CSCI 161 Introduction to Computer Science

Review Guide 2 (Soln)

Practice Problems

1. While loops have the same expressive power as for loops (i.e., can be used interchangeably).

   TRUE / FALSE
   Explain:

2. Private methods can only be called by methods within the same class, which is why they are not very useful in practice.

   TRUE / FALSE
   Explain:

   False. Private methods are simply hidden so that they aren't visible from classes. This doesn't mean that they aren't useful. They usually help perform important subtasks for other methods. For example, in our `ChatBot`, `fillSnarkyResponses()` is called only once ever from the constructor. It doesn't make any sense for others to call it any other time! Exposing it to users would just confuse them.

3. **Explain why the following code prints 0.

```
ArrayList<Integer> A = new ArrayList<Integer>(100); // create 100 elements?
System.out.println(A.size());
```

4. What is a "wrapper class?" What is its purpose?

5. Define a class called `Point` that represents an (x, y) pair of integers. For full credit, your class must have a two-argument constructor, getters for retrieving x and y values, a `toString()` method, and an `equals()` method for comparing instances of Point as shown below. (Two instances are equal if their x coordinates are equal, as are their y coordinates.)

```java
public class Point
{
    private int x;
    private int y;

    public Point(int newX, int newY)
    {
        this.x = newX;
        this.y = newY;
    }

    public int getX()
    {
        return this.x;
    }

    public int getY()
    {
        return this.y;
    }

    public boolean equals(Point other)
    {
        return x == other.x && y == other.y;
    }

    public String toString()
    {
        return "(" + this.x + "," + this.y + ")";
    }
}
```

6. Next, define a class called `PointCollection`, which stores an `ArrayList` of `Point` objects. Implement the following methods:

   - `public void insertInPlace(Point p)` – inserts the given `Point` into the collection. It must be inserted in such a way that the collection of points is sorted in ascending order of the x value.
   - `public boolean notExists(Point p)` – returns true if the given point is not found in the collection, and false otherwise.
   - `public String toString()` – returns a String representing the collection of points.

```java
import java.util.ArrayList;

public class PointCollection
{
    private ArrayList<Point> pointlist;

    public PointCollection()
    {
        this.pointlist = new ArrayList<Point>();
    }

    public void insertInPlace(Point p)
    {
        //skip elements in the list as long as p's X-value
        //is greater than the current element
        int i = 0;
        while (i < this.pointlist.size() && p.getX() > this.pointlist.get(i).↩
            getX())
        {
            i++;
        }

        //i now refers to the position in which to insert p
        this.pointlist.add(i, p);    //insert it in place
    }

    public boolean notExists(Point p)
    {
        for (int i = 0; i < this.pointlist.size(); i++)
        {
            //look to see if p is equal to the i_th point
            //if so, then p exists in the list, and we return false
            if (p.equals(this.pointlist.get(i)))
            {
                return false;
            }
        }
        //made it through the loop without finding p
        return true;
    }

    public String toString()
    {
        String s = "";
        for (int i = 0; i < this.pointlist.size(); i++)
        {
            //append string representation of i_th point
            s += this.pointlist.get(i).toString();

            //append a comma behind the point if we
            //haven't reached the last element
            if (i < this.pointlist.size()-1)
            {
                s += ", ";
            }
        }
        return s;
    }
}
```

7. Consider the method below:

```java
public void mystery(int x)
{
    Circle c;
    for(int i=x; i > 0; i -= 1)
    {
        c = new Circle();
        c.changeSize(i*20);
        if (i % 2 == 0) {
            c.changeColor("red");
        }
        else {
            c.changeColor("blue");
        }
        c.makeVisible();
    }
}
```

(a) What does this method do if passed a zero when invoked?

(b) What does this method do if passed a three when invoked?

(c) Describe what the method does. (To get in the right frame of mind, think about what you'd write as a comment for this method.)

8. ** Write a method that inputs two int values a and b. The method should return $a^b = a \times a \times a \times ...$ (b times). You may assume $a \geq 0$ and $b \geq 0$.

9. ** Write a method that accepts an integer, then print out all positive odd numbers less than or equal to that integer. For instance, if 6 was input, then print 1, 3, 5.

10. ** The Fibonacci Sequence starts off with 0, followed by 1. The next number is always the sum of the two previous numbers. Therefore, the first 6 Fibonacci numbers in the sequence is: 0, 1, 1, 2, 3, 5. Write a method called `void fib(int n)` that prints the first n Fibonacci numbers.

11. Consider the method defined below. Explain what it does using a few brief sentences.

```
public int mystery(int[] A, int B)
{
    int x = 0;
    for (int i = 0; i < A.length; i++)
    {
        if (A[i] < B) {
            x++;
        }
    }
    return x;
}
```

This method returns the number of values in A[] that are smaller than B.

12. ** Consider the method defined below. Explain what it does using a few brief sentences.

```
public boolean mystery(int[] A)
{
    for (int i = 0; i < A.length; i++) {
        for (int j = i+1; j < A.length; j++) {
            if (A[i] == A[j]) {
                return true;
            }
        }
    }
    return false;
}
```

This method returns true if there is a duplicate value found in A, and false otherwise.

13. ** Consider the method defined below. Explain what it does using a few brief sentences.

```
public boolean mystery(int[][] A)
{
    for (int i = 0; i < A.length; i++)
    {
        for (int j = 0; j < A[i].length; j++)
        {
            if (A[i][j] < 0)
            {
                return false;
            }
        }
    }
    return true;
}
```

This method determines whether all elements in the 2D array A are non-negative.

14. ** Consider the method defined below. What is the output when `mystery("john doe")` is called?

```
public String mystery(String name)
{
    String[] name_tokens = name.split(" ");

    String ret = name_tokens[1].substring(0,1).toUppercase() + name_tokens[1].↩
        substring(1,name_tokens[1].length()).toLowercase() + ", " + name_tokens↩
        [0].substring(0,1).toUppercase();

    return ret;
}
```

"Doe, J" is the returned value. This method takes any full name separated by space, and returns a string that contains the last name, first initial.

15. Write a method `public int findLargest(int[][] A)` that returns the largest number found in the given 2D array. You **may** assume that `A` is not `null` and is of a square shape. Consider the following usage:

```
public static int findLargest(int[][] A)
{
    int max = Integer.MIN_VALUE;    //set max to -infinity
    for (int i = 0; i < A.length; i++)
    {
        for (int j = 0; j < A[i].length; j++)
        {
            //saw a larger element, replace the max
            if (max < A[i][j])
            {
                max = A[i][j];
            }
        }
    }
    return max;
}
```

16. Write a method `public int maxRow(int[][] A)` that returns the row number that has the maximum sum. You **may** assume that `A` is not `null` and is of a square shape.

```
public static int maxRow(int[][] A)
{
    int maxRow = 0; //assume first row has the max sum
    int maxSum = 0; //now compute the sum of the first row
    for (int j = 0; j < A[0].length; j++)
    {
        maxSum += A[0][j];
    }

    //now, loop through all other rows and compare their sums
    for (int i = 1; i < A.length; i++)
    {
        int rowSum = 0; //holds the sum for the current row
        for (int j = 0; j < A[i].length; j++)
        {
            rowSum += A[i][j];
        }

        //found a larger row
        if (rowSum > maxSum)
        {
            maxSum = rowSum;
            maxRow = i;
        }
    }
    return maxRow;
}
```

17. ** Consider the Notebook class below. The constructor has not been provided.

   (a) Write the default constructor, which creates an empty collection of notes.
   (b) Write a new Notebook method called `countNotesContaining(..)` that takes a String as argument, and returns the number of notes in the notebook that contain the specified string.
   (c) Write a new Notebook method called `getNotesContaining(..)` that takes a String as argument, and returns an array list of notes in the notebook that contain the specified string.
   (d) Write a new method called `countWords(..)` that counts the total number of words in all the stored notes.

```java
import java.util.ArrayList;
public class Notebook
{
    private ArrayList<String> notes;

    public Notebook()
    {
        this.notes = new ArrayList<>();
    }

    public void storeNote(String n)
    {
        this.notes.add(n);
    }

    public int countNotesContaining(String term)
    {
        int numNotes = 0;
        for (int i = 0; i < this.notes.size(); i++)
        {
            //contains() comes from the String class
            //and the ith note is a String
            if (this.notes.get(i).contains(term))
            {
                numNotes++;
            }
        }
        return numNotes;
    }

    public ArrayList<String> getNotesContaining(String term)
    {
        ArrayList<String> found = new ArrayList<>();
        for (int i = 0; i < this.notes.size(); i++)
        {
            //contains() comes from the String class
            //and the ith note is a String
            if (this.notes.get(i).contains(term))
            {
                found.add(this.notes.get(i));
            }
        }
        return found;
    }

    public int countWords()
    {
        int numWords = 0;
        for (int i = 0; i < this.notes.size(); i++)
        {
            //to get the number of words per note, you need to
            //split a note up by spaces, and count the tokens
            String[] tokens = this.notes.get(i).split(" ");
            numWords += tokens.length;
        }
        return numWords;
    }
}
```