



Polymorphism

Filiz Zevri

CSE

Contents

- Definitions of polymorphism
- Mechanism of realization. Examples
- Conclusions

Definitions

- Polymorphism = the ability to substitute objects of matching interface for one another at run time.
- Polymorphism = the ability to perform the same operation on many types, as long as each type shares a common subset of characteristics.

Mechanism of realization

The general mechanism used to realize the polymorphism is called overriding.

Overriding, which is also known as late binding mechanism, is performed in two different ways:

- inheriting classes
- implementing interfaces

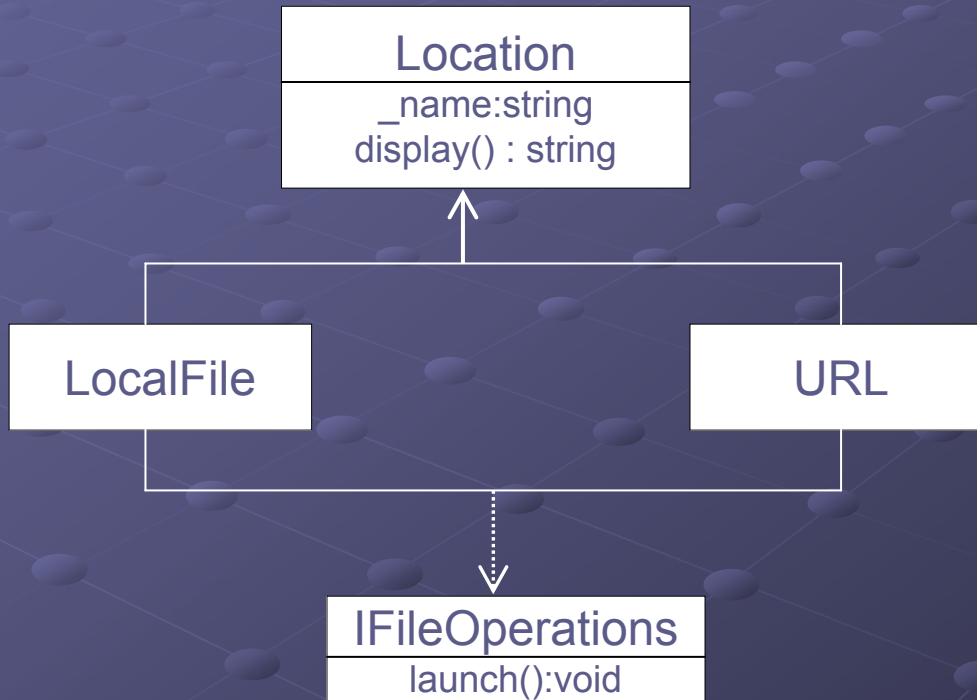
Examples

Let's consider the following classes:

- Location: the base class
- LocalFile, URL: classes derived from Location, which implement the interface IFileOperations
- IFileOperations: an interface, that provides only a single method to be implemented in the classes LocalFile and URL

Examples

Class diagram for the defined entities



Examples

Implementation of classes

```
public class Location
{
    // instance variables
    protected String _name;

    //Constructor of the class Location
    public Location(String name)
    {
        _name = name;
    }
    //This method will be overridden in the derived classes.
    public String display()
    {
        return "General location is "+ _name;
    }
}
```

```
public interface IFileOperations
{
    //The interface specifies only a single method
    public void launch();
}
```

```
public class LocalFile extends Location implements IFileOperation
{
    //The constructor only calls the base class constructor
    public LocalFile(String name)
    {
        super(name);
    }
    //The method display is overridden in the derived class LocalF
    public String display()
    {
        return "Absolute path name is "+ _name;
    }
    public void launch()
    {
        System.out.println("Run Win32");
    }
}
```

```
public class URL extends Location implements IFileOperations
{
    //The constructor calls only the base class constructor
    public URL(String name)
    {
        super(name);
    }

    //The method "display" is overridden in the derived class URL
    public String display()
    {
        return "URL is "+ _name;
    }
    //Implementation of the method "launch" belonging to interface IFileOperat
    public void launch()
    {
        System.out.println("Run Internet Explorer");
    }
}
```

Examples

The „main“ method implementation

```
public static void main(String[] args)
{
    //create objects for each class
    Location generalLocation = new Location("some location");
    LocalFile file = new LocalFile("c:\\Systems");
    URL url = new URL("www.cse.tum.de");

    showName(generalLocation); //will display "General location is some location"
    showName(file); //will display "Absolute path name is c:\\Systems"
    showName(url); //will display "URL is www.cse.tum.de"

    //1st example: polymorphism by inheriting a class
    //the upcasting from LocalFile to Location
    generalLocation = file;
    //the concrete type of the object hidden by the variable generalLocation at run time
    //is LocalFile
    showName(generalLocation); //will display "Absolute path name is c:\\Systems"

    //2nd example: polymorphism by implementing an interface
    launchFile(file); //will display "Run Win32"
    launchFile(url); //will display "Run Internet Explorer"

    //the following cast is possible, because LocalFile implements the IFileOperations interface
    IFileOperations fileOperations = file;
    launchFile(fileOperations); //will display "Run Win32", because at run time the type of the
    //object is LocalFile
}

public static void showName(Location loc)
{
    System.out.println(loc.display());
}

public static void launchFile(IFileOperations fileOp)
{
    fileOp.launch();
}
}
```

Conclusions

- The example proved that the calling of the appropriate method takes place at execution time. The so called late binding mechanism recognizes the right type of the object and calls the right implementation of the method.
- Each type can implement a shared characteristic in its own way. In our case, URL has now a custom way of displaying itself.
- The signatures of the overridden methods must be identical.