SUNY, Korea Computer Science

CSE 114 Handout 10: Problem Set 10

Due: Dec 6, 2019
Points: 30

Read This Right Away

This problem set is due at **11:59 pm on Dec 6, 2019, KST**. Don't go by the due date that you see on Blackboard because it is in EST. Go by the one given in this handout.

• To solve each problem below, you will be implementing a Java class.
• Please read carefully and follow the directions exactly for each problem. Files and classes should be named exactly as directed in the problem (including capitalization!) as this will help with grading.
• You should create your programs using emacs.
• Your programs should be formatted in a way that is readable. In other words, indent appropriately, use informative names for variables, etc. If you are uncertain about what a readable style is, see the examples from class and textbook as a starting point for a reasonable coding style.
• This problem set assumes you have installed Java and emacs on your computer. Please see the course web for installation instructions.
• You must use the command-line interface to run your programs. That is, you must use the `javac` and `java` commands to do this problem set. Do not use Eclipse yet.
• **Important!** You must use only 1 Scanner object. Assignments may require a Scanner object to be passed into other methods but do not create a new Scanner inside those methods! Use the one passed to the method. ***Creating multiple scanners causes a problem with the automated grading procedure and doing so will result in a loss of points on the assignment!***
• **Be sure to include** a comment at the top of each file submitted that gives **your name** and **email address**.
• **Remember:** We are now also assessing coding style. Be sure to use consistent indentation, good and informative variable names and write comments in your code to describe what it is doing.

What Java Features to Use

For this assignment, you are *not* allowed to use more advanced features in Java than what we have studied in class so far.

Problem 1 (15 Points)

Place each of the following problem solutions in a class called `Recursion` in a file named `Recursion.java`.

1. (3 pts) This following algorithm is known as Euclid's Algorithm because it appears in Euclid's Elements (Book 7, ca. 300 B.C.). It may be the oldest nontrivial algorithm. The algorithm is based on the observation that if r is the remainder when a is divided by b, then the common divisors of a and b are the same as the common divisors of b and r. Thus we can use the equation:

   gcd(a, b) = gcd(b, r)

   to successively reduce the problem of computing a GCD to the problem of computing the GCD of smaller and smaller pairs of integers. For example,

   gcd(36, 20) = gcd(20, 16) = gcd(16, 4) = gcd(4, 0) = 4

   implies that the GCD of 36 and 20 is 4. It can be shown that for any two starting numbers, this repeated reduction eventually produces a pair where the second number is 0. Then, the GCD is the other number in the pair.

   Write a recursive method called *gcd* that takes two integer parameters and that uses Euclid's Algorithm to compute and return the greatest common divisor of the two numbers. In your main call *gcd* as follows:

   ```
   System.out.println("gcd(36, 20) = " + gcd(36, 20));
   System.out.println("gcd(34, 0) = " + gcd(34, 0));
   System.out.println("gcd(3346, 468) = " + gcd(3346, 468));
   ```

2. (3 pts) A function f is defined by the following rule:

   ```
   f(n) = n if n < 3
          3f(n − 1) + 2f(n − 2) + f(n − 3) if n >= 3
   ```

   Write a recursive method called f that takes an int as a parameter and that computes and returns the value of the given function. In your main call f as follows:

   ```
   System.out.println("f(2) = " + f(2));
   System.out.println("f(3) = " + f(3));
   System.out.println("f(10) = " + f(10));
   ```

3. (3 pts) We have seen how to use the charAt, substring, and length functions defined on String. In fact, for substring, we saw two variations: one taking one argument and another taking two arguments. You will be using those functions (and others if you need) to solve the next several problems. First, write a recursive method called printBackward that takes a String as a parameter and that prints the characters of the String backwards (one character per line). It should be a void method. In your main call printBackward as follows:

   ```
   printBackward("Java Fun!");
   ```

4. (3 pts) Write a recursive method called `printString` that does the same thing as `printBackward` but that prints the `String` in the same order of characters in the `String` given as the argument (one character per line). In your `main` call `printString` as follows:

   ```
   printString("Java Fun!");
   ```

5. (3 pts) Write a recursive method called `reverseString` that takes a `String` as a parameter and that returns a new `String` as a return value. The new `String` should contain the same characters as the parameter, but in reverse order. For example, the output of the following code

   ```
   String reversed = reverseString("Java Fun!");
   System.out.println(reversed);
   ```

should be

!nuF avaJ

In your `main,` write calls to each of the above as shown in the problem descriptions.

Turn in your completed `Recursion.java` holding the solutions to the above 5 problems.

Problem 2 (5 Points)

Write a method that satisfies the following description:

```
// Returns true if x occurs in the range [i, nums.length - 1] in nums.
// That is, if x is found in nums between the indices i and
// nums.length - 1 inclusive. This method must be recursive.
public static boolean occurs(int x, int[] nums, int i) {
// fill this in
}
```

Now write another method that satisfies the following description. This method would find `occurs` that you defined above useful.

```
// Returns true if the elements in nums are unique, i.e., if there
// are no duplicates.
public static boolean unique(int[] nums) {
// fill this in
}
```

Note that `unique` itself may not be recursive, but you would find it helpful to define an auxiliary function that is recursive. *If you write such an auxiliary function, make it recursive*.

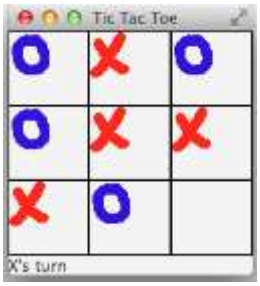
In the [given] folder (or in the assignment dropbox on blackboard), I provided a skeleton (`Unique.java`) with a `main` that you should use to develop your solution.

Turn in your finished `Unique.java`.

Problem 3 (10 points)

Note that for this problem, you should use the command line java compiler. There have been difficulties setting up and using the JavaFX libraries in eclipse. Start with `TTT.java` in the [given] folder (or in the assignment dropbox on blackboard) and do the following.

1. Run `TTT.java` to see how it works and how it appears to you, the user.
2. The original `TTT.java` given displays an 'X' with two lines and an 'O' as an ellipse. In this problem you will revise the given `TTT.java` in such a way that it will display 'X's and 'O's as images [red Xs and blue Os given in x.gif and o.gif in the [given] folder or on the assignment dropbox. Once you are done, it will look something like the following, which is also given as `Sample.png` in the [given] folder (or in the assignment dropbox):



3. This much is required for this problem set

Optionally (meaning this part is *not* required and I am **not** offering any extra credit), you can improve the given game as follows:

1. As you can see in `Sample.png`, the letters 'X' and 'O' are not centered in each cell. Make them appear in the center of each cell. They should stay in the center even if you resize the window.
2. The given version of Tic Tac Toe continues until the entire board is filled up even when it is already a tie game. Improve your game so that it can detect a tie game if it is a tie game even if the board has not been completely filled up yet.
3. Improve it further so that it is played by a human against a computer.

Hand in your revised `TTT.java` along with the image files. Make sure that you preserve the folder structure in such a way that I will not have to edit your `TTT.java` to run your program.

Submission Instructions

Please follow this procedure for submission:

1. Place the deliverable files (`Recursion.java, Unique.java,` and `TTT.java`) into a folder by themselves. The folder's name should be CSE114_PS10_<yourname>_<yourid>. So if your name is Joe Cool and your id is 12345678, the folder should be named 'CSE114_PS10_JoeCool_12345678.

2. Compress the folder and submit the zip file.
    a. On windows, do this by pressing the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Blackboard dropbox for PS10.
    b. On mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Blackboard dropbox for PS10.
3. Navigate to the course blackboard site. Click **Assignments** in the left column menu. Click **PS10** in the content area. Under **ASSIGNMENT SUBMISSION**, click **Browse My Computer** next to **Attach Files.** Find the zip file and click it to upload it to the web site.
4. ***Important!!*** Click the **Submit** button! If you fail to click **Submit**, the assignment will not be downloadable and will not be graded.