Sukkur Institute of Business Administration University
Department of Computer Science
BS – II (CS/SE/AI) Spring 2024

Object Oriented Programming

Lab # 06: To become familiar with Inheritance and Super Keyword

Instructor: Engr. Zainab Umair Kamangar

Lab Report Rubrics (Add the points in each column, then add across the bottom row to find the total score)					**Total Marks**
S.No	**Criterion**	**0.5**	**0.25**	**0.125**	
1	Accuracy	☐ Desired output	☐ Minor mistakes	☐ Critical mistakes	
2	Timing	☐ Submitted within the given time	☐ 1 day late	☐ More than 1 day late	

Submission Profile

Name: Submission date (dd/mm/yy):
Enrollment ID: Receiving authority name and signature:
Comments:

_____
Instructor Signature

Note: Submit this lab hand-out in the next lab with attached solved activities and exercises

Objectives

After performing this lab, students will be able to understand,

- Inheritance in Java
- Super Keyword

Inheritance in JAVA

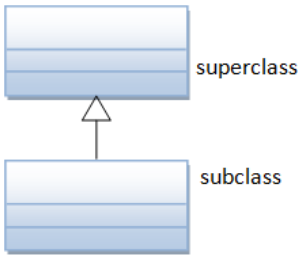
Inheritance is an important object-oriented concept that allows classes to be reused in order to define similar, but distinct classes. In OOP, classes are organized in *hierarchy* to *avoid duplication and reduce redundancy.*

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.*

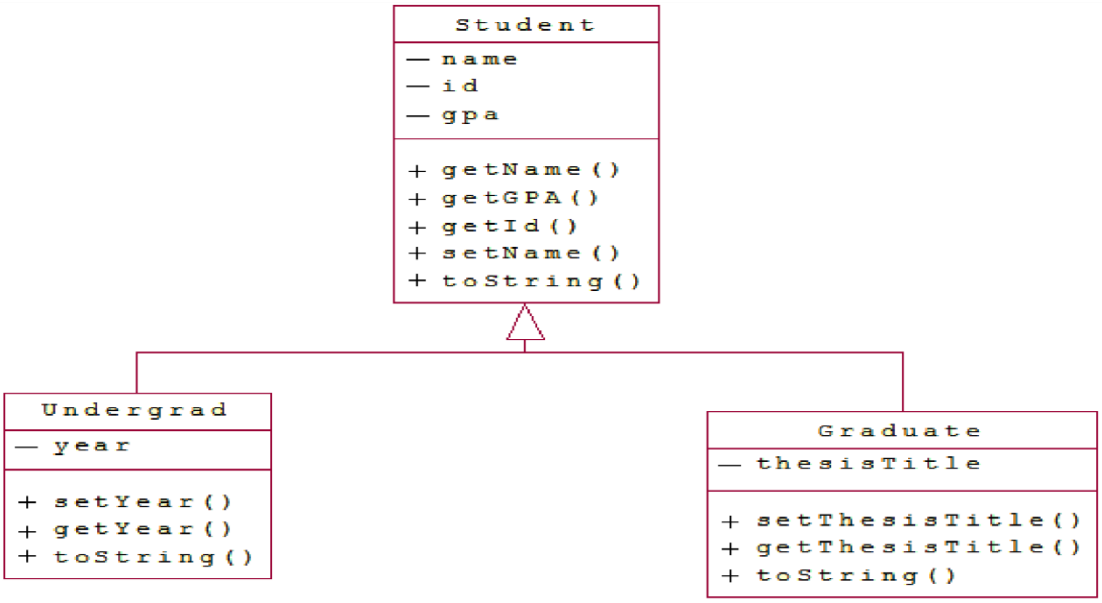
For example:

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, double 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();
   }
}
```

Exercises

Question 1: (Hierarchical Inheritance) (Employee.java, Teacher.java, Professor.java)

Define Employee as a super(Parent) class, it contains three instances name, qualification, and grade. It contains Intro( ) method that displays name, qualification, and grade.

Employee class derives Teacher class, it contains one instance subject. It contains teachingMethod () that displays all subjects taught by teacher.

Employee class derives Professor Class, it contains two instances publication and experience. It contains two methods ResearchPublication ( ) it should display number of publications, name, qualification, and grade. Experience ( ) methods returns number of years of experience by Professor.

Employee class should contain main method.

Question 2: (Multilevel Inheritance) (Student.java, BSStudent.java, MSStudent.java)

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

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

Student should contain main method.

Question: 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 (name, field_specialization, field_type, dept_head) 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 (name, no_books, type_books) 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.

Question: 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.

Question: 5 (TestCompany.java)

```
public class TestCompany {
    public static void main(String [] args){
        Product p1 = new Product("TV",4,34000);
        Product p2 = new Product("Bicycle", 4, 5500);
        Product p3 = new Product("Oven", 3,70000);

        Store s1 = new Store("Makro", "Karachi");
        Store s2 = new Store("Hypermart","Karachi");
        s1.addProduct(p1);
        s1.addProduct(p2);
        s1.addProduct(p3);
        s1.displayAll();
        Product tempProduct = s1.deleteProduct("Bicycle");
        if (tempProduct!=null)
            System.out.println("Product "+tempProduct.getName()+ " is
deleted");
        else
            System.out.println("There is no product to delete");
         s1.displayAll();
         s2.addProduct(p1);
         s2.addProduct(p2);
         s2.addProduct(p3);
         s2.displayAll();
         Company c1 = new Company("Unilever");
         c1.addStore(s1);
         c1.addStore(s2);
         c1.displayAll();
         int n= c1.searchNbofStore("TV");
         System.out.println("Number of stores have TV "+n);

    }
}
```

Implement Product, Store and Company classes and use the following class to test.

Question: 6 (Inheritance)

Define a class named Message that contains an instance variable of type String named text that stores any textual content for the Message. Create a method named toString that returns the text field and also include a method to set this value.

Next, define a class for SMS that is derived from Message and includes instance variables for the recipientContactNo. Implement appropriate accessor and mutator methods. The body of the SMS message should be stored in the inherited variable text. Redefine the toString method to concatenate all text fields.

Similarly, define a class for Email that is derived from Message and includes an instance variable for the sender, receiver, and subject. The textual contents of the file should be stored in the inherited variable text. Redefine the toString method to concatenate all text fields.

Create sample objects of type Email and SMS in your main method. Test your objects bypassing them to the following subroutine that returns true if the object contains the specified keyword in the text property.

```
public static boolean ContainsKeyword(Message messageObject,String keyword) {
        if (messageObject.toString().indexOf(keyword,0) >= 0)
         return true;
        return false; }
```

Finally, include a method to encode the final message "This is Java" using an encoding scheme, according to which, each character should be replaced by the character that comes after it. For example, if the message contains character B or b, it should be replaced by C or c accordingly, while Z or z should be replaced with an A or a. If the final message is "This is Java", then the encoded message should be "UijtjtKbwb".

Answer the following Questions:

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