Why inheritance

- Frequently, a class is merely a modification of another class. In this way, there is minimal repetition of the same code.
- Localization of code:
  - Fixing a bug in the base class automatically fixes it in the subclasses.
  - Adding functionality in the base class automatically adds it in the subclasses.
  - Less chances of different (and inconsistent) implementations of the same operation.

Inheritance in real Life

- A new design created by the modification of an already existing design:
  - The new design consists of only the changes or additions from the base design.
- CoolPhoneBook inherits PhoneBook:
  - Add mail address and cell number.

Example of inheritance tree

Inheritance terminology

- Class one above:
  - Parent class
- Class one below:
  - Child class
- Class one or more above:
  - Superclass, Ancestor class, Base class
- Class one or more below:
  - Subclass, Descendent class
Messy example (cont’d)

```java
electric2 = c; // NO Downcast
ec2 = (ElectricCar) cc; // 5, OK
ec2. recharge(); // OK
```

Avoid wrong down-casting

- Use the `instanceof` operator

```java
Car c = new Car();
ElectricCar ec;
if (c instanceof ElectricCar ){
   ec = (ElectricCar) c;
   ec.recharge();
}
```

Upcast to Object

- Each class is either directly or indirectly a subclass of `Object`
- It is always possible to upcast any instance to `Object` type (see `Collection`)

```java
AnyClass foo = new AnyClass();
Object obj;
obj = foo;
```

Abstract classes
Java interface

- An interface is a special type of “class” where methods and attributes are implicitly public
  - Attributes are implicitly static and final
  - Methods are implicitly abstract (no body)

- Cannot be instantiated (no new)
- Can be used to define reference

Example (cont’d)

```java
public interface Comparable {
    void isEqual(String s);
}

class Car implements Comparable {
    private String licencePlate;
    public void isEqual(String s) {
        return licencePlate.equals(s);
    }
}

class Employee implements Comparable {
    private String name;
    public void isEqual(String s) {
        return name.equals(s);
    }
}
```

Example

```java
public class Foo {
    private Comparable objects[];
    public Foo() {
        objects = new Comparable[3];
        objects[0] = new Employee();
        objects[1] = new Car();
        objects[2] = new Employee();
    }
    public Comparable find(String s) {
        for (int i = 0; i < objects.length; i++)
            if (objects[i].isEqual(s))
                return objects[i];
    }
}
```