OBJECT ORIENTED PROGRAMMING

Affefah Qureshi

Department of Computer Science

Iqra University, Islamabad Campus.

INHERITANCE

- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object
- We can create a **general class** that defines traits common to a set of related items
- This class can then be inherited by others, more specific classes, each adding those things that are unique to it
- A class that is inherited is called a **superclass**.
- The class that does the inheriting is called a **subclass**.
- Therefore, a subclass is a **specialized** version of a superclass.

INHERITANCE

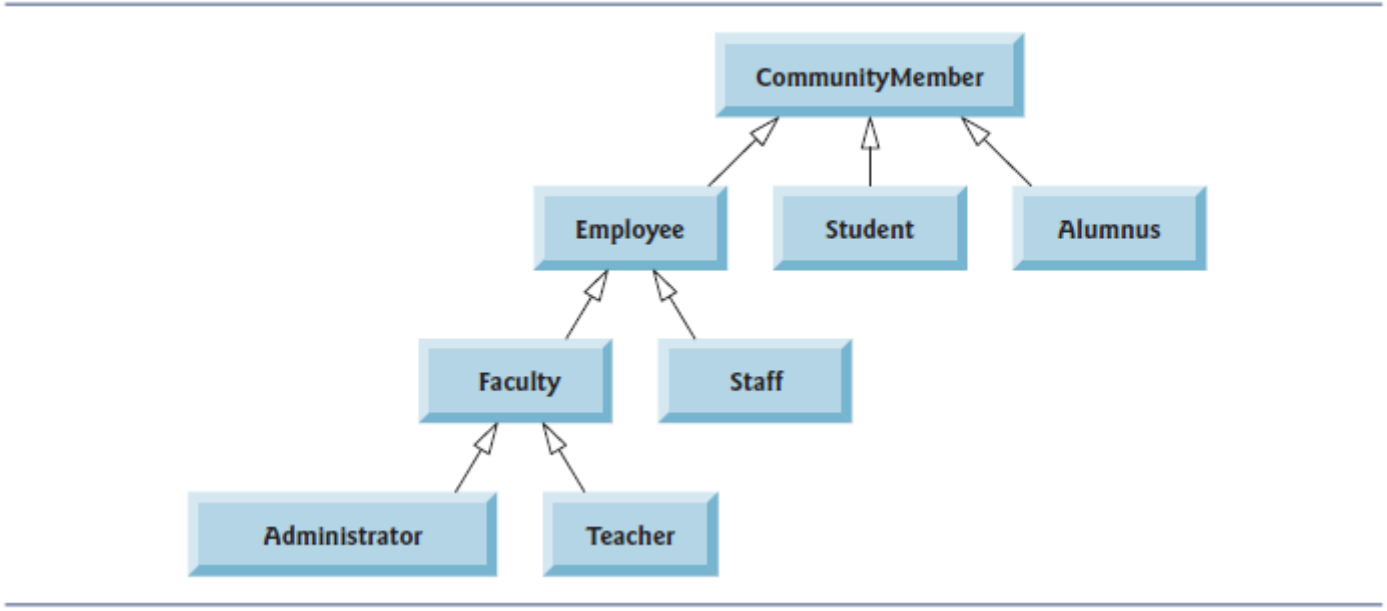
- To inherit a class, we simply incorporate the definition of one class into another by using the **extends** keyword.
- The general form of a subclass declaration is:

  class subclass-name extends superclass-name

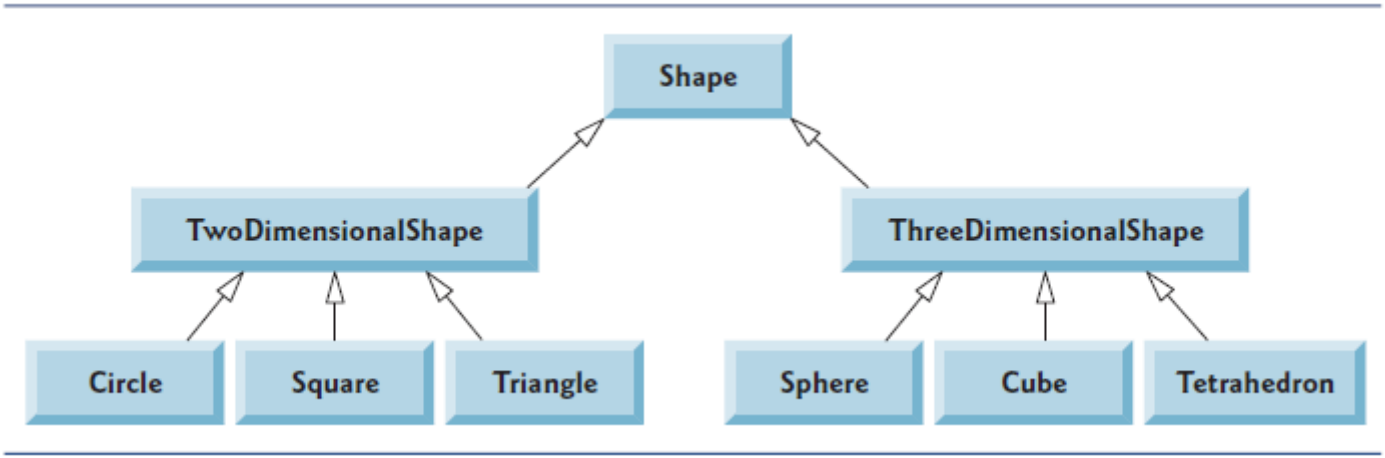
  { /*body of class*/ }

- We can specify only one superclass for any subclass.
- This is because Java does not support **multiple** inheritance. e.g C++

INHERITANCE EXAMPLE



INHERITANCE EXAMPLE



TERMS USED IN INHERITANCE

- **Class**: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class**: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class**: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability**: As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

INHERITANCE EXAMPLE

Superclass	Subclasses
Student	GraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle, Sphere, Cube
Loan	CarLoan, HomeImprovementLoan, MortgageLoan
Employee	Faculty, Staff
BankAccount	CheckingAccount, SavingsAccount

ACCESS SPECIFIER: PROTECTED

- Using protected access offers an intermediate level of access between public and private.
- A superclass's protected members can be accessed by members of that superclass, by members of its subclasses and by members of other classes in the same package—protected members also have package access.
- All public and protected superclass members retain their original access modifier when they become members of the subclass
  - public members of the superclass become public members of the subclass,
  - protected members of the superclass become protected members of the subclass.
  - private members are not accessible outside the class itself. Rather, they're hidden in its subclasses. m the superclass.

WHY USE INHERITANCE IN JAVA

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

INHERITANCE

```
// A simple example of inheritance.
// Create a superclass.
class A {
  int i, j;
  void showij() {
    System.out.println("i and j: " + i + " " + j);
  } }
// Create a subclass by extending class A.
class B extends A {
  int k;
  void showk() {
    System.out.println("k: " + k);
  }
  void sum() {
    System.out.println("i+j+k: " + (i+j+k));
  } }

class SimpleInheritance {
  public static void main(String args[]) {
    A superOb = new A();
    B subOb = new B();
    superOb.i = 10;
    superOb.j = 20;
    System.out.println("Contents of superOb: ");
    superOb.showij();
    System.out.println();
     subOb.i = 7;
    subOb.j = 8;
    subOb.k = 9;
    System.out.println("Contents of subOb: ");
    subOb.showij();
    subOb.showk();
    System.out.println();
    System.out.println("Sum of i, j and k in subOb:");
    subOb.sum(); } }
```

```
Contents of superOb:
i and j: 10 20
Contents of subOb:
i and j: 7 8
k: 9
Sum of i, j and k in subOb:
i+j+k: 24
```

INHERITANCE EXAMPLE

```
// This program uses inheritance to
   extend Box.
class Box {
  double width;
  double height;
  double depth;
  // construct clone of an object
  Box(Box ob) { // pass object to
    constructor
    width = ob.width;
    height = ob.height;
    depth = ob.depth;
  }
 // constructor used when all
    dimensions specified
  Box(double w, double h, double d) {
    width = w;
    height = h; // constructor used
   when cube is created
  depth = d;
}
// constructor used when no
 dimensions specified
Box() {
  width = -1;  // use -1 to indicate
  height = -1; // an uninitialized
  depth = -1;  // box
}
Box(double len) {
  width = height = depth = len;
}

// compute and return volume
double volume() {
  return width * height * depth;
} }
```

INHERITANCE EXAMPLE

```
// Here, Box is extened to include weight.
class BoxWeight extends Box {
  double weight; // weight of box
  // constructor for BoxWeight
  BoxWeight(double w, double h, double d,
    double m) {
    width = w;
    height = h;
    depth = d;
    weight = m;
  }
}
class DemoBoxWeight {
  public static void main(String args[]) {
    BoxWeight mybox1 = new BoxWeight(10,
    20, 15, 34.3);
    BoxWeight mybox2 = new BoxWeight(2, 3, 4,
    0.076);
    double vol;
    vol = mybox1.volume();
    System.out.println("Volume of mybox1 is " +
    vol);
    System.out.println("Weight of mybox1 is " +
    mybox1.weight);
    System.out.println();
    vol = mybox2.volume();
    System.out.println("Volume of mybox2 is " +
    vol);
    System.out.println("Weight of mybox2 is " +
    mybox2.weight);
  }
}
```

Output:

Volume of mybox1 is 3000.0
Weight of mybox1 is 34.3

Volume of mybox2 is 24.0
Weight of mybox2 is 0.076