AP CS: Lesson 21: Polymorphism

Name: _____ Period: ___

Inheritance: (BJP* Chapter 9.3)

Polymorphism: Ability for the same code to be used with different types of objects and behave differently with each.

So a variable of type *T* can hold an object of any subclass of *T*.

Syntax:

```
Superclass name = new Subclass ();
```

Example:

```
Employee ed = new Lawyer();
```

Polymorphism Rules:

- You can call any & only methods from the Type (Employee) class (i.e. `getSallary()` & `getVacationsForm()`)
- Method calls are always determined by the type of the actual object, not the type of the object reference, so when a method is called on ed, it behaves as a Lawyer.

```
System.out.println(ed.getSalary());          // 50000.0
System.out.println(ed.getVacationForm());    // pink
```

You can pass any subtype of a parameter's type.

```
public class EmployeeMain {
    public static void main(String[] args) {
        Lawyer lisa = new Lawyer();
        Secretary steve = new Secretary();
        printInfo(lisa);
        printInfo(steve);
    }
    public static void printInfo(Employee empl) {
        System.out.println("salary: " + empl.getSalary());
        System.out.println("v.days: " + empl.getVacationDays());
        System.out.println("v.form: " + empl.getVacationForm());
        System.out.println();
    }
}
```

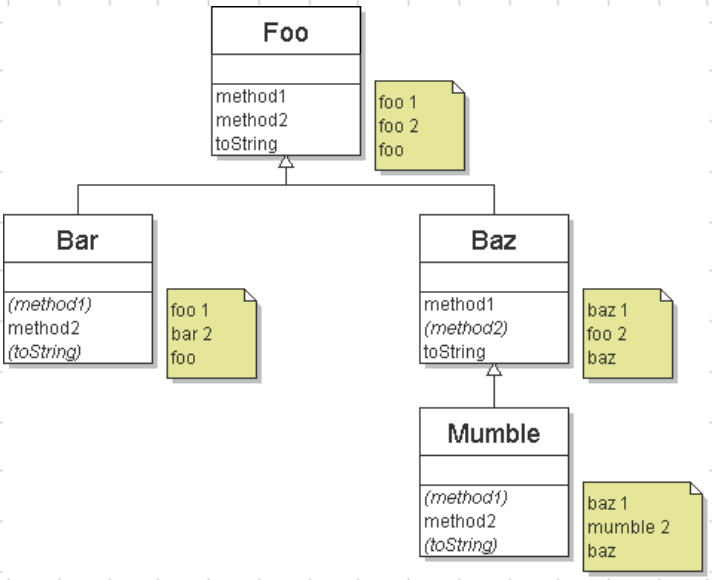
Arrays of superclass types can store any subtype as elements.

```
public class EmployeeMain2 {
    public static void main(String[] args) {
        Employee[] e = { new Lawyer(),   new Secretary(),
                         new Marketer(), new LegalSecretary() };
        for (int i = 0; i < e.length; i++) {
            System.out.println("salary: " + e[i].getSalary());
            System.out.println("v.days: " + e[i].getVacationDays());
            System.out.println();
        }
    }
}
```

AP CS: Lesson 21: Polymorphism

Polymorphism Problem Example:

```
public class Foo {
    public void method1() {
        System.out.println("foo 1");
    }
    public void method2() {
        System.out.println("foo 2");
    }
    public String toString() {
        return "foo";
    }
}
public class Bar extends Foo {
    public void method2() {
        System.out.println("bar 2");
    }
}
public class Foo {
    public void method1() {
        System.out.println("foo 1");
    }
    public void method2() {
        System.out.println("foo 2");
    }
    public String toString() {
        return "foo";
    }
}
public class Bar extends Foo {
    public void method2() {
        System.out.println("bar 2");
    }
}
```



method	Foo	Bar	Baz	Mumble
method1				
method2				
toString				

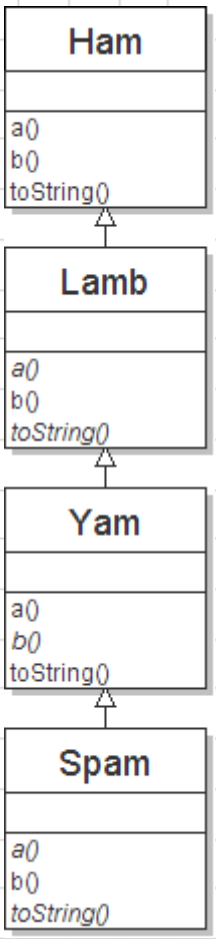
- **What would be the output of the following client code?**

```
Ham[] food = {new Lamb(), new Ham(), new Spam(), new Yam()};
for (int i = 0; i < food.length; i++) {
    System.out.println(food[i]);
    food[i].a();
    System.out.println();      // to end the line of output
    food[i].b();
    System.out.println();      // to end the line of output
    System.out.println();
}
```

AP CS: Lesson 21: Polymorphism

Another Example calling other methods:

```
public class Lamb extends Ham {
    public void b() {
        System.out.print("Lamb b   ");
    }
}
public class Ham {
    public void a() {
        System.out.print("Ham a   ");
        b();
    }
    public void b() {
        System.out.print("Ham b   ");
    }
    public String toString() {
        return "Ham";
    }
}
public class Spam extends Yam {
    public void b() {
        System.out.print("Spam b   ");
    }
}
public class Yam extends Lamb {
    public void a() {
        System.out.print("Yam a   ");
        super.a();
    }
    public String toString() {
        return "Yam";
    }
}
```



What would be the output of the following client code?

```
Ham[] food = {new Lamb(), new Ham(), new Spam(), new Yam()};
for (int i = 0; i < food.length; i++) {
    System.out.println(food[i]);
    food[i].a();
    System.out.println();      // to end the line of output
    food[i].b();
    System.out.println();      // to end the line of output
    System.out.println();
}
```

method	Ham	Lamb	Yam	Spam
a				
b				
toString				

AP CS: Lesson 21: Polymorphism

Casting References:

- A variable can only call that type's methods, not the subtype's.

```
Employee ed = new Lawyer();
int hours = ed.getHours();  // ok; it's in Employee
ed.sue(); //** compiler error **
```

  - The compiler's reasoning is, variable `ed` could store any kind of employee, and not all kinds know how to `sue`
  - To use `Lawyer` methods on `ed`, we can type-cast it.

```
Lawyer theRealEd = (Lawyer) ed;
theRealEd.sue();                    // ok
((Lawyer) ed).sue();                // shorter version
```

- The code crashes if you cast an object too far down the tree.

```
Employee eric = new Secretary();
((Secretary) eric).takeDictation("hi");     // ok
((LegalSecretary) eric).fileLegalBriefs();  // ** exception ** Secretary object
                                               doesn't know how to file briefs)
```

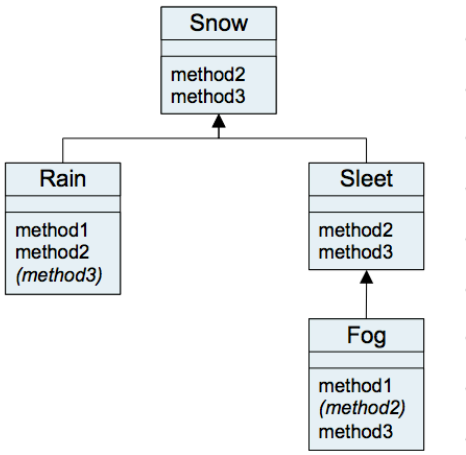
- You can cast only up and down the tree, not sideways.

```
Lawyer linda = new Lawyer();
((Secretary) linda).takeDictation("hi");  // ** error **
```

- Casting doesn't actually change the object's behavior. It just gets the code to compile/run.

```
((Employee) linda).getVacationForm()    // pink (Lawyer's)
```

```
public class Snow {
    public void method2() {
        System.out.println("Snow 2");
    }
    public void method3() {
        System.out.println("Snow 3");
    }
}
public class Rain extends Snow {
    public void method1() {
        System.out.println("Rain 1");
    }
    public void method2() {
        System.out.println("Rain 2");
    }
}
public class Sleet extends Snow {
    public void method2() {
        System.out.println("Sleet 2");
        super.method2();
        method3();
    }
    public void method3() {
        System.out.println("Sleet 3");
    }
}
public class Fog extends Sleet {
    public void method1() {
        System.out.println("Fog 1");
    }
    public void method3() {
        System.out.println("Fog 3");
    }
}
```



What happens when the following examples are executed?

- Example 1:

```
Snow var1 = new Sleet();
var1.method2();
```

- Example 1b:

```
Snow var1 = new Fog();
var1.method2();
```

- Example 2:

```
Snow var2 = new Rain();
var2.method1();
```

- Example 3:

```
Snow var3 = new Rain();
((Sleet) var3).method3();
```

method	Snow	Rain	Sleet	Fog
method1				
method2				
method3				