COMP2396B Tutorial 5

Inheritance, overloading, abstract classes

1. When a method is called on an object reference, the _________________ of the method for that object type will be called —In other words, the ______ one in the inheritance tree wins!

2. It is possible to call an overridden method of the superclass using the keyword _____.

3. A superclass can choose whether or not it wants a subclass to inherit a particular member by the access level assigned to that particular member .

—_____ members are inherited.

—______ members are not inherited.

4. Method overloading is nothing more than having 2 or more methods with the same ____ but different ___________ —It has nothing to do with inheritance and polymorphism.

—For overloaded methods

—Argument lists ____ be different.

—Return types ___ be different.

—Access levels ___ be different.

5. An abstract class can be used as a _________ (e.g., using it as a polymorphic argument or return type, or to make a polymorphic array).

6. Besides classes, ______ can also be marked as abstract.

7. An abstract class means the class must be _______, whereas an abstract method means the method must be ________.

8. An abstract method has no _______, just ends with a semicolon.

9. A class must be marked as abstract if it has ________ abstract method. It is illegal to have an abstract method in a non-abstract class!

— An abstract class, on the other hand, can have ________ abstract and non-abstract methods.

10. A concrete class in the inheritance tree must _________ the abstract methods from its superclass.

Inheritance

What is the output of the following program? It has no compile-time or run-time error.

```java
class A {
	int n;
	public A() {
		this.n = 1;
	}
	public void calculate() {
		this.n = 4 * this.n;
	}
	public void print() {
		calculate();
		System.out.println("In A: " + this.n);
	}
}

class B extends A {
	int n;
	public B() {
		this.n = 10;
	}
	public void calculate() {
		this.n = 4 * super.n;
	}
	public void print() {
		this.calculate();
		System.out.println("In B: " + this.n);
	}
}

class C extends B {
	int n;
	public C() {
		this.n = 100;
	}
	public void calculate() {
		this.n = 4 * this.n;
	}
	public void print() {
		super.print();
		System.out.println("In C: " + this.n);
	}
}

public class Main {
	public static void main(String[] args) {
		A x1 = new A();
		x1.print();

		B x2 = new B();
		x2.print();

		C x3 = new C();
		x3.print();
	}
}
```

Answer:

1. When a method is called on an object reference, the most specific version of the method for that object type will be called —In other words, the lowest one in the inheritance tree wins!

2. It is possible to call an overridden method of the superclass using the keyword super.

—Example

```
public class Animal {
   public void roam() {
      System.out.println("Animal roams");
   }
   // ...
}
```

```
public class Canine extends Animal {
   public void roam() {
      super.roam(); // roam() in Animal class is called
      System.out.println("Canine roams");
   }
}
```

```
public class SuperTestDrive {
   public static void main(String[] args) {
      Canine c = new Canine();
      c.roam();
   }
}
```

—Sample output

```
Animal roams
Canine roams
```

3. A superclass can choose whether or not it wants a subclass to inherit a particular member by the access level assigned to that particular member .

—public members are inherited.

—private members are not inherited.

	Access level	Access modifier	Class	Package	Sub-class	World
more restrictive	private	private	Y	N	N	N
	default	(none)	Y	Y	N	N
	protected	protected	Y	Y	Y	N
less restrictive	public	public	Y	Y	Y	Y

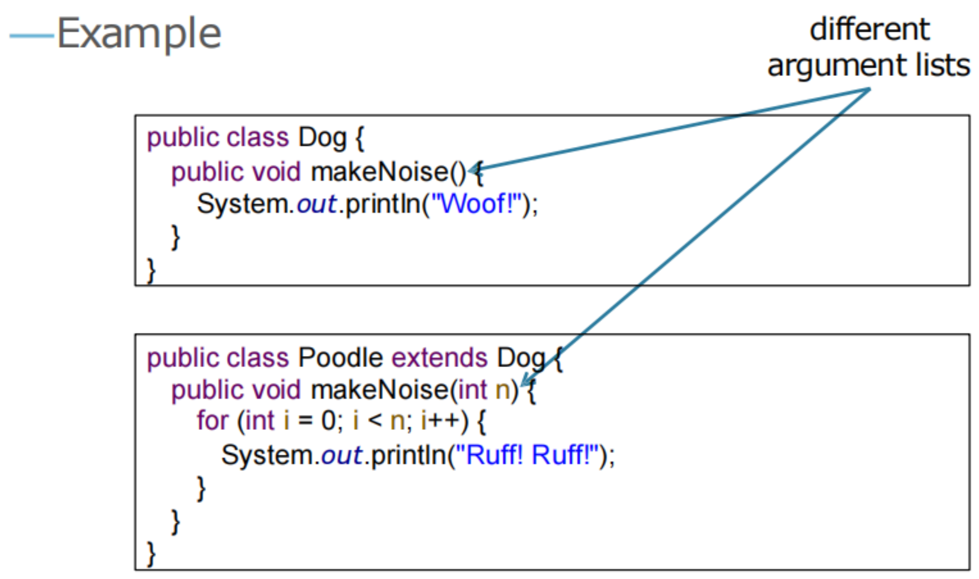
4. Method overloading is nothing more than having 2 or more methods with the same name but different argument lists —It has nothing to do with inheritance and polymorphism.

—For overloaded methods

—Argument lists must be different.

—Return types can be different.

—Access levels can be different.



—Example

```
public class Dog {
  public void makeNoise() {
    System.out.println("Woof!");
  }
}
```

```
public class Poodle extends Dog {
  public void makeNoise(int n) {
    for (int i = 0; i < n; i++) {
      System.out.println("Ruff! Ruff!");
    }
  }
}
```

5. An abstract class can be used as a reference type (e.g., using it as a polymorphic argument or return type, or to make a polymorphic array).

6. Besides classes, methods can also be marked as abstract.

7. An abstract class means the class must be extended, whereas an abstract method means the method must be overridden.

8. An abstract method has no method body, just ends with a semicolon.

```
public abstract void eat();
```

9. A class must be marked as abstract if it has at least one abstract method. It is illegal to have an abstract method in a non-abstract class!

— An abstract class, on the other hand, can have either or both abstract and non-abstract methods.

10. A concrete class in the inheritance tree must implement all the abstract methods from its superclass.