BSCCS2005: Questions with Test Cases and Solutions

0.1 Cloning

Problem Statement

Write a Java program to create two objects `t1` and `t2` of type `Team`. `t2` should be created from `t1` using cloning such that any later changes to `t2` do not affect `t1`.

- Class `Project` implements a `Cloneable` interface and has/should have the following members:
  - Instance variables `String proj_name` and `double budget`
  - Constructor to initialize the instance variables
  - Overridden method `toString()`
  - Method `clone()`, to be implemented
- Class `Manager` implements a `Cloneable` interface and has/should have the following members:
  - Instance variables `String mngr_name` and `Project proj`
  - Constructor to initialize the instance variables
  - Overridden method `toString()`
  - Method `clone()`, to be implemented
- Class `Team` implements a `Cloneable` interface and has/should have the following members:
  - Instance variables `String teamName` and `Manager mngr`
  - Constructor to initialize the instance variables
  - Mutator methods as needed
  - Overridden method `toString()`
  - Method `clone()`, to be implemented
- Class `CloneTest` has the `main` method that takes the inputs and invokes appropriate methods to achieve the functionality.

What you have to do

- Implement method `clone()` in class `Project`
- Implement method `clone()` in class `Manager`
- Implement method `clone()` in class `Team`

Template Code

```
import java.util.*;
class Project implements Cloneable{
    private String proj_name;
    private double budget;
    public Project(String nm, double b) {
        proj_name = nm;
        budget = b;
    }
    public void setProjectName(String nm) {
        proj_name = nm;
    }
   public String toString() {
        return "Project: " + proj_name + ", budget: " + budget;
    }
    \\ Write code to implement clone() method
}
class Manager implements Cloneable {
    private String mngr_name;
    private Project proj;
    public Manager(String mn, Project p) {
        mngr_name = mn;
        proj = p;
    }
   public String toString() {
        return proj + "\n" + "Manager: " + mngr_name ;
    }
    \\ Write code to implement clone() method
}
class Team implements Cloneable {
    private String teamName;
    private Manager mngr;
    public Team(String tn, Manager m) {
        teamName = tn;
        mngr = m;
    }
    public void setTeamName(String tn) {
        teamName = tn;
    }
    public void setManager(Manager m) {
        mngr = m;
    }
   public String toString() {
        return teamName + "\n" + mngr ;
    }
    \\ Write code to implement clone() method
}
public class CloneTest {
    public static void main(String[] args) throws CloneNotSupportedException {
        Scanner sc = new Scanner(System.in);
        Project p1 = new Project("AI Development", 100000);
        Manager m1 = new Manager("Madhu", p1);
        Team t1 = new Team("Alpha", m1);
        Team t2 = t1.clone();
        t2.setTeamName(sc.nextLine());
        t2.setManager(new Manager(sc.nextLine(),
                      new Project(sc.nextLine(), sc.nextDouble())));
        System.out.println("Team t1: " + t1);
        System.out.println("Team t2: " + t2);
        sc.close();
    }
}
```

Public test case 1:
Input:

```
Beta
Rahul
ML Development
150000
```

Output:

```
Team t1: Alpha
Project: AI Development, budget: 100000.0
Manager: Madhu
Team t2: Beta
Project: ML Development, budget: 150000.0
Manager: Rahul
```

Public test case 2:
Input:

```
Gamma
Priya
IoT Deployment
0
```

Output:

```
Team t1: Alpha
Project: AI Development, budget: 100000.0
Manager: Madhu
Team t2: Gamma
Project: IoT Deployment, budget: 0.0
Manager: Priya
```

Private test case 1:
Input:

```
UltraMegaTeam
MegaManager
MegaManager
500000
```

Output:

```
Team t1: Alpha
Project: AI Development, budget: 100000.0
Manager: Madhu
Team t2: UltraMegaTeam
Project: MegaManager, budget: 500000.0
Manager: MegaManager
```

Private test case 1:
Input:

```
Beta
Akash
Cloud Migration
250000
```

Output:

```
Team t1: Alpha
Project: AI Development, budget: 100000.0
Manager: Madhu
Team t2: Beta
Project: Cloud Migration, budget: 250000.0
Manager: Akash
```

Solution:

```
import java.util.*;
class Project implements Cloneable{
    private String proj_name;
    private double budget;
    public Project(String nm, double b) {
        proj_name = nm;
        budget = b;
    }
    public void setProjectName(String nm) {
        proj_name = nm;
    }
   public String toString() {
        return "Project: " + proj_name + ", budget: " + budget;
    }
    public Project clone() throws CloneNotSupportedException {
        return (Project) super.clone();
    }
}
class Manager implements Cloneable {
    private String mngr_name;
    private Project proj;
    public Manager(String mn, Project p) {
        mngr_name = mn;
        proj = p;
    }
   public String toString() {
        return proj + "\n" + "Manager: " + mngr_name ;
    }
    public Manager clone() throws CloneNotSupportedException {
        Manager mngr_cloned = (Manager) super.clone();
        mngr_cloned.proj = proj.clone(); // Deep cloning the Project
        return mngr_cloned;
    }
}
class Team implements Cloneable {
    private String teamName;
    private Manager mngr;
    public Team(String tn, Manager m) {
        teamName = tn;
        mngr = m;
    }
    public void setTeamName(String tn) {
        teamName = tn;
    }
    public void setManager(Manager m) {
        mngr = m;
    }
   public String toString() {
        return teamName + "\n" + mngr ;
    }
    public Team clone() throws CloneNotSupportedException {
        Team tm_cloned = (Team) super.clone();
        tm_cloned.mngr = mngr.clone();
        return tm_cloned;
    }
}
public class CloneTest {
    public static void main(String[] args) throws CloneNotSupportedException {
        Scanner sc = new Scanner(System.in);
        Project p1 = new Project("AI Development", 100000);
        Manager m1 = new Manager("Madhu", p1);
        Team t1 = new Team("Alpha", m1);
        Team t2 = t1.clone();
        t2.setTeamName(sc.nextLine());
        t2.setManager(new Manager(sc.nextLine(),
                      new Project(sc.nextLine(), sc.nextDouble())));
        System.out.println("Team t1: " + t1);
        System.out.println("Team t2: " + t2);
        sc.close();
    }
}
```

0.2 Exception Handling

Problem Statement

Write a Java program to simulate an automatic car with speed control. The car should raise an exception in the following scenario:

- The driver tries to accelerate beyond the maximum speed limit (120 km/h).

The program should have the following classes:

- **Class** `SpeedLimitException` extends `Exception` and has/should have the following members:
  - Constructor accepting an error message.
- **Class** `Car` has/should have the following members:
  - Instance variables `String model` and `double speed`
  - Constructor to initialize the instance variables.
  - Method `void accelerate(double increment) throws SpeedLimitException`.
    - If the new speed (current speed + increment) exceeds the maximum speed limit of 120 km/h, the method throws a `SpeedLimitException` with the message `"Speed limit exceeded, Max allowed is 120 km/h."`
    - If the speed increase is within the allowed range, the method updates the speed of the car by adding the increment value to the current speed.
  - Overridden method `toString()`.
- **Class** `CarTest` with the `main` method, which:
  - Reads the car model and speed increments from user input.
  - Creates a `Car` object and calls the `accelerate`.
  - Catches and handles the `SpeedLimitException`, printing the error message if the speed limit is exceeded.
  - Prints the final state of the car.

What you have to do

- Implement class `SpeedLimitException`.
- Implement the `accelerate()` method in class `Car`.

Template Code

```
import java.util.*;
//Define class SpeedLimitException

class Car {
    private String model;
    private double speed;

    public Car(String model) {
        this.model = model;
        this.speed = 0.0;
    }

    public void accelerate(double increment) throws SpeedLimitException {
        // Implement the method
    }

    public String toString() {
        return "Car Model: " + model + ", Speed: " + speed + " km/h";
    }
}

public class CarTest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Car myCar = new Car(sc.nextLine());

        try {
            myCar.accelerate(sc.nextDouble());
            myCar.accelerate(sc.nextDouble());
        } catch (SpeedLimitException e) {
            System.out.println(e.getMessage());
        }

        System.out.println(myCar);
        sc.close();
    }
}
```

Public Test Case 1

Input:

```
Toyota
50
80
```

Output:

```
Speed limit exceeded, Max allowed is 120 km/h.
Car Model: Toyota, Speed: 50.0 km/h
```

Public Test Case 2

Input:

```
Honda
40
60
```

Output:

```
Car Model: Honda Civic, Speed: 100.0 km/h
```

Private Test Case 1

Input:

```
Ford
30
100
```

Output:

```
Speed limit exceeded, Max allowed is 120 km/h.
Car Model: Ford, Speed: 30.0 km/h
```

Private Test Case 2

Input:

```
Tesla
50
50
```

Output:

```
Car Model: Tesla, Speed: 100.0 km/h
```

Solution

```
import java.util.*;

class SpeedLimitException extends Exception {
    public SpeedLimitException(String message) {
        super(message);
    }
}

class Car {
    private String model;
    private double speed;

    public Car(String model) {
        this.model = model;
        this.speed = 0.0;
    }

    public void accelerate(double increment) throws SpeedLimitException {
        if (speed + increment > 120) {
            throw new SpeedLimitException("Speed limit exceeded, Max allowed is 120 km/h
        }
        speed += increment;
    }

    public String toString() {
        return "Car Model: " + model + ", Speed: " + speed + " km/h";
    }
}

public class CarTest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Car myCar = new Car(sc.nextLine());

        try {
            myCar.accelerate(sc.nextDouble());
            myCar.accelerate(sc.nextDouble());
        } catch (SpeedLimitException e) {
            System.out.println(e.getMessage());
        }

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

0.3 Streams

Problem Statement

Complete the Java program that, given a list of students, prints the list of students who are eligible for a scholarship. These include the students with an average CGPA > 7.5 and whose annual family income is less than ₹1,00,000. The program should also update the scholarship status of eligible students as "grade-1 scholarship" if their average CGPA is > 9.0; otherwise, the scholarship status should be updated as "grade-2 scholarship".

- Class `Student` has the following members:
  - Four instance variables: `name, scholarshipStatus, avgCGPA, income`
  - A constructor to initialize these instance variables
  - Mutator and accessor methods as needed
  - Overridden method `toString` to print the object.
- Class `StreamsTest` has / should have the following members:
  - Method `main` that accepts the details of four students, calls method `getEligibleStream` and prints the output list.
  - Method `getEligibleStream` that accepts a list of students, filters the students eligible for scholarship, and returns a stream of eligible students.
  - Method `updateScholarshipStatus` that accepts the list of eligible students and update their scholarship status.

What you have to do

- Define method `getEligibleStream` in class `StreamsTest`
- Define method `updateScholarshipStatus` in class `StreamsTest`

```
import java.util.ArrayList;
import java.util.Scanner;
import java.util.List;
import java.util.stream.*;

class Student {
    private String name, scholarshipStatus;
    private double avgCGPA, income;

    public Student(String n, double a, double i){
        name = n;
        avgCGPA = a;
        income = i;
        scholarshipStatus = "not eligible";
    }
    public String toString(){
        return name + " : " + avgCGPA + " : "
            + income + " : " + scholarshipStatus;
    }
    public double getAvgCGPA(){
        return avgCGPA;
    }
    public double getIncome(){
        return income;
    }
    public void setScholarshipStatus(String ss){
        scholarshipStatus = ss;
    }
}
public class StreamsTest{

     //Define method getEligibleStream here
     //Define method updateScholarshipStatus here

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        ArrayList<Student> sList = new ArrayList<Student>();
        Student s;
        for (int i = 0; i < 4; i++){
            s = new Student(sc.next(), sc.nextDouble(), sc.nextDouble());
            sList.add(s);
        }
        List<Student> eList =
            getEligibleStream(sList).collect(Collectors.toList());
        updateScholarshipStatus(eList);

        for (Student es : eList)
            System.out.println(es);

        sc.close();
    }
}
```

Public test case 1:
Input:

```
geet 9.5 80000
preet 8 90000
```

```
ravi 7 80000
kumar 8.5 200000
```

Output:

```
geet : 9.5 : 80000.0 : grade-1 scholarship
preet : 8.0 : 90000.0 : grade-2 scholarship
```

Public test case 2:
Input:

```
anuska 7.9 70000
ram 9.8 250000
geetha 9.1 90000
riya 8.5 90000
```

Output:

```
anuska : 7.9 : 70000.0 : grade-2 scholarship
geetha : 9.1 : 90000.0 : grade-1 scholarship
riya : 8.5 : 90000.0 : grade-2 scholarship
```

Private test case 1:
Input:

```
ravi 6.8 250000
rahul 7.8 20000
rajiv 9.1 180000
riya 9.2 180000
```

Output:

```
rahul : 7.8 : 20000.0 : grade-2 scholarship
```

Private test case 2:
Input:

```
dilip 7.8 90000
arun 9.2 80000
apurva 9.7 85000
kumar 8.5 95000
```

Output:

```
dilip : 7.8 : 90000.0 : grade-2 scholarship
arun : 9.2 : 80000.0 : grade-1 scholarship
apurva : 9.7 : 85000.0 : grade-1 scholarship
kumar : 8.5 : 95000.0 : grade-2 scholarship
```

Solution:

```
import java.util.ArrayList;
import java.util.Scanner;
import java.util.List;
import java.util.stream.*;

class Student {
    private String name, scholarshipStatus;
    private double avgCGPA, income;

    public Student(String n, double a, double i){
        name = n;
        avgCGPA = a;
        income = i;
        scholarshipStatus = "not eligible";
    }
    public String toString(){
        return name + " : " + avgCGPA + " : "
            + income + " : " + scholarshipStatus;
    }
    public double getAvgCGPA(){
        return avgCGPA;
    }
    public double getIncome(){
        return income;
    }
    public void setScholarshipStatus(String ss){
        scholarshipStatus = ss;
    }
}
public class StreamsTest{
    public static Stream<Student> getEligibleStream(ArrayList<Student> sList){
        Stream<Student> sStream
                    = sList.stream()
                            .filter((s) -> (s.getIncome() < 100000
                                 && s.getAvgCGPA() > 7.5));
        return sStream;
    }
    public static void updateScholarshipStatus(List<Student> eList){
        for (Student es : eList){
            if (es.getAvgCGPA() > 9)
                es.setScholarshipStatus("grade-1 scholarship");
            else
                es.setScholarshipStatus("grade-2 scholarship");
        }
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        ArrayList<Student> sList = new ArrayList<Student>();
        Student s;
        for (int i = 0; i < 4; i++){
            s = new Student(sc.next(), sc.nextDouble(), sc.nextDouble());
            sList.add(s);
        }
        List<Student> eList =
            getEligibleStream(sList).collect(Collectors.toList());
        updateScholarshipStatus(eList);

        for (Student es : eList)
            System.out.println(es);

        sc.close();
    }
}
```