Homework 6

Due: Dec. 5, 11:59 PM

Name:

Student ID:

Submission Instructions: This PDF contains fillable fields where you can input your answers. (For example, you can input your name and ID in the fields above.) Please save the document regularly so your answers are not lost. After you complete the assignment, upload a copy to Gradescope. A link to Gradescope can be found on the left side of the course Canvas page.

Important Note on Academic Integrity: This assignment should be completed individually. In recent semesters, improper collaboration on homework has led to multiple cases of plagiarism, where we receive identical or nearly identical submissions from two or more students. If you decide to discuss this assignment with other students before the deadline, make sure you first read the section of the syllabus on proper and improper collaboration. Additionally, you must include the names of these students below.

Names of Collaborators (if any):

Question 1 (15 points): Complete the UML class diagram on the next page for a class that stores movie information. Each object should store the following data about a single movie:

1. Title (e.g., Groundhog Day)
2. Release year (e.g., 1993)
3. Length in minutes (e.g., 101)
4. Rating (e.g., PG)
5. Director name (e.g., Harold Ramis)
6. List of actors (e.g., Bill Murray, Andie MacDowell)
7. List of genre tags (e.g., comedy, fantasy, romance)
8. List of review scores (Each review is an integer between 0 and 8.)
9. Average review score

Make sure you include the data type for each field using the correct UML notation (i.e., identifier followed by type and separated by a colon). For fields that store a list, use an ArrayList of the appropriate type (e.g., ArrayList<Boolean>).

The data stored in each object should be private. In order to access and modify the data, the class needs the following public instance methods:

1. Accessor methods to return each field. For ArrayList fields, the corresponding accessor should return a normal array with a copy of the data.[1]
2. Mutator methods to add and remove items from the actor, genre, and review score lists.

Make sure you include the parameters and return type for each method (in UML notation).

Finally, the class needs a constructor to initialize each object. Note that there are no mutator methods to set the title, release year, length, rating, or director. This means that any constructor must include parameters for these fields. Include signatures in the class diagram for the following two constructors:

1. A constructor with parameters for the five fields listed above.
2. A constructor with parameters for every field other than the average review score. (The average is calculated by the class every time a new score is added, rather than being set by the user.)

[1] If a reference to an ArrayList is returned, then the calling method can use it to modify the field (e.g., by calling add, set, or remove). Returning a copy ensures that the data is protected. Note that the ArrayList class has a method named “toArray” that returns a normal array with a copy of the data.

Question 2 (20 points): Trace the execution of the program shown on the next two pages in the memory diagram below. Use the following guidelines:

- When constructing an ArrayList, show the empty elements by writing their indices in the identifier column and leaving their contents blank. (Don't forget to include the size.)
- Show the address stored in each reference variable rather than the contents of the corresponding object. Note that this applies to class fields and ArrayList elements.
- If the contents of a variable change, use a comma, space, or some other delimiter to separate the old and new values.
- The memory diagram only includes a stack frame for the main method. You do not need to trace the variables stored in the stack frame of any other method.

main Stack Frame

Identifier	Address	Content
	200	
	201	
	202	
	203	
	204	
	205	
	206	
	207	
	208	
	209	
	210	
	211	
	212	
	213	
	214	
	215	
	216	
	217	
	218	
	219	
	220	
	221	
	222	
	223	
	224	
	225	
	226	
	227	
	228	
	229	

Heap

Identifier	Address	Content
	5000	
	5001	
	5002	
	5003	
	5004	
	5005	
	5006	
	5007	
	5008	
	5009	
	5010	
	5011	
	5012	
	5013	
	5014	
	5015	
	5016	
	5017	
	5018	
	5019	
	5020	
	5021	
	5022	
	5023	
	5024	
	5025	
	5026	
	5027	
	5028	
	5029	

Course.java

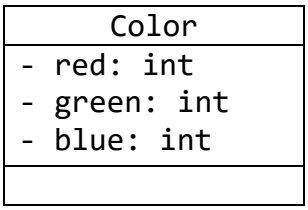
```
import java.util.ArrayList;

public class Course {

    public static void main(String[] args) {

        Student first = new Student("Frodo", 83, 74, 91);
        Student second = new Student("Sam", 60, 93, 87);
        Student third = new Student("Merry", 86, 92, 75);

        ArrayList<Student> students = new ArrayList<Student>(5);
        students.add(first);
        students.add(second);
        students.add(third);

        Student update = students.get(0);
        update.setMidterm1(85);
        update = students.get(2);
        update.setMidterm3(81);
    }
}
```

The program continues on the next page.

Student.java

```
public class Student {

    private String name;
    private int midterm1;
    private int midterm2;
    private int midterm3;

    public Student(String name, int midterm1, int midterm2, int midterm3) {
        this.name = name;
        this.midterm1 = midterm1;
        this.midterm2 = midterm2;
        this.midterm3 = midterm3;
    }

    public char findGrade() {

        double avg = getAverage();

        if (avg >= 90.0) {
            return 'A';
        } else if (avg >= 80.0) {
            return 'B';
        } else if (avg >= 70.0) {
            return 'C';
        } else if (avg >= 40.0) {
            return 'D';
        } else {
            return 'F';
        }
    }

    public String getName() {
        return name;
    }

    public double getAverage() {
        return (midterm1 + midterm2 + midterm3) / 3.0;
    }

    public void setMidterm1(int midterm1) {
        this.midterm1 = midterm1;
    }

    public void setMidterm2(int midterm2) {
        this.midterm2 = midterm2;
    }

    public void setMidterm3(int midterm3)
    {
        this.midterm3 = midterm3;
    }
}
```

Question 3 (15 points; 3 per part): Consider the incomplete UML class diagram below.



Objects of this class represent colors in the RGB model, where each color is uniquely specified by a different combination of red, green, and blue light intensities.

The bottom compartment of the diagram is missing the method signatures of the class. Descriptions of some of these methods are given below. For each description, write the method signature. Be sure to indicate whether each method is static.

a) The class has a constructor that creates a new Color object that is a copy of an existing Color object.

b) The class has a constructor that creates a new Color object by averaging the RGB values of two existing Color objects.

c) The class has an instance method named “mix” that creates a new Color object by averaging the RGB values of an existing object and the object the method is called on.

d) The class has a class method named “mix” that creates a new Color object by averaging the RGB values of two existing Color objects.

e) The class has an instance method named “rgbValues” that returns the RGB values of a Color object in an array.