COMP249 Midterm Review

Weiliang Xie

Winter 2020

Answer the questions in the spaces provided. If you run out of room for an answer, continue on the back of the page.

Name and section: ______________________________

1 True or False

1. In Java, a function with code that might throw an exception that is not caught in said function must declare it by using the **throws**.

2. If a parent class has a method **public int func()**, a child class can reduce its access modifier (e.g. **protected int func()**).

3. A child class automatically overrides every method of its parent class so long as they have the same signature.

4. The **super()** constructor can always be called in a child constructor as long as it is called first.

5. In Java, a class can both inherit from multiple classes and implement multiple interfaces.

6. Java allows you to declare a variable of an abstract class. For example, if **A** is an abstract class, we can declare: **A a1;**

7. All types of exceptions must follow the **Catch or Declare** rule.

8. When using a **PrintWriter**, there is no way to append to a file. It is always overwritten with new data.

9. A grand-child class can easily call its grand-parent's constructor by calling **super().super()**.

10. The order in which catch blocks are written is unimportant because Java will automatically match the thrown exception which its corresponding catch block.

11. The super constructor is always called, either explicitly or implicitly.

12. A derived class with an overridden method can add exceptions to the list of thrown exceptions.

13. The **finally** block is executed after an exception is successfully caught.

2 Output Questions

1. Consider the following classes and determine the output or the error:

```java
public class Parent {
    public Parent() { this.greet2(); }
    public static void greet() { System.out.println("Hello!"); }
    public void greet2() { System.out.println("Hey!"); }
}
```

```java
public class Child extends Parent {
    public Child() {
        this.greet();
        this.greet3();
    }
    public static void greet() { System.out.println("Bonjour!"); }
    public void greet2() { System.out.println("Salut!"); }
    public void greet3() { System.out.println("Allo!"); }
}
```

```java
public class GrandChild extends Child {
    public GrandChild() { this.greet3(); }
    public void greet3() { System.out.println("Ave!"); }
}
```

(a) `Child c = new Child();`

(b) `Parent p = new Child();`

(c) `GrandChild gc = new GrandChild();`

(d)
```
c.greet();
p.greet();
gc.greet();
```

(e)
```
c.greet2();
p.greet2();
gc.greet2();
```

(f)
```
System.out.println(c.getClass() == p.getClass());
```

(g)
```
GrandChild gc2 = new Child();
```

(h)
```
Child c2 = new GrandChild();
GrandChild gc3 = (GrandChild) c2;
gc3.greet3();
```

2. Consider this function:

```
public static double whatAmI(double d, int n) {
    if(n == 0)
        return 1;
    return d * whatAmI(d, n - 1);
}
```

(a) What is the output of:

```
whatAmI(4, 3);
```

(Show all steps).

(b) What does this function do?

3 UML

1. You've had enough of your messy room and decide to start cleaning. After all, spring is almost upon us and no better way to forget the horrors of winter than to wipe the slate clean and start the season off right. However, instead of *actually* cleaning, you decide to "practice" programming instead. Before you start programming, you must always plan! There are all sorts of different things in your room, so a diagram could maybe help you organize the hierarchy of your possessions. Draw a simple UML diagram to classify objects in your room. (Use these words: Electronics, Phone, Looseleaf Paper, Office Supplies, Possessions, Phone Charger, Clothes, Pencil Case)

(a) **Electronics** has the attribute **price** and **Phone** has attributes **model** and **size**. Assuming **Electronics** has a parameterized constructor, write a parameterized constructor for **Phone**. (Bonus points for elegant use of what is given).

(b) Write an **equals** method for a **Phone**.

4 Programming Questions

1. You're at your new internship and you finally get your first assignment. You're excited, but also scared. You don't want to mess up after all! Though your task seems simple, good programming practices are expected (You can skip commenting for this test, but don't forget it in real life!) As an exercise, they want you to populate a data file with random integer values. You are to read from this file and export it into a CSV (Comma Separated Values) file. The data file is named 'data.dat' and the CSV is named 'data.csv'. (Bonus points if the last element in the csv doesn't have a comma).

```
Ex: 1234 3456 5678 ... 9847 => 1234,3456,5678,...,9847
```

```
import java.util.Random;
import java.util.Scanner;
import java.io.*;

public class ConvertToCSV {
    public static void main(String[] args) {
        populateFile();
        convertAndExport();
    }
}
```