

FPP Quiz #2

```
1. class MyClass {
    static int amount = 1;
    public static void main(String[] args) {
        System.out.println(this.amount);
    }
}
```

What happens when the program is compiled/run?

- a. Compiler error
- b. Runtime error
- c. Outputs 1 to the console
- d. Outputs 0 to the console

```
2. class MyClass extends MySuperClass {
    public static void main(String[] args) {
        MySuperClass cl = new MyClass();
        System.out.println(cl.getType());
    }

    public int getType() {
        return 3;
    }
}

class MySuperClass {
    public int getType() {
        return 2;
    }
}
```

What happens when the program is compiled/run?

- a. Compiler error
- b. Runtime error
- c. Outputs 2 to the console
- d. Outputs 3 to the console

```

3. class MyClass extends MySuperClass {
    public static void main(String[] args) {
        MySuperClass cl = new MySuperClass();
        System.out.println(cl.getType());
    }

    public int getType() {
        return 3;
    }
}
class MySuperClass {
    public int getType() {
        return 2;
    }
}

```

What happens when the program is compiled/run?

- a. Compiler error
- b. Runtime error
- c. Outputs 2 to the console
- d. Outputs 3 to the console

```

4. class MyClass {
    public static void main(String[] args) {
        new MyClass();
    }
    MyClass() {
        System.out.println(value);
    }
    class MyInnerClass {
        private int value = 3;
    }
}

```

What happens when the program is compiled/run?

- a. Compiler error
- b. Runtime error
- c. Outputs 3 to the console

```

5. class MyClass {
    public static void main(String[] args) {
        new MyClass();
    }
    private int value = 3;
    MyClass() {
        MyInnerClass c = new MyInnerClass();
        System.out.println(c.compute());
    }
    class MyInnerClass {
        private int compute() {

```

```

        return value;
    }
}

```

What happens when the program is compiled/run?

- a. Compiler error
- b. Runtime error
- c. Outputs 3 to the console

```

6. class MyClass {
    public static void main(String[] args) {
        new MyClass();
    }
    private int value = 3;
    MyClass() {
        MyNestedClass c = new MyNestedClass();
        System.out.println(c.compute());
    }
    static class MyNestedClass {
        private int compute() {
            return value;
        }
    }
}

```

What happens when the program is compiled/run?

- a. Compiler error
- b. Runtime error
- c. Outputs 3 to the console

```

7. class MyClass {
    public static void main(String[] args) {
        MyClass cl = new MySubClass();
        System.out.println(cl.compute());
    }
    private int value = 3;
    public int compute() {
        return value;
    }
}

class MySubClass extends MyClass {
    MySubClass() {}
    public int compute() {
        return (new MyInnerClass()).compute();
    }
    class MyInnerClass {
        private int compute() {
            return value*value;
        }
    }
}

```

```
}
```

What happens when the program is compiled/run?

- a. Compiler error
- b. Runtime error
- c. Outputs 3 to the console

```
8. class TheClass {
    TheClass() {
        TheSubclass sub = new TheSubclass();
        System.out.println("The Class constructor");
    }
    public static void main(String[] args) {
        new TheSubclass();
    }
}
class TheSubclass extends TheClass {
    TheSubclass() {
        System.out.println("The Subclass constructor");
    }
}
```

What happens when the program is compiled/run?

- a. Compiler error
- b. Outputs "The Subclass constructor" followed by "The Class constructor".
- c. Outputs "The Class constructor" followed by "The Subclass constructor".
- d. Stack overflow exception

```
9. public class TheClass {
    private int value =5;
    public int getValue() {
        return value;
    }
    private static TheSubclass cls;
    public TheClass() {
    }
    public class TheInnerClass {
        public int evaluate() {
            return value;
        }
    }
    void run() {
        TheClass.TheInnerClass inner =
            cls.new TheSubInner();
        System.out.println(inner.evaluate());
    }
    public static void main(String[] args) {
        cls = new TheSubclass();
    }
}
```

```

        clsub.run();
    }
}
class TheSubclass extends TheClass {
    private int value = getValue()+1;
    public class TheSubInner extends TheClass.TheInnerClass {
        public int evaluate() {
            return super.evaluate()+value;
        }
    }
    TheSubclass() {}
}

```

What happens when the code is compiled/run?

- Outputs 5 to console
- Outputs 6 to console
- Outputs 11 to console
- Runs, but outputs nothing to the console
- Compiler error
- Runtime exception

10. Consider the following code:

```

public class Main {
    String managerInfo = (new Employee("Manager", 200000) {
        //a percentage increase applied to salary
        double bonus;
        {
            bonus = .05;
        }
        int computeSalaryWithBonus() {
            return (int)((1+bonus) * getSalary());
        }
        @Override
        public String toString() {
            return "name : " + getName() + "\n"
                + " base salary : " + getSalary() + "\n"
                + " bonus : " + bonus + "\n"
                + " actual salary : " + computeSalaryWithBonus()
        }
    }).toString();

    public static void main(String[] args) {
        Main m = new Main();
        System.out.println(m.managerInfo + "\n\n");
    }
}

```

What happens when the main method is compiled and run?

- There is no output
- Compiler error
- Runtime exception
- A string is printed to the console giving information about a Manager object

(continued on next page)

```
public class Employee {  
    private String name;  
    private int salary;  
    public Employee(String n, int s) {  
        this.name = n;  
        this.salary = s;  
    }  
    public String getName() {  
        return name;  
    }  
    public int getSalary() {  
        return salary;  
    }  
}
```

