COMP 333
Fall 2024

Class-based Inheritance, Subtyping, and Virtual Dispatch

1.) Consider the Java class/interface definitions and snippets below. What is the output of the snippet?

1.a)

```
public class Base {
  public void method() { System.out.println("base"); }
}
public class Sub1 extends Base {
  public void method() { System.out.println("sub1"); }
}
public class Sub2 extends Base {}

// Begin program
Base a = new Base(); a.method();
Base b = new Sub1(); b.method();
Base c = new Sub2(); c.method();
Sub1 d = new Sub1(); d.method();
Sub2 e = new Sub2(); e.method();
```

1.b)

```
public interface MyInterface {
  public void doSomething();
}
public class Foo implements MyInterface {
  public void doSomething() { System.out.println("Foo"); }
}
public class Bar implements MyInterface {
  public void doSomething() { System.out.println("Bar"); }
}

MyInterface a = new Foo(); a.doSomething();
Foo b = new Foo(); b.doSomething();
MyInterface c = new Bar(); c.doSomething();
Bar d = new Bar(); d.doSomething();
```

2.) Consider the following Java snippet:

```
boolean b = (randomBoolean()) ? true : false;
if (b) {
  System.out.println("foo");
} else {
  System.out.println("bar");
}
```

This code can be rewritten to entirely avoid the `if`, by using virtual dispatch instead. This code is partially rewritten below, where `...` are different `Conditional` expressions:

```
Conditional c = (randomBoolean()) ? ... : ...;
c.operation();
```

Write the remaining code necessary to make the above snippet operate the same as with the `if`. Include what the two uses of `...` above will need to be. As a hint, you will need classes corresponding to `true` and `false`.