CSCI 1933 Lab 5

Midterm 1 Review Solutions

1 Class Design

```
public class Train {
	private Passenger[] passengers;
	private int size = 0;
	public Train() {
		passengers = new Passenger[100];
	}

	public Train(int capacity) {
		passengers = new Passenger[capacity];
	}

	public void addPassenger(Passenger p) {
		if (size < passengers.length) {
			passengers[size] = p;
			size++;
		}
	}

	public int numberOfPassengers() {
		return size;
	}

	public int availableCapacity() {
		return passengers.length - size;
	}
}
```

2 Cloning an Array

```
public static int[] clone(int[] inputArray) {
        int[] ret = new int[inputArray.length];
        for (int i = 0; i < ret.length; i++) {
                ret[i] = inputArray[i];
        }
        return ret;
}
```

3 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(int data) {
        this.data = data;
}
public void setData(int newData) {
        data = newData;
}
```

1. The type of the parameter in the constructor and member variable must match. Changed `double data` to `int data`.
2. Changed `data = data` in the constructor to `this.data = data`. `this.data` refers to the member variable `data` and not the local variable `data`.
3. Removed `static` from the setData method header.

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

1. Changed `data == "stop"` to `data.equals("stop")`. Always use .equals() when comparing Objects and == when comparing primitive types.
2. Changed `sum += data` to `sum += Integer.parseInt(data)`. You are unable to add a String to a integer, so data was parsed into an integer.

```
// Cross out the line with the error.
// What are the values of b1, b2, b3, and b4?
Object a = new Object();
Object b = new Object();
Object c = null;
boolean b1 = a.equals(b); //false
boolean b2 = b.equals(c); //false
//boolean b3 = c.equals(a); null pointer exception
boolean b4 = (a == null || c == null) || c.equals(b); //true
```

4 Code Comprehension

```
main(1): w: 1, i: 2; d: 3.0
doWhatever(1): w: 1, i: 2; d: 3
doWhatever(2): w: 2, i: 2; d: 2
main(2): w: 2, i: 2; d: 3.0
main(3): w: 2, i: 2; d: 0.0
```

5 Recursive Algorithms

```
public static int productDigits(int n) {
      if (n < 10) {
            return n;
      }
      return (n%10) * productDigits(n/10);
}
```

6 Building a 2-D Array

```
public static double[][] make2DArray(int row, int column) {
       double[][] arr = new double[row][column];
       for (int i = 0; i < arr.length; i++) {
             for (int j = 0; j <arr[i].length; j++) {
                   arr[i][j] = i + j;
             }
       }
       return arr;
}
```

7 Least Common Multiple

```
public static int leastCommonMultiple(int a, int b) {
      int lcm = Math.max(a,b);
      while (true) {
            if (lcm % a == 0 && lcm % b == 0) {
                  return lcm;
            }
            lcm++;
      }
```

8 Polynomials

```
public class Polynomial {
	private double a;
	private double b;
	private double c;

	public Polynomial(double a, double b, double c) {
		this.a = a;
		this.b = b;
		this.c = c;
	}

	public double getA() {
		return a;
	}

	public double getB() {
		return b;
	}

	public double getC() {
		return c;
	}

	public void add(Polynomial p) {
		this.a += p.a;
		this.b += p.b;
		this.c += p.c;
	}

	public double evaluate(double x) {
		return a*(x*x) + b*x + c;
	}

}
```