Classes for Object-Oriented Programming, Inheritance, Polymorphism

IST 256

Application Programming for Information Systems

Building a Class

- Simple employee class

- Two public fields

```
public class Employee
{       // employee fields (visible to subclasses)
        protected String name;
        protected double salary;

        // constructor with initial values
        public Employee(String n, double s)
        {       name = n;
                salary = s;
        }
        // can have multiple constructors
        public Employee( )  {  }

        // other methods
        public void raiseSalary(double percent)
        {    salary = … ;  }
        . . .
}
```

Using the Class

- Employee array with two people

```
public class EmployeeTest
{   public void main(String[ ] args)
    {
            Employee [ ]  staff = new Employee[2];

            staff[0] = new Employee ("Hannah Hacker", 50000);
            staff[1] = new Employee ("Tony Tester", 40000);

            // give everyone a 3% raise
            for (int i = 0; i < staff.length; i++)
            {
                    staff[i].raiseSalary(3.0);
            }
    }
}
```

3

Inheritance

- Another class can be defined that has all of the properties and operations as the previous class, and has additional ones
- Suppose that among our employees, some are managers. They may also have assistants and can have bonus amounts.
- We can make a Manager class that extends the Employee class
  - It will automatically have the fields and methods of the Employee class
  - It can have additional fields and methods
  - It can also redefine some of the methods (more on this later)

superclass

Employee
name
salary

inherits

subclass

Manager
name
salary
assistant
bonus
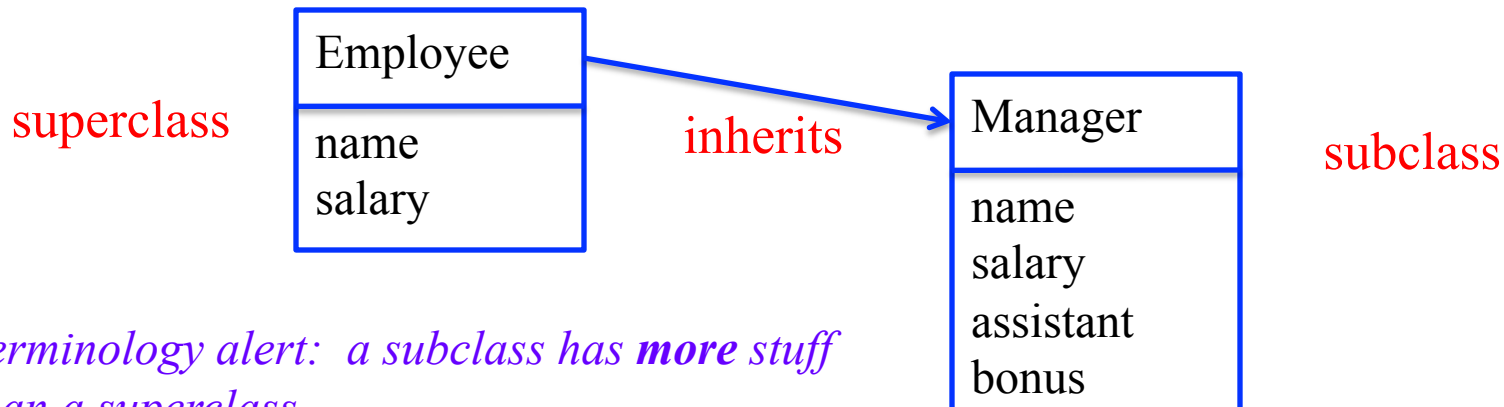
*Terminology alert: a subclass has **more** stuff than a superclass.*

4

Building a Subclass

- Manager class uses the extends keyword to inherit the two fields from the Employee class

- Adds two more fields

```
public class Manager extends Employee
{          // inherits name and salary fields
           // adds two more fields
           public String assistantname;
           public double bonus;


           // constructor with initial values
           public Manager(String n, double s,
                                     String a, double b)
           {          name = n;  salary = s;
                      assistant = a; bonus = b;
           }


                      . . .
}
```

Using the Subclass

- Employee array with three people

- One element of the array is a Manager, which counts as an Employee

```
public class EmployeeTest
{    public void main(String[ ] args)
     {
            Employee [ ]  staff = new Employee[3];

            staff[0] = new Employee ("Hannah Hacker", 50000);
            staff[1] = new Employee ("Tony Tester", 40000);
                                "Tony Tester", 10000)
            staff[2] = new Manager ("Boss Boring", 90000,

            // give everyone a 3% raise
            for (int i = 0; i < staff.length; i++)
            {
                    staff[i].raiseSalary(3);
            }
      }
}
```

6

Polymorphism

- In the previous example, the subclass Manage used the same raiseSalary method as did all Employees

- But we can also redefine a method of the superclass by giving a new definition in the subclass

Subclass with a polymorphic method

- Manager class now adds its own definition of the raiseSalary method, which can do something special for managers

```
public class Manager extends Employee
{        // inherits name and salary fields
         // adds two more fields
         public String assistantname;
         public double bonus;

         // constructor with initial values
         public Manager(String n, double s,
                                  String a, double b)
         {        name = n;  salary = s;
                  assistant = a; bonus = b;
         }
         // another version of raiseSalary
         public void raiseSalary(double percent)
         {    salary = … ;
              bonus = … ;  }
                  . . .
}
```

Using a polymorphic method

- Same code as before

- Each call to raiseSalary will use the appropriate method, either from the Employee class or the Manager class

```
public class EmployeeTest
{    public void main(String[ ] args)
     {

            Employee [ ]  staff = new Employee[3];

            staff[0] = new Employee ("Hannah Hacker", 50000);
            staff[1] = new Employee ("Tony Tester", 40000);
                                     "Tony Tester", 10000)
            staff[2] = new Manager ("Boss Boring", 90000,

            // give everyone a 3% raise
            for (int i = 0; i < staff.length; i++)
            {

                    staff[i].raiseSalary(3);

            }

     }

}
```

9

Inheritance Hierarchies

- An application may create and use many extended classes.
- Information systems may use these hierarchies in the design of a system.