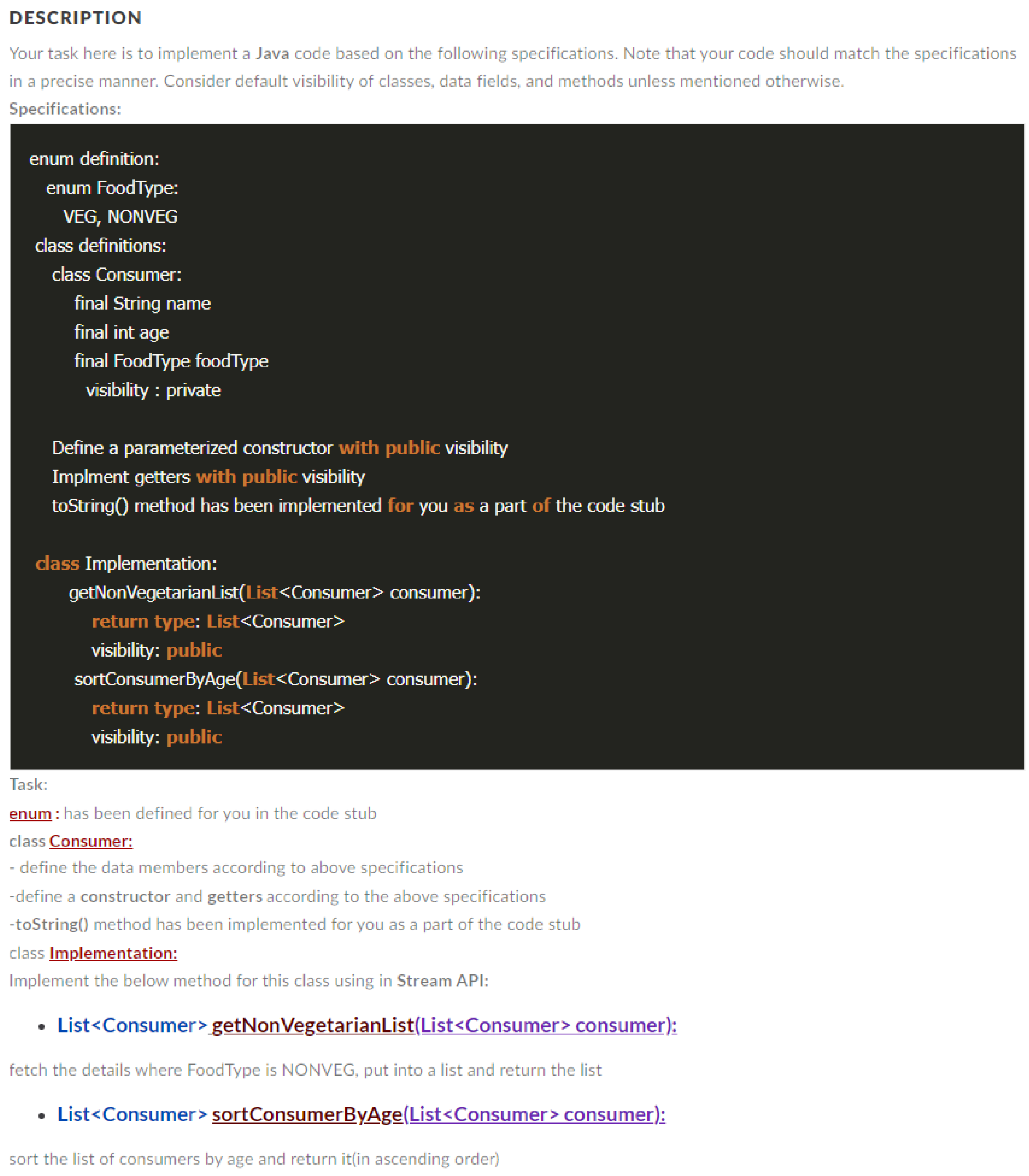
```
public Cue(int pieces, boolean retain){
    this.pieces = pieces;
```

Topic: Stream API & Lambda Expressions

Question Title: Nutrients

Correction in Question:

1) Add import for ArrayList because List in Java is Abstract and cannot be directly instantiated.
2) ~~sortConsumerByAge(List<Consumer> consumer):~~
   sortConsumersByAge(List<Consumer> consumer):
   return type: List<Consumer>
   visibility: public



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 unless mentioned otherwise.

Specifications:

```
enum definition:
   enum FoodType:
      VEG, NONVEG
 class definitions:
    class Consumer:
        final String name
        final int age
        final FoodType foodType
          visibility : private

    Define a parameterized constructor with public visibility
    Implment getters with public visibility
    toString() method has been implemented for you as a part of the code stub

 class Implementation:
       getNonVegetarianList(List<Consumer> consumer):
          return type: List<Consumer>
          visibility: public
        sortConsumerByAge(List<Consumer> consumer):
          return type: List<Consumer>
          visibility: public
```

Task:

enum : has been defined for you in the code stub

class **Consumer:**

- define the data members according to above specifications

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

-**toString()** method has been implemented for you as a part of the code stub

class **Implementation:**

Implement the below method for this class using in **Stream API:**

- List<Consumer> getNonVegetarianList(List<Consumer> consumer):

fetch the details where FoodType is NONVEG, put into a list and return the list

- List<Consumer> sortConsumerByAge(List<Consumer> consumer):

sort the list of consumers by age and return it(in ascending order)

Refer Sample Input Output for more details

Sample Input

```
Implementation imp = new Implementation();

 Consumer p = new Consumer("Sarah", 45, FoodType.VEG);
 Consumer p1 = new Consumer("John", 26, FoodType.NONVEG);
 Consumer p2 = new Consumer("Mirra", 7, FoodType.NONVEG);

List<Consumer> consumers = Arrays.asList(p, p1, p2);

imp.getNonVegetarianList(consumers)
imp.sortConsumersByAge(consumers)
```

Sample Output

```
[Consumer{name='John', age=26, foodType=NONVEG}, Consumer{name='Mirra', age=7, foodType=NONVEG}]
--------------------METHOD 1-----------------------------
[Consumer{name='Mirra', age=7, foodType=NONVEG}, Consumer{name='John', age=26, foodType=NONVEG}, Consumer{r
--------------------METHOD 2-----------------------------
```

NOTE

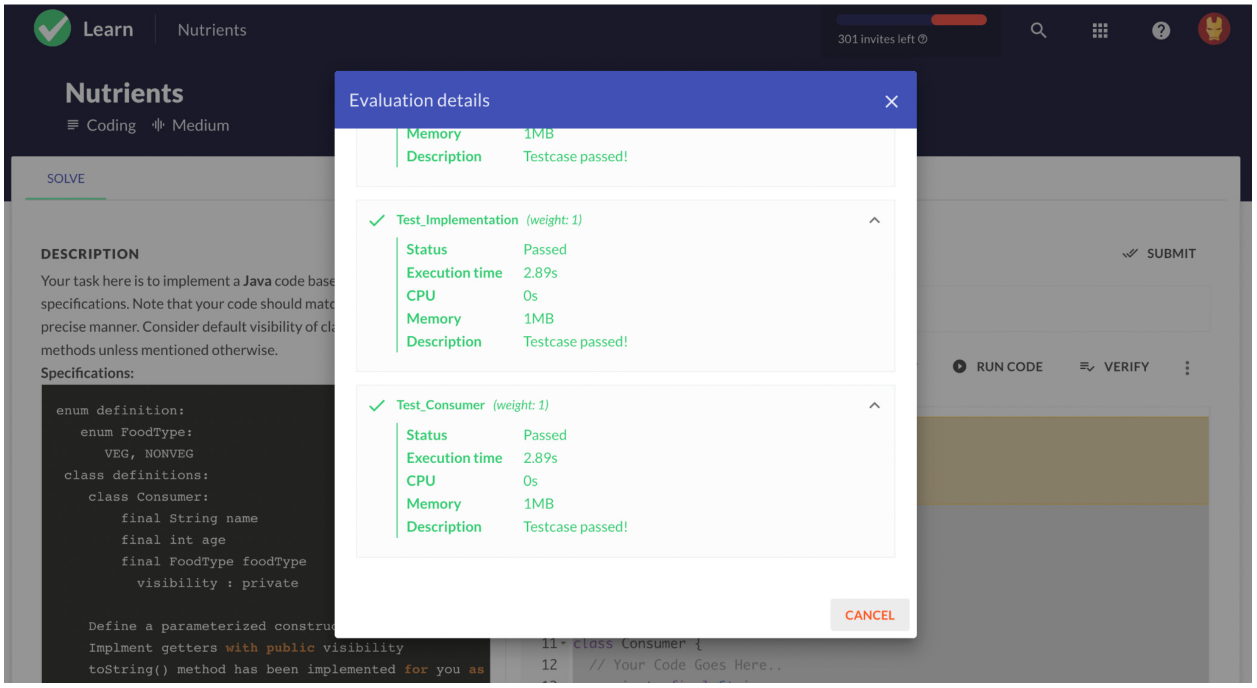
- You can make suitable function calls and use **the RUN CODE** button to check your **main()** method output.

Solution:

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;
import java.util.ArrayList;

enum FoodType {
  VEG, NONVEG
}

class Consumer {
  // Your Code Goes Here..
    private final String name;
    private final int age;
    private final FoodType foodType;

    public Consumer(String name, int age, FoodType foodType){
        this.name=name;
        this.age=age;
        this.foodType=foodType;
    }

    public String getName(){
        return this.name;
    }
    public int getAge(){
        return this.age;
    }

    public FoodType getFoodType(){
        return this.foodType;
    }


    @Override
    public String toString() {
        return "Consumer{" +
                "name='" + name + '\'' +
                ", age=" + age +
                ", foodType=" + foodType +
                '}';
    }
}

class Implementation{

  public List<Consumer> getNonVegetarianList(List<Consumer> consumer){

      List<Consumer> nonVegFoodType = new ArrayList<Consumer>();

      consumer.stream().forEach((c) -> {
          if(c.getFoodType().name().equals("NONVEG"))
           nonVegFoodType.add(c);
      });

      return nonVegFoodType;
  }

  public List<Consumer> sortConsumersByAge(List<Consumer> consumer){

      consumer = consumer.stream()
        .sorted((p1, p2)->Integer.valueOf(p1.getAge()).compareTo(Integer.valueOf(p2.getAge())))
        .collect(Collectors.toList());;

     return consumer;
  }

}
```

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


            Implementation imp = new Implementation();

    Consumer p = new Consumer("Sarah", 45, FoodType.VEG);
    Consumer p1 = new Consumer("John", 26, FoodType.NONVEG);
    Consumer p2 = new Consumer("Mirra", 7, FoodType.NONVEG);

   List<Consumer> consumers = Arrays.asList(p, p1, p2);

   System.out.println(imp.getNonVegetarianList(consumers));
   System.out.println(imp.sortConsumersByAge(consumers));
        }
}
```

Verification:



Nutrients

Coding   Medium

Evaluation details

DESCRIPTION

Your task here is to implement a **Java** code base
specifications. Note that your code should matc
precise manner. Consider default visibility of cla
methods unless mentioned otherwise.

Specifications:

```
enum definition:
   enum FoodType:
      VEG, NONVEG
 class definitions:
    class Consumer:
        final String name
        final int age
        final FoodType foodType
          visibility : private

    Define a parameterized construc
    Implment getters with public visibility
    toString() method has been implemented for you as
```

Memory	1MB
Description	Testcase passed!

✓ Test_Implementation (weight: 1)

Status	Passed
Execution time	2.89s
CPU	0s
Memory	1MB
Description	Testcase passed!

✓ Test_Consumer (weight: 1)

Status	Passed
Execution time	2.89s
CPU	0s
Memory	1MB
Description	Testcase passed!

```
class Consumer {
 // Your Code Goes Here..
```