|  |  |
| --- | --- |
|  | **CSC-150 – Object Oriented Programming** |
| **Semester II (Fall 2023)**  **Course Instructor(s): Dr. Fahim Akhtar, Engr. Asif Ali** |

|  |
| --- |
| **Lab 06**  **Inheritance in JAVA** |

**Objective(s):**

1. Inheritance in JAVA
2. Super keyword

**Theory**

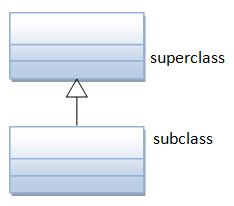
|  |
| --- |
| **1: Inheritance in JAVA** |

Inheritance is an important object-oriented concept that allows classes to be reused in order to define similar, but distinct, classes. In OOP, classes are organized in *hierarchy* to *avoid duplication and reduce redundancy.*

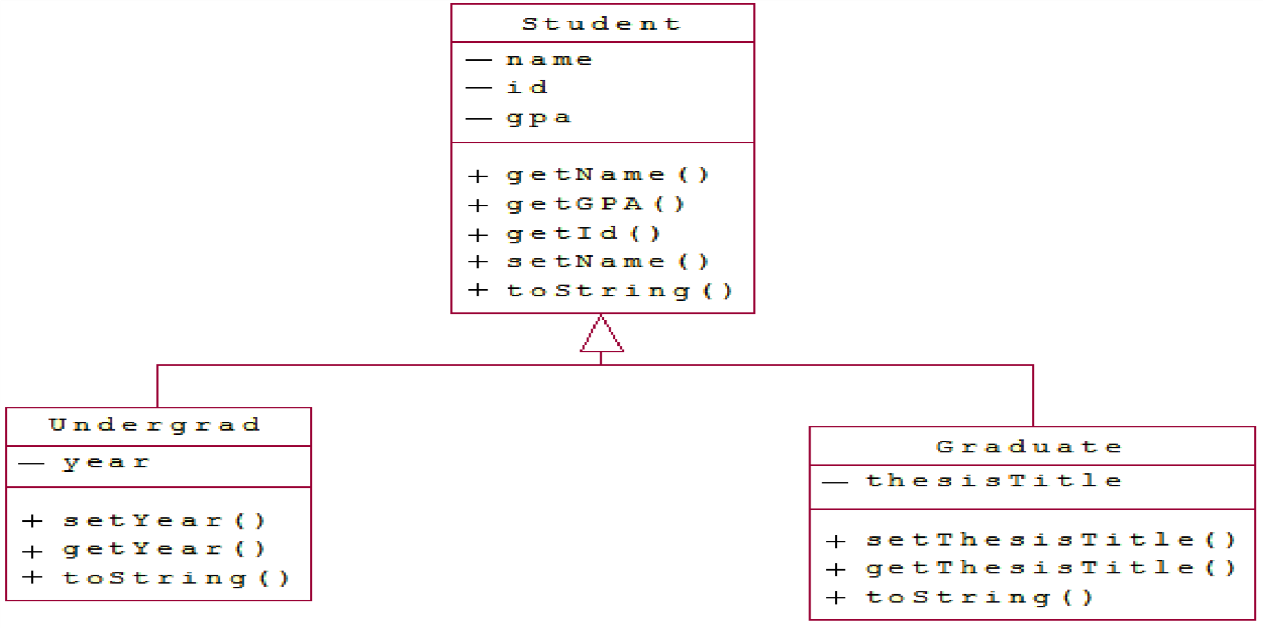
The classes in the lower hierarchy inherit all the members (variables and methods) from the higher hierarchies, it cannot access those members of the higher hierarchies that have been declared as private. A class in the lower hierarchy is called a *subclass* (or *derived*, *child*, *extended class*). A class in the upper hierarchy is called a *superclass* (or *base*, *parent class*). By pulling out all the common variables and methods into the superclasses, and leave the specialized variables and methods in the subclasses, *redundancy* can be greatly reduced or eliminated as these common variables and methods do not need to be repeated in all the subclasses. ***extends****is the keyword used to inherit the properties of a class.*

For example:

A subclass inherits all the variables and methods from its superclasses, including its immediate parent as well as all the ancestors. It is important to note that a subclass is not a "subset" of a superclass. In contrast, subclass is a "superset" of a superclass. It is because a subclass inherits all the variables and methods of the superclass; in addition, it extends the superclass by providing more variables and methods.

**UML Notation:** The UML notation for inheritance is a solid line with a hollow arrowhead leading from the subclass to its superclass. By convention, superclass is drawn on top of its subclasses as shown.

**Example:**



The class Student is the parent class. Note that all the variables are private and hence the child classes can only use them through accessor and mutator methods. Also note the use of **overloaded** constructors.

|  |
| --- |
| public class Student{  private String name; private int id; private double gpa;  public Student(int id, String name, double gpa) {  this.id = id;  this.name = name;  this.gpa = gpa;  }    public Student(int id, double gpa){  this(id, "", gpa);  }  public String getName(){return name;}  public int getId() {return id;}  public double getGPA(){return gpa;}  public void setName(String newName){  this.name = newName;  }  public String toString(){  return "Student:\nID: "+id+"\nName: "+name+"\nGPA: "+gpa;  }    }// Student class ends |

The class Undergrad extends the Student class. Note the overridden toString() method

|  |
| --- |
| public class Undergrad extends Student {  private String year;  public Undergrad(int id, String name, doubl gpa, String year) {  super(id, name, gpa); // super() can be used to invoke immediate parent class constructor.  this.year = year;  }  public String getYear() {return year;}    public void setYear(String newYear) {this.year = newYear;}  public String toString() {  return "Undergraduate "+super.toString()+"\nYear: "+year; } } //Undergrad class ends |

The class Graduate extends the Student class too. Note the overridden toString() method

|  |
| --- |
| public class Graduate extends Student {  private String thesisTitle;  public Graduate(int id, String name, double gpa, String thesisTitle) {  super(id, name, gpa);  this.thesisTitle = thesisTitle;  }  public String getthesisTitle() { return thesisTitle; }  public void setThesisTitle(String newthesisTitle) {  this.thesisTitle = newthesisTitle;  }  public String toString() {  return "Graduate " +super.toString()+"\nThesis: "+thesisTitle; } } // Graduate class ends |

TestStudents is a driver class to test the above classes

|  |
| --- |
| public class TestStudents {  public static void main(String[] args) {  Student s1 = new Student(123456, "Aariz", 3.27);  Student s2 = new Student(234567, 3.22);  Undergrad u1 = new Undergrad(345678, "Asad", 2.73, "Junior");  Graduate g1 = new Graduate(456789, "Ahmed", 3.67, "Algorithms and Complexity");  System.out.println(s1);  System.out.println(s2);  System.out.println(u1);  System.out.println(g1);  }  }// TestStudents class ends |

**The super keyword**

The super keyword is like this keyword. Following are the scenarios where the super keyword is used.

It is used to differentiate the members of superclass from the members of subclass, if they have same names.

It is used to invoke the superclass constructor from subclass.

super.variable

super.method();

Syntax for using *Super* keyword

class Super\_class {

int num = 20;

// display method of superclass

public void display() {

System.out.println("This is the display method of superclass");

}

}

public class Sub\_class extends Super\_class {

int num = 10;

// display method of sub class

public void display() {

System.out.println("This is the display method of subclass");

}

public void my\_method() {

// Instantiating subclass

Sub\_class sub = new Sub\_class();

// Invoking the display() method of sub class

sub.display();

// Invoking the display() method of superclass

super.display();

// printing the value of variable num of subclass

System.out.println("value of the variable named num in sub class:"+ sub.num);

// printing the value of variable num of superclass

System.out.println("value of the variable named num in super class:"+ super.num);

}

public static void main(String args[]) {

Sub\_class obj = new Sub\_class();

obj.my\_method();

}

**Lab Tasks**

Exercise 1 (Hierarchical Inheritance) (Employee.java, Teacher.java, Professor.java)

Define **Employee** as a super(Parent) class, it contains three instances name, qual, and grade. It contain Intro () method that display name, qual, and grade

Employee class derive **Teacher** class, it contain one instance subject. It contains teachingMethod () that display all subjects taught by teacher.

**Employee** class derive **Professor** class, it contains two instances publications, and experience. It contains two methods ResearchPublication( ) it should display number of publications. Experience ( ) methods returns number of experience by Professor.

Exercise 2 (Multilevel Inheritance) *(Student.java,*BSStudent.java, MSStudent*.java)*

Define a class **Student**, it contains three instances name, id, and year. It contain one constructor that display name, id, and year.

Student class derive **BSStudent** class, it contains two instances gpa, and award. It contains one method Award( ) that should display name, id, year, gpa, and award.

**BSStudent** class derive **MSStudent** class, it contains two instances specialization, and publication. It contains one method Publication( ) that should display name, id, year, specialization,and publication.

Exercise 3 (Inheritance)

Define a class named **University** that contains two instance variables of name, and rank.

Also create a method named **displayDetails** that outputs the details of a University.

Next, define a class named **Department** that is derived from **University**, it contains three instances field\_specialization, field\_type (e.g: Engineering, Science etc), and dept\_head. Also, it contains **departmentDetails** that output the details of a department.

Next, define a class name **Library** that is derived from **University**, it contains two instances no\_books, and type\_books. Also, it contains **libraryDetails** that display the details of Library.

Create **a main method** that creates at least two Library objects and two Department objects with different values and calls displayDetails for each.

Exercise 4 (Super, this) Keywords.java

Define a class named SuperClass that contains an instance variable of type String named.Create a method named Info( ) that returns the name.

Next, define a class for **SubClass** that is derived from **SuperClass** and includes instance variables String name. Also it contains method name Info( ) that return name.

|  |
| --- |
| **Post lab questions** |

1. Can class extend more than one classes?If yes, give reason.
2. What is difference between composition and aggregation?
3. Can you restrict a class from inheriting another class?
4. Can we access private instances in derived classe?
5. What is difference between this and super keyword?
6. Why multiple level inheritance is not allowed in Java?

**END**