CS2020: Data Structures and Algorithms (Accelerated)

Discussion Group Problems for Week 3

For: Jan. 27–Jan. 31

Problem 1. (**Bad Java**)
Each of the three parts below asks for a short Java program that demonstrates the use of a particular Java construct. In each case, explain in comments what is wrong (or right) about your program.

Problem 1.a. Write an example class (or classes) that uses the modifier `private` incorrectly (i.e., the program will not compile as is, but would compile if `private` were changed to `public`).

Problem 1.b. Write an example class that uses a `static` variable incorrectly (i.e., the program will not compile due to the way in which `static` is used).

Problem 1.c. Write an example program that uses an `interface` incorrectly.

Problem 2. (Drinking Coke)
For this problem, consider the *Drink* class in Figure 1 and the *Coke* class in Figure 2.

Problem 2.a. What goes wrong if you try to create a new class Sprite as follows:

```
public class Sprite extends Drink {

        Sprite(int temperature, String type){
                System.out.println("Sprite constructor!");
        }
}
```

Problem 2.b. Consider the Coke class defined in Figure 2. Notice that the getADrink() method returns a Drink in the parent class (Drink), and returns a Coke in the child class (Coke). Is that ok? Does this compile?

Problem 2.c. Consider again the Coke class. Would it be ok to change the `isGood` method to private in the parent class Drink? What if you changed the `isGood` method to protected? What if you changed it in the Coke class, setting it to private for Coke?

Problem 2.d. Consider again the Coke class. There are several interesting and potentially problematic things about this class. What is going on here? What does this code output?

```
public class Drink {

        protected String name;
        private int temperature;
        private String color;

        public Drink(String n, int temp, String c){
                System.out.println("Drink constructor.");
                name = n;
                temperature = temp;
                color = c;
                checkSoda();
        }

        public void checkSoda(){
                System.out.println("Drinking a drink, with name: " + name);
        }

        public Drink getADrink(){
                System.out.println("Getting a drink.");
                return new Drink("Unknown", 10, "Opaque");
        }

        public static boolean isGood(int temp, String color){
                if (color == "black"){
                        System.out.println("It's good! ");
                        return true;
                } else{
                        System.out.println("Ugh! ");
                        return false;
                }
        }

}
```

Figure 1: Drink Class

```
public class Coke extends Drink {

        String cokeType;

        Coke(int temp, String c){
                super("Coke", temp, "Black");
                System.out.println("Done with super; on to myself.");
                name = "Coke";
                cokeType = c;
                checkSoda();
                isGood(3, "yellow");
                System.out.println("Coke constructor done!");
        }

        public static boolean isGood(int temp, double a){
                System.out.println("Coke is good!");
                return true;
        }

        public void checkSoda(){
                System.out.println("Drinking a coke.");
                System.out.println("Coketype = " + cokeType);
        }

        public Coke getADrink(){
                System.out.println("Getting a coke.");
                return new Coke(5, "Black");
        }

        public static boolean isGood(int temp, String color){
                System.out.println("Coke is always good!");
                return true;
        }

        static public void main(String[] args){
                Coke soda = new Coke(10, "black");
        }
}
```

Figure 2: Coke Class

Problem 3. (Integers) For this problem, see the code in Figure 3. What does this program output? Why? (Note the bad programming style, where the class variable `m_int` is public and accessed from outside the class. Also note the definition of `myInteger` as a class inside class `IntegerExamination`.)

```
public class IntegerExamination {

        static class myInteger {
                public int m_int;

                myInteger(int k){m_int = k;}

                public String toString(){return Integer.toString(m_int);};
        }

        public static void main(String[] args){
                // Initialize integers
                int i = 7;
                myInteger j = new myInteger(7);
                myInteger k = new myInteger(7);

                // Add one to each integer
                addOne(7);
                myIntAddOne(j);
                myOtherIntAddOne(k);

                // Print the output
                System.out.println(i);
                System.out.println(j);
                System.out.println(k);
        }

        static public void addOne(int i){
                i = i+1;
        }

        static public void myIntAddOne(myInteger i){
                i.m_int = i.m_int + 1;
        }

        static public void myOtherIntAddOne(myInteger i){
                i = new myInteger(i.m_int + 1);
        }


}
```

Figure 3: Testing Integers