Inheritance and Extended Classes in Java
The Class Hierarchy

• A class that is derived from another class is called a subclass, extended class or child class.
• If A is a subclass of B, then B is called the parent class, base class or superclass.
• These are often called “is-a” relationships.
  • E.g., Cat “is-a” Animal, Car “is-a” Vehicle, etc.
• The Object is the ultimate superclass because everything “is-a” object.
• A subclass inherits fields and methods from its superclass.
The Class Hierarchy

• If A is a subclass of B, we say that A extends B.
• Except Object, every class in Java has one and only one superclass.
  • This is called single inheritance.
• If we don’t explicitly say that a class extends another class, then it is implicitly understood that it extends Object.
• A class can be derived from a class that is derived from
  ... a class that is derived from
  ... a class that is derived from
  ...
  ...
  ... Object!
The Class Hierarchy

- If A is a subclass of B, we say that **A extends B**.
- Except Object, every class in Java has **one and only one** superclass.
  - This is called *single inheritance*.
- If we don’t explicitly say that a class extends another class, then it is implicitly understood that it extends Object.
- A class can be derived from a class that is derived from
  - ... a class that is derived from
  - ... a class that is derived from
  - ... Object!

  A class is said to *descend* from all the superclasses in this chain.
  - Every class *descends* from the Object class!
  - Hence ... class hierarchy!
The Class Hierarchy

• If A is a subclass of B, we say that A extends B.
• Except Object, every class in Java has one and only one superclass.
  • This is called single inheritance.
• Inheritance is a powerful concept.
  • If there already exists a class that is similar to what you want ... you can derive your class from that.
• A subclass inherits all the members of its superclass.

... all members?
The Class Hierarchy

• If A is a subclass of B, we say that A extends B.
• Except Object, every class in Java has one and only one superclass.
  • This is called single inheritance.
• Inheritance is a powerful concept.
  • If there already exists a class that is similar to what you want ... you can derive your class from that.
• A subclass inherits all the members of its superclass.

What about constructors?

• Constructors are not NOT inherited.
• But the constructor of the superclass can be invoked from a subclass.
Declaring an extended class

• If A is a subclass of B, we say that A extends B.

```java
public class Organism {
    private String species;
    private double age;
    private double health; // say, a score between 0 - 10, 0 being "death"

    public Organism(String species) { this.species = species; }
}

public class Animal extends Organism { ... }
```
Constructors of an extended class

```java
public class Organism {
    private String species;
    private double age;
    private double health;  // say, a score between 0 - 10, 0 being "death"

    public Organism(String species) {
        this.species = species;
    }
}

public class Animal extends Organism {
    public Animal(String species) {
        super(species);
    }
}
```

We can invoke a constructor of the superclass using the "super" keyword.

**Note:**
Inherited methods can also be called via "super".
public class Organism {
    private String species;
    private double age;
    private double health; // say, a score between 0 - 10, 0 being "death"

    public Organism(String species) {
        this.species = species; }
}

public class Animal extends Organism {

    public Animal(String species) {
        super (species);
    }

    public static void main(String[] args) {
        Animal ant = new Animal("ant");
        ant.setHealth(10);
    }
}

Inherited method
Animal doesn’t have a “health” field. So ... is this valid?
Overriding inherited methods

• Sometimes, a subclass may need to do a certain task in a slightly different way from its superclass. This is called **overriding** an inherited method.

```java
class Phoenix extends Animal {

    private double lastFireUp;

    public Phoenix(String species) {
        super(species);
    }

    @Override
    public double getAge() {
        return super.getAge() - lastFireUp;
    }
}
```

Overridden method

Overrides the `getAge()` method in ... ?
Overriding versus Overloading

• **Overloading**
  • Method overloading is having two or more methods in the same class with the same name but different arguments.
  • *E.g.*, println

• **Overriding**
  • This is when two methods have the same name, and the same arguments
  • But different implementations
  • One implementation is in a parent class, and the other is in a derived class.
Covariance

• Overriding a method is not just about keeping the same method name!
  • The whole method signature must be the same as the method signature in the superclass.

• With one tiny modification ...

• The return type of an overriding method in a subclass can be a subclass of the return type of the original method in the superclass.
Covariance

• Overriding a method is not just about keeping the same method name!
  • The whole method signature must be the same as the method signature in the superclass.

• With one tiny modification ...

• The return type of an overriding method in a subclass can be a subclass of the return type of the original method in the superclass.
Covariance

- The return type of an overriding method in a subclass can be a subclass of the return type of the original method in the superclass.
- Why?
  - The *inherited method* is in a subclass, i.e., it is in a class that is *narrower* than the superclass. (Remember, Object is the most general class, and then things starts getting more and more specific.)
  - So ... if the subclass is *more specific*, then the methods should be allowed to return something more specific!
Why is there only one superclass?

- If A is a subclass of B, we say that A extends B.
- Except Object, every class in Java has one and only one superclass.
  - This is called single inheritance.
- Why can’t there be multiple inheritance?
  - i.e., why can’t a class extend multiple superclasses?
  - The “diamond” problem
Overriding vs Hiding

- An instance method in a subclass with the same signature and return type as an instance method in the superclass *overrides* the superclass's method.
- This allows a class to inherit from a superclass whose behavior is *similar*.
  - Otherwise, even for slight changes, we would have to write the whole class again!
- If a subclass defines a static method with the same signature as a static method in the superclass, then the method in the subclass *hides* the one in the superclass.

| The version of the overridden instance method that gets invoked is the one in the subclass. | The version of the hidden static method that gets invoked depends on whether it is invoked from the superclass or the subclass. |
public class Animal {
    public static void testClassMethod() {
        System.out.println("The static method in Animal");
    }
    public void testInstanceMethod() {
        System.out.println("The instance method in Animal");
    }
}

public class Cat extends Animal {
    public static void testClassMethod() {
        System.out.println("The static method in Cat");
    }
    public void testInstanceMethod() {
        System.out.println("The instance method in Cat");
    }
    public static void main(String[] args) {
        Cat myCat = new Cat();
        Animal myAnimal = myCat;
        Animal.testClassMethod();
        myAnimal.testInstanceMethod();
    }
}