**National University of Computer & Emerging Sciences, Karachi**

**Computer Science Department**

**Fall 2023, Lab Manual – 07**

|  |  |
| --- | --- |
| **Course Code: CL-1004** | **Course : Object Oriented Programming Lab** |
| **Instructor(s) :** | **Shafique Rehman** |

**LAB - 7**

Hierarchical inheritance & Polymorphism

**Hierarchical Inheritance:**

When two or more classes inherits a single class, it is known as hierarchical inheritance. The base class has attributes that are protected but are accessible from the subclass.

**Example:**

****

Figure : hierarchial inheritance example

**Polymorphism:**

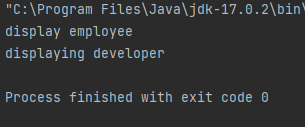
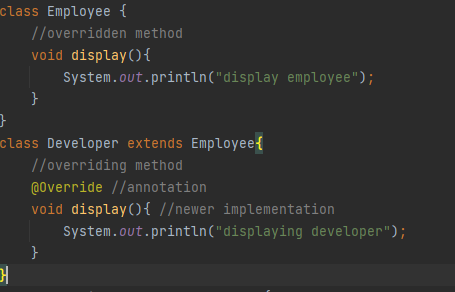
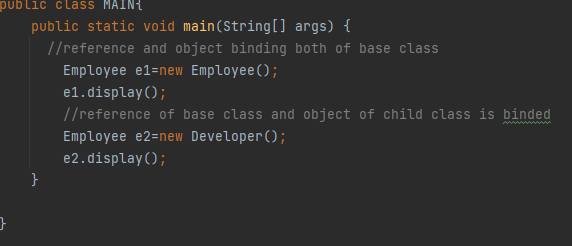
Polymorphism is derived from poly and morphs. The word "poly" means many and "morphs" means forms. So, polymorphism means many forms. Itis a concept by which we can perform a *single action in different ways*. There are two types of polymorphism in Java:

* compile-time polymorphism / static polymorphism / static binding
* runtime polymorphism/ dynamic polymorphism/ dynamic binding.

|  |  |
| --- | --- |
| **Static binding** | **Dynamic binding** |
| Done at compile time | Done at run time |
| Actual object not used | Actual objects are used |
| Also early binding | Also, late binding |
| e.g., method overloading | e.g., method overriding |

**Static binding:**

Early binding done at compile time is static binding. Here, in successive example I have shown that (in red box) when an employee object is called, all its methods are bind with the functions call. So, when I call display function using e1 that will show the output of the Employee class as the reference object is bind with the member function of employee class.



**Dynamic binding:**

Late binding or runtime binding is the concept in which the JVM binds the function at runtime with their definition so any overridden method will have newer definition at runtime. As in the example (in blue box), we can see that the reference object e2 binds the display method of Developer class with the e2’s display function call

Figure : static vs dynamic binding e.g.



**Method Overloading:**

A feature that allows a class to have more than one method having the same name but with different argument list. It is similar to [constructor overloading](https://beginnersbook.com/2013/05/constructor-overloading/) in Java, that allows a class to have more than one constructor having different argument lists.

In order to overload a method, the argument lists of the methods must differ using either of the given options:

* Different number of parameters.
* Different data type of parameters
* Sequence of data type of parameters.

Methods with different return types doesn’t matter in case of overloading.

Figure 1 shows the overloading of constructor. We can see that; the function names are same but only the argument list varies in both the cases. Same goes for any other function in which return type and function remains same. Only the argument lists will change. It is like making multiple versions of a functions in a class.

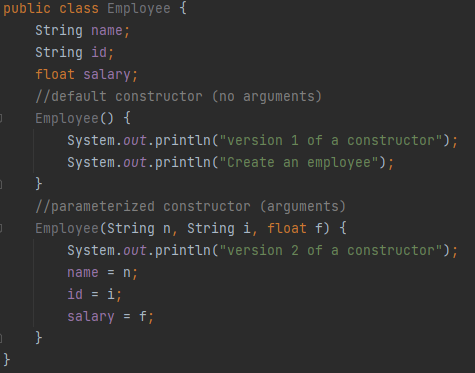


Figure : constructor overloading

**Method Overriding:**

Two methods having exactly same signature can have different definitions when they are in different classes. This is the concept of overriding. Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class.

In this case the method in parent class is called **overridden method** and the method in child class is called **overriding method.**

Some important points to note are:

* Annotation (i.e., @override) for overriding may or may not be used as IntelliJ allowed it without using annotation. Annotations provide additional information about a program. Annotations have no direct effect on the functioning of the code they annotate
* We **cannot override the static methods** as overriding is the concept of dynamic binding and static methods are bind using static binding at compile time.

Figure :method overriding

