Essentials of Information Technology

PC-CS-305

Polymorphism

Objectives

In this chapter, we will

- Review inheritance
- Introduce polymorphism

gauravgambhir.cse@piet.co.in

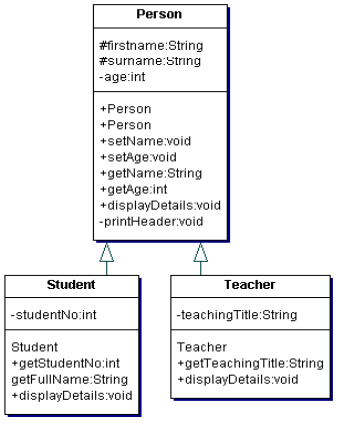
Inheritance

- Three key topics in object oriented programming are:
  - Encapsulation
  - Inheritance
  - Polymorphism

gauravgambhir.cse@piet.co.in

Inheritance – an example

- Person is a base class
  - A super class
- Student and Teacher are derived classes
  - Sub classes
- The new classes are publicly derived from the base class
  - all the public members of the base class are available to instances of the derived classes



gauravgambhir.cse@piet.co.in

The Teacher Class

- The Teacher class adds a teaching title such as Lecturer to the details encapsulated by a Person

```
public class Teacher extends Person
{
   private String teachingTitle ;
   public Teacher (String theFirstname, String theSurname,
            int theAge, String theTeachingTitle)
   {
      super(theFirstname, theSurname, theAge);
      teachingTitle =theTeachingTitle;
   }
```

gauravgambhir.cse@piet.co.in

The Teacher Class (continued)

```
   public String getTeachingTitle()
   {
      return teachingTitle;
   }

   public void displayDetails ()
   {
     super.displayDetails();
     System.out.println(teachingTitle);
   }
}
```

gauravgambhir.cse@piet.co.in

The IS-A Relationship

- The IS-A relationship is used to describe the relationship between an instance of a derived class and the base class
- A Student IS-A Person
- An instance of Student is at least an instance of Person and can be used anywhere that a Person can be used
  - the converse is not true
  - a Person cannot be used everywhere that a Student can be used
    - a Person does not have to be a Student
- Similarly a Teacher IS-A Person and can be used anywhere that a Person can be used
- A reference to a Student is also a reference to a Person

gauravgambhir.cse@piet.co.in

References to Person Objects

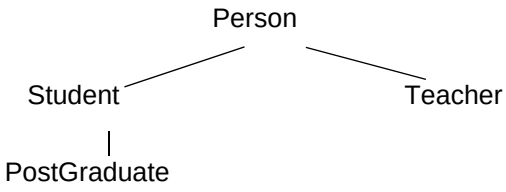
- Consider the following code:
  Student Alex = new Student("Alex","King",20,199913);
  Teacher Pete = new Teacher ("Pete", "King",52, "Principal Lecturer"); Person theTutor = new Person ("Bill", "Smith",52);
  Person thePeople[] = new Person[3];
  thePeople[0] = Pete;
  thePeople[1] = Alex;
  thePeople[2] = theTutor;

- A Teacher can be added to the array because a Teacher is derived from a Person; a Teacher IS-A Person
  - The array actually holds references
  - **Example** Person.java , Teacher.java , Student.java , TestPerson.java

gauravgambhir.cse@piet.co.in

PostGraduate Class

- How should the PostGraduate class be built?



- A PostGraduate is a Student and is a Person
- A PostGraduate should extend the Student class
- If super is used in Postgraduate it will refer to the Student class
  - Only the immediate ancestor is accessible

gauravgambhir.cse@piet.co.in

Polymorphism

- Polymorphism literally means state of having many shapes or the capacity to take on many different forms. ***Box.java***
- Polymorphism can be of two types
  - Compile time polymorphism also called method overloading.
  - Runtime polymorphism also called method overriding.
- **Method overloading** is the ability of one function to perform different task.
- Method overloading allows creating several methods with the same name which differ from each other in the type of input and the output of the method. **Sum.java**
- **Method overriding** allows a subclass or a child class to provide a specific implementation of a method that is already provided by one of its super class or parent class. ***Override.java , PersonOverride.java***

gauravgambhir.cse@piet.co.in

Polymorphism

- The implementation in the subclass overrides (replaces) the implementation in the super class by providing a method that has same name , same parameters or signature and same return type as the method in the parent class.
- The version of the method that is executed will be determined by the object that is used to invoke it. If the object of the parent class is used to invoke the method, then the version in the parent class will be executed but if an object of the subclass is used to invoke the method then the version in the child class will be executed.
- Animal.java , Dog.java, Fish.java, Polymorphism.java

gauravgambhir.cse@piet.co.in

Polymorphism

- Consider the following code:

```
for(int i =0; i < 3; ++i)
    thePeople[i].displayDetails();
```

- What will be output?
- The array holds references to Person objects so on the face of it the displayDetails method from the Person class should be invoked
- The objects are not all of type Person
- At run time the Java environment determines the type of object and binds to the correct version of displayDetails
- This is Polymorphism
- Example : Apple.java , Food.java , Rice.java , Maggi.java

gauravgambhir.cse@piet.co.in

Polymorphism

- Java makes use of dynamic binding with classes
  - Late binding
- The method to bind to is determined at run time based upon the type of object that is actually referenced
- The alternative to this is early binding
  - Static binding
  - This is used by many other languages
  - The method to bind to is determined at compile time based upon available type information
- Languages like C++ allow the programmer to choose

gauravgambhir.cse@piet.co.in

Polymorphism

- **Overloading –** Which version of the method to be called is decided at compile time based on reference type
- **Overriding –** Which version of the method to be called is decided as run time based on object type.
- **Example** TestAnimals.java

gauravgambhir.cse@piet.co.in

Casting and instanceof

- It is possible to check if an object reference is to a specific class using instanceof
  - Then a cast can be carried out safely
- For example:

```
for ( int i =0; i < 3; ++i)
{
    if (thePeople[i] instanceof Teacher)
    {
        Teacher theTeacher = (Teacher)thePeople[i];
        System.out.println(theTeacher.getTeachingTitle());
    }
}
```

Example TestPerson.java

gauravgambhir.cse@piet.co.in

Object

- In Java the ultimate base class of all classes is Object
- In the class hierarchy discussed earlier the base class is Person
  - Person inherits from Object
  - An instance of Person is an instance of Object
  - Student inherits from Person
  - An instance of Student is an instance of Person and is an Instance of Object
- The class Object is defined in the namespace java.lang
- Provides some common functionality to all classes

gauravgambhir.cse@piet.co.in

Object and the toString method

- The toString method is provided by the base class Object
- The toString method is invoked, for example, when an object is referred to in println
  - System.out.println(myObject)
- The default return value of toString is a String made up of
  - The name of the class that the object is an instance of
  - Followed by an @ symbol and a numeric code
- It is possible to override the toString method in any derived class
- The signature of the method is:
  - public String toString( )

gauravgambhir.cse@piet.co.in

The Person class

- A toString method could easily be added to the Person class:

```
public class Person {
    private String name;
    private int age;

    public String toString( )
    {
       return "Name is : " + name + " " +
              surname + " Age is  : " + age ;
    }
    …
```

Example : Person.java

gauravgambhir.cse@piet.co.in

Using the toString method of Person

- The toString method is called by default when necessary:

```
System.out.println("\nTesting toString of Person");
System.out.println(theTutor);
```

- The method can be called directly by name:

```
System.out.println("\nTesting toString of Person");
System.out.println(theTutor.toString());
```

gauravgambhir.cse@piet.co.in

Overriding toString in derived classes

- The toString method can be overriden in the Student class for example:

```
public String toString( )
{
    return "Student " + super.toString() ;
}
```

- Note the toString method of the super class can be used wherever it is required
  - It is not compulsory to use it but often avoids repeating code
  - **Example** Student.java

gauravgambhir.cse@piet.co.in

Summary

In this lecture we have:

- Reviewed inheritance
- Discussed the IS-A relationship
- Introduced polymorphism

gauravgambhir.cse@piet.co.in

Question and Answer Session

Q & A

gauravgambhir.cse@piet.co.in