CSCI 1933 Lab 6

Midterm 1 Review

Rules

This lab is to be completed without use of a computer. **You must use only a pen and paper to complete these steps.**

This lab is to be done individually. If you complete the lab before the end of the session then you may be checked off by a TA. Due to the midterm this week, if you work on the assignment for the entire time and show your answers to a TA at the end of the lab then you will receive full credit. No portion of this lab can be checked off during office hours.

1 Class Design ★

In this step you must write the methods to complete a `Bus` class. Assume that the Passenger class has already been written.

- Each bus has a capacity. A default constructor should initialize the capacity to 40, and another constructor should allow the programmer to specify any positive capacity.
- An `addPassenger(Passenger p)` method should add the given `Passenger` object to the bus. Passengers can't be added if the bus is full. *Hint: what new member variable is needed?*
- A `numberOfPassengers()` method should return the current number of `Passengers` on board the bus.

2 Cloning an Array ★

A common operation on data in Java is "cloning". A clone is an identical but separate copy of an object. Note that clones will yield `true` when compared to each other with the `Arrays.equals()` method, but will yield `false` when compared with `==`.

Write a method `public static int[] clone(int[] inputArray)` that will return "clone" of the `inputArray`, and explain to the TA why your method satisfies the properties of cloning listed above.

3 Recursive Algorithms ★

Write a **recursive** method `public static int productDigits(int n)` to return the product of the digits in the integer `n`. For example, `productDigits(1552)` is 50.

4 Debugging ★

The following snippets of code all have at least one bug. Write down fixed code, and explain to a TA what the bugs are.

```
// Constructs a "Whatever" object
private int data;
public Whatever(double data) {
        data = data;
}
public static void setData(int newData) {
        data = newData;
}
```

```
// Computes the sum of the numbers the user enters
Scanner s = new Scanner(System.in);
int sum = 0;
while (s.hasNext()) {
        String data = s.next();
        if (data == "stop") {
                break;
        }
        sum += data;
}
System.out.println("The sum is:" + sum);
```

```
// Checks if an object is null
Object a, b; // Assume these could be anything
boolean b1 = a.equals(b);
boolean b2 = a.equals(null);
boolean b3 = (a == null && b == null) || a.equals(b);
boolean b4 = a == null || a.equals(b);
```

5 Code Comprehension ★

Give the output (`System.out`) of the following `main` method.

```
public class Whatever {
    private int data;

    public Whatever(int newData) {
        data = newData;
    }

    public void setData(int newData) {
        data = newData;
    }

    public int getData() {
        return data;
    }

    public static void doWhatever(Whatever w, int i, int d) {
        System.out.println("doWhatever(1): w: "+w.getData()+", i: "+i+"; d: "+d);
        w.setData(i);
        d = i;
        i = d;
        System.out.println("doWhatever(2): w: "+w.getData()+", i: "+i+"; d: "+d);
    }

    public static void main(String[] args) {
        Whatever w = new Whatever(1);
        int i = 2;
        double d = 3;
        System.out.println("main(1): w: "+w.getData()+", i: "+i+"; d: "+d);
        doWhatever(w, i, (int)d);
        System.out.println("main(2): w: "+w.getData()+", i: "+i+"; d: "+d);
        w = new Whatever(i);
        d = i / 4;
        System.out.println("main(3): w: "+w.getData()+", i: "+i+"; d: "+d);
    }
}
```

Gopher It

The following problems are optional extra practice.

Building a 2-D Array

Write a method `public static double[][] make2DArray(int row, int column)` to return an array of dimension $row \times column$ of doubles such that the values in each row start with the row index and go up by 1 for each column. For example, if $row = 4$ and $col = 3$, the array returned will look like this:

0.0	1.0	2.0
1.0	2.0	3.0
2.0	3.0	4.0
3.0	4.0	5.0

Least Common Multiple

Write a method `public static int leastCommonMultiple(int a, int b)` to return the least common multiple of two integers. The least common multiple of two positive integers a and b is defined as the smallest positive number that is evenly divisible by both a and b.

Polynomials

In this step you will write a class to represent a polynomial of the form:

$$ax^2 + bx + c$$

The class should be called `Polynomial`. It should have the following constructor and methods:

- `public Polynomial(double a, double b, double c)` – the constructor
- `public double getA()` – returns the value of `a`
- `public double getB()` – returns the value of `b`
- `public double getC()` – returns the value of `c`
- `public Polynomial add(Polynomial p)` – returns `p` + the current polynomial. To add polynomials, add coefficients of like terms; non-existent terms have a 0 coefficient
- `double evaluate(double x)` - returns the value of this polynomial evaluated at `x`