CSE1007 – JAVA PROGRAMMING

Module - 2

Polymorphism

Polymorphism

- Polymorphism is one of three pillars of object-orientation.
- Polymorphism: many different (poly) forms of objects that share a common interface respond differently when a method of that interface is invoked:

  1) a super-class defines the common interface
  2) sub-classes have to follow this interface (inheritance)

  but are also
- Permitted to provide their own implementations (overriding)
- A sub-class provides a specialized behaviors relying on the common elements defined by its super-class.

Method overridding

Overriding Methods

Override the method defined in the superclass by defining a method in the subclass that has

same name,
same arguments and
same return type

as the method in the superclass

When the method is called, the method defined in the subclass is invoked instead of the one in the superclass.

•When an overridden method is called from within the sub-class:

1) it will always refer to the sub-class method

2) super-class method is hidden

Method overridding

```
package CSE1007_MODULE2;
class PolyA
{
    int i, j;
    PolyA(int a, int b)
    {
        i = a;
        j = b;
    }
    void show()
    {
        System.out.println("PARENT CLASS: i and j: " + i + " " + j);
    }
}
```

```
CHILD CLASS k: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
class PolyB extends PolyA
{
    int k;
    PolyB(int a, int b, int c)
    {
        super(a, b);
        k = c;
    }
    void show()
    {
        System.out.println("CHILD CLASS k: " + k);
    }
}
public class Polymorphism1
{
    public static void main(String args[])
    {
        PolyB subOb = new PolyB(1, 2, 3);
        subOb.show();
    }
}
```

Example1

Method overridding

```
package CSE1007_MODULE2;
class PolyA2
{
    int i, j;
    PolyA2(int a, int b)
    {
        i = a;
        j = b;
    }
    void show()
    {
        System.out.println("PARENT CLASS: i and j: " + i + " " + j);
    }
}
```

```
PARENT CLASS: i and j: 1 2
CHILD CLASS k: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Example2

Super and Method Overriding

```
class PolyB2 extends PolyA2
{
    int k;
    PolyB2(int a, int b, int c)
    {
        super(a, b);
        k = c;
    }
    void show()
    {
        super.show();
        System.out.println("CHILD CLASS k: " + k);
    }
}
public class Polymorphism2
{
    public static void main(String args[])
    {
        PolyB2 subOb = new PolyB2(1, 2, 3);
        subOb.show();
    }
}
```

Overriding versus Overloading

Example3

```
package
CSE1007_MODULE2;
class Sum1
{
       int c;
       void add(int a, int b)
       {
              c=a+b;
        }
     void display()
      {
             System.out.println(c);
      }
}
```

```
class Sum2  extends Sum1
{
     int d;
     void add(int a, int b, int c)
     {
         d=a+b+c;
      }
      void display()
      {
System.out.println("SUM="+d);
      }
}
```

```
public class Polymorphism3
{
    public static void main(String
args[])
    {
      Sum2 s2  = new Sum2();
      s2.add(2,3,5);
      s2.display();
     }
}
```

```
SUM=10
BUILD SUCCESSFUL (total time: 0 seconds)
```

Overriding versus Overloading

Example 4

```
package CSE1007_MODULE2;
class PolyA3
{
    int i, j;
    PolyA3(int a, int b)
    {
        i = a;
        j = b;
    }
    void show()
    {
        System.out.println("PARENT CLASS: i and j: " + i + " " + j);
    }
}
class PolyB3 extends PolyA3
{
    int k;
    PolyB3(int a, int b, int c)
    {
        super(a, b);
        k = c;
    }
    void show(String msg)
    {
        //super.show();
        System.out.println("CHILD CLASS with string parameter k: " + msg+"  "+k);
    }
}
public class Polymorphism4
{
    public static void main(String args[])
    {
        PolyB3 subOb = new PolyB3(1, 2, 3);
        subOb.show();
        subOb.show("Sub class");
    }
}
```

```
PARENT CLASS: i and j: 1 2
CHILD CLASS with string parameter k: Sub class   3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Polymorphism

- Suppose we have a hierarchy of classes:

  1) The top class in the hierarchy represents a common interface to all classes below. This class is the base class.

  2) All classes below the base represent a number of forms of objects, all referred to by a variable of the base class type.

Dynamic Method Invocation

- Method overriding allows for dynamic method invocation:

  1) an overridden method is called through the super-class variable

  2) Java determines which version of that method to execute based on the type of the referred object at the time the call occurs

  3) when different types of objects are referred, different versions of the overridden method will be called.

Dynamic Method Invocation

```
package CSE1007_MODULE2;
class PolyA5
{
   void callme()
   {
     System.out.println("Inside A's callme method");
   }
}
class PolyB5 extends PolyA5
{
   void callme()
   {
      System.out.println("Inside B's callme method");
   }
}
```

```
Inside A's callme method
Inside B's callme method
Inside C's callme method
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
class PolyC5 extends PolyA5
{
   void callme()
   {
      System.out.println("Inside C's callme
method");
   }
}
public class Polymorphism5
{
  public static void main(String args[])
  {
     PolyA5 a = new PolyA5();
     PolyB5 b = new PolyB5();
     PolyC5 c = new PolyC5();
     PolyA5 r;
     r = a;
     r.callme();
     r = b;
     r.callme();
     r = c;
     r.callme();
  }
}
```

Dynamic Method Invocation

```
package CSE1007_MODULE2;
class PolyA5
{
   void callme()
   {
    System.out.println("Inside A's callme method");
   }
}
class PolyB5 extends PolyA5
{
   void callme()
   {
      System.out.println("Inside B's callme method");
   }
}
class PolyC5 extends PolyA5
{
   void callme()
   {
      System.out.println("Inside C's callme method");
   }
}
public class Polymorphism5
{
 public static void main(String args[])
 {
    PolyA5 r1=new PolyA5();
    PolyA5 r2=new PolyB5();
    PolyA5 r3=new PolyC5();
    r1.callme();
    r2.callme();
    r3.callme();
  }
}
```

```
Inside A's callme method
Inside B's callme method
Inside C's callme method
BUILD SUCCESSFUL (total time: 0 seconds)
```