University of Washington
Department of Computer Science & Engineering

CSE 142 Winter 2003 -- Homework #4

CSE 142 Homework #4

Due: Wednesday, February 19 at 9:00 PM. No late assignments will be accepted.

Directions: Please answer the following questions. Use full sentences when answering questions that require explanations. Remember to acknowledge any help you receive from individuals outside the course staff. These questions are based on material from lecture and from Chapters 7 thru 12 in the textbook.

You answers should be formatted using Microsoft Word or WordPad (.doc file), Notepad or any plain text editor (.txt), or any other file format that can be read by recent versions of Microsoft Word. When you are done, use this turnin form to submit this assignment along with part A of programming project 2 over the web. If you make a mistake, you can resubmit your assignment as often as needed. We will grade the last one that you hand in. Your graded assignment will be returned to you via email. Please include the questions along with your answers in your homework file.

Remember to get started right away. Don't wait until Wednesday after dinner to begin, or you are likely not to finish on time. Just as a reminder, homework in this course is to help you understand the conceptual material that we cover. If you need assistance, please get help from the course staff.

Grading Guidelines: Unless indicated otherwise, each question or part of a question is worth 2 points. To gain the full 2 points, your answer must have no significant flaws, must be clear to the reader, and must be complete (but remember that complete does not mean verbose - be concise and to the point). One point will be awarded to answers that are partially correct, are somewhat unclear, or are incomplete. You must show credible effort to gain at least one point per question.

1a. [2 points] Finding and testing for prime numbers plays an important part in encrypting information to keep data transfers between computers secure. Write the Java specification and implementation for a method called `isPrime` that takes an integer `num` as a parameter and **returns true** if `num` is prime and **returns false** if `num` is not prime (also known as composite). Remember, a number is prime if and only if the only divisors of the number are 1 and the number. For example, 11 is a prime number while 8 is not. It only makes sense to check positive numbers to see if they are prime, so you just need to handle positive numbers in your code. Where appropriate, include the preconditions and postconditions for this method. *Hint: you'll probably need to loop through some possible divisors for the number. If you are clever, you won't need to loop through all possible divisors.*

1b. [1 point] Describe why you chose the type of loop(s) -- for, while, etc.-- you used in `isPrime`.

1c. [2 points] Suggest a list of test cases for values of `num`. Describe why you included each test case (typical case, edge case, "incorrect" case) in your set of test cases.

2a. [2 points] Now that you can test for prime numbers, use that method to find the next prime number after a given number. Write the Java specification and implementation for a method called `nextPrimeAfter` that takes an integer `num` as a parameter and **returns the smallest integer that is prime and is greater than `num`**. For example, if `num` is 20, then `nextPrimeAfter` should return 23, since 23 is the smallest prime number after 20. Where appropriate, include the preconditions and postconditions for this method. You should use `isPrime` from question 1 to check if a number is prime.

2b. [1 point] Describe why you chose the type of loop(s) -- for, while, etc. -- you used in `nextPrimeAfter`.

2c. [2 points] Suggest a list of test cases for values of `num`. Describe why you included each test case (typical case, edge case, "incorrect" case) in your set of test cases.

Use the following definition for the Movie class to answer questions 3 - 6.

```
/** A simple Movie class */

public class Movie {

   // static variables
     private static double overdueRateNew = 1.5;
     private static double overdueRateOld = 1.2;
   // instance variables
    String title;             // movie title
    boolean isNewRelease;     // true if movie is a new release
    double rentalPricePerDay; // movie rental price per day
                              // class invariant: price must be >= 0.0
    int idNumber;             // id number for movie
    // methods
    /** calculates and returns movie rental price for numDays
    * postcondition: returns the numDays multiplied by the rentalPricePerDay
    */
    public double rentalPrice(int numDays) {
        if (numDays <= 0) {
            return 0.0;
        }
        else {
            double totalPrice = numDays * rentalPricePerDay;
            return totalPrice;
        }
    }

    /** calculates and returns the overdue fine for numDaysOverdue
    * postcondtion: returns the total fine for the number of days.
    * new release movies have a fine of the rentalPricePerDay times overdue
    * rate for new movies per day
    * non-new release movies have a fine of the overdue rate for old movies
    * times the rentalPricePerDay
    */
    public double overduePrice(int numDaysOverdue) {
        double finePrice;
        if (numDaysOverdue <= 0) {
            return 0.0;
        }
```

```
        if (isNewRelease) {
            finePrice = numDaysOverdue * rentalPricePerDay * overdueRateNew;
        }
        else {
            finePrice = numDaysOverdue * rentalPricePerDay * overdueRateOld;
        }
        return finePrice;
    }

    // other methods omitted

        /** create a new Movie object given the title, new release information,
    * rental price per day, and id number
    * postcondition: all instance variables initialized
    */
    public Movie (String title, boolean isNewRelease,
        double rentalPricePerDay, int idNumber) {
        this.title = title;
        this.isNewRelease = isNewRelease;
        this.rentalPricePerDay = rentalPricePerDay;
        this.idNumber = idNumber;
    }
}
```

3. [2 points] Show the object diagrams with appropriate scope of methods and variables for the execution of the following code in the interactions window. Please be sure to leave crossed out boxes visible -- just put one line through crossed out boxes. Paste your diagram into your homework file.

```
Movie music = new Movie ("The Sound of Music", false, .50, 1234);

double rentalPriceMusic = music.rentalPrice(4);

Movie wedding = new Movie("My Big Fat Greek Wedding", true, 1.25, 2314);

double finePriceWedding = wedding.overduePrice(3);
```

4. [2 points] Be sure the class definition for Movie works properly for the two methods defined and the constructor. Ensure that it works correctly by placing appropriate `System.out.println` statements in the code. Be sure that the code does what is expected for negative values of `numDays` and `numDaysOverdue`. Be sure that the overdue price is calculated correctly for new releases and non-new releases. Paste your code with the `System.out.println` statements into your homework file for your answer to this question. Also, create a `toString` method for this class to print out the object's state and paste the code for this method in your homework file. *Hint: recall that the `toString` method returns a String.*

Use the following code to help answer questions 5 and 6.

```
import java.util.*;

/** A simple MovieInventory class */

public class MovieInventory {
    // instance variable
    ArrayList movieList; // list of movies
```

```
    // methods

    /** add movie m to inventory */
    public void addToInventory (Movie m) {
        movieList.add(m);
    }
    /** create a new movie inventory object with no movies */
    public MovieInventory () {
        movieList = new ArrayList();
    }
}
```

5a. [2 points] Suppose the movie rental store keeps an inventory of its movies by storing the Movie objects in an `ArrayList` in the class `MovieInventory`. Write the Java specifications and implementation for a method called `inInventory` in the `MovieInventory` class that takes as a parameter an int `idNumber` and **returns true** if a movie with the `idNumber` is in the movie list and **returns false** if no idNumbers of movies match the parameter `idNumber`. Paste your specifications and implementation into your homework file. *Hint: You'll want to create an iterator for the ArrayList to get each movie in order to check it's id number. Remember, you need to cast to the appropriate type when getting objects out of the ArrayList.*

5b. [2 points] Test your method inInventory by creating a test case using a main method. You'll probably want to create a separate class just for the main method. Inside main, create a MovieInventory object, create a bunch of movie objects, and add the movie objects to the MovieInventory. Test to see if the method inInventory works properly. Paste the code for your test case into your homework file. *Hint: writing toString for MovieInventory may be useful.*

6a. [2 points] Write the Java specifications and implementation for a method called `getNumNewReleases` that does not take any parameters and **returns the total number of movies that are new releases**. For example, if there are 10 movies in the ArrayList and 6 of the movies are new releases, the method should return the number 6. Paste your specification and implementation into your homework file. *Hint: you'll want to create an iterator for the ArrayList to get each movie.*

6b. [2 points] Test your method getNumNewReleases by creating a test case using a main method. You can use the same main method as in 5b, but this time test the method getNumNewReleases. Paste the code for your test case into your homework file.

7. [2 points] [Based on exercise 11.4] [No Java programming required] Assume a university registration system has been designed in Java. The designers of the system chose to model a student with the following properties: name, id number, address, courses enrolled in, and instructors of courses enrolled in. Explain how you would modify the design to minimize coupling and maximize cohesion.

8. [2 points] Please complete part A of Project 2 with your partner. Use this turnin form to submit the files for part A of the project along with this assignment. (Submit the files separately; don't paste the code for the project into the file containing your solutions to this homework.) Both you and your partner should turn in the code for this homework assignment, but the code should be the same since you are working on the project together.