National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2022, Lab Manual - 07

Course Code: CL-1004	Course: Object Oriented Programming Lab
Instructor(s) :	Mr. Shoaib Rauf

Contents:

INTRODUCTION TO POLYMORPHISM

The word polymorphism means having many forms.

- Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance.
- C++ polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function.

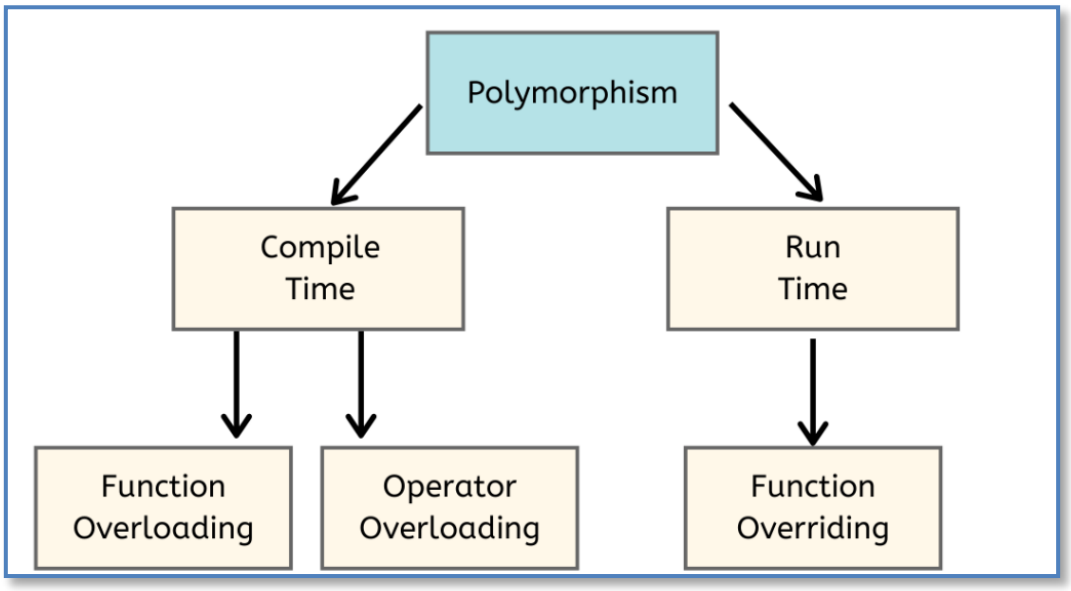
Real World Example:

- A real – life example of polymorphism is that a person at the same time can have different characteristics. A man at the same time is a father, a husband, an employee, so the same person possesses different behavior in different situations. This is called as polymorphism.
- Polymorphism is considered as one of the important features of Object Oriented Programming.

TYPES OF POLYMORPHISM:

In C++ polymorphism is mainly divided into two types:

- Compile time Polymorphism
- Runtime Polymorphism



Compile time Polymorphism:

This type of polymorphism is achieved by function overloading or operator overloading.

Function Overloading:

- When there are multiple functions with same name but different parameters then these functions are said to be overloaded.
- Functions can be overloaded by a change in the number of arguments or/and change in the type of arguments.

Example Code for Function Overloading:

```
// C++ program for function overloading
#include <bits/stdc++.h>

using namespace std;
class Geeks
{
   public:

   // function with 1 int parameter
   void func(int x)
   {
      cout << "value of x is " << x << endl;
   }

   // function with same name but 1 double parameter
   void func(double x)
```

```
    {
        cout << "value of x is " << x << endl;
    }

    // function with same name and 2 int parameters
    void func(int x, int y)
    {
        cout << "value of x and y is " << x << ", " << y << endl;
    }
};

int main() {

    Geeks obj1;

    // Which function is called will depend on the parameters passed
    // The first 'func' is called
    obj1.func(7);

    // The second 'func' is called
    obj1.func(9.132);

    // The third 'func' is called
    obj1.func(85,64);
    return 0;
}
```

Sample Run:

```
value of x is 7
value of x is 9.132
value of x and y is 85, 64
```

In the above example, a single function named ***func*** acts differently in three different situations which is the property of polymorphism.

Run time Polymorphism:

This type of polymorphism is achieved by Function Overriding.

Function Overriding:

Function overriding is a feature that allows us to have a same function in child class which is already present in the parent class.

- A child class inherits the data members and member functions of parent class, but when you want to override a functionality in the child class then you can use function overriding. It is like creating a new version of an old function, in the child class.
- To override a function you must have the same signature in the child class.

Syntax for Function Overriding:

```
 public class Parent{
access_modifier:
return_type method_name(){}
};
}
public class child : public Parent {
access_modifier:
return_type method_name(){}
};
```

Example Code for Function Overriding:

```
#include <iostream>
using namespace std;
class BaseClass {
public:
  void disp(){
    cout<<"Function of Parent Class";
  }
};
class DerivedClass: public BaseClass{
public:
  void disp() {
    cout<<"Function of Child Class";
  }
};
int main() {
  DerivedClass obj = DerivedClass();
  obj.disp();
  return 0;
}
```

Sample Run:

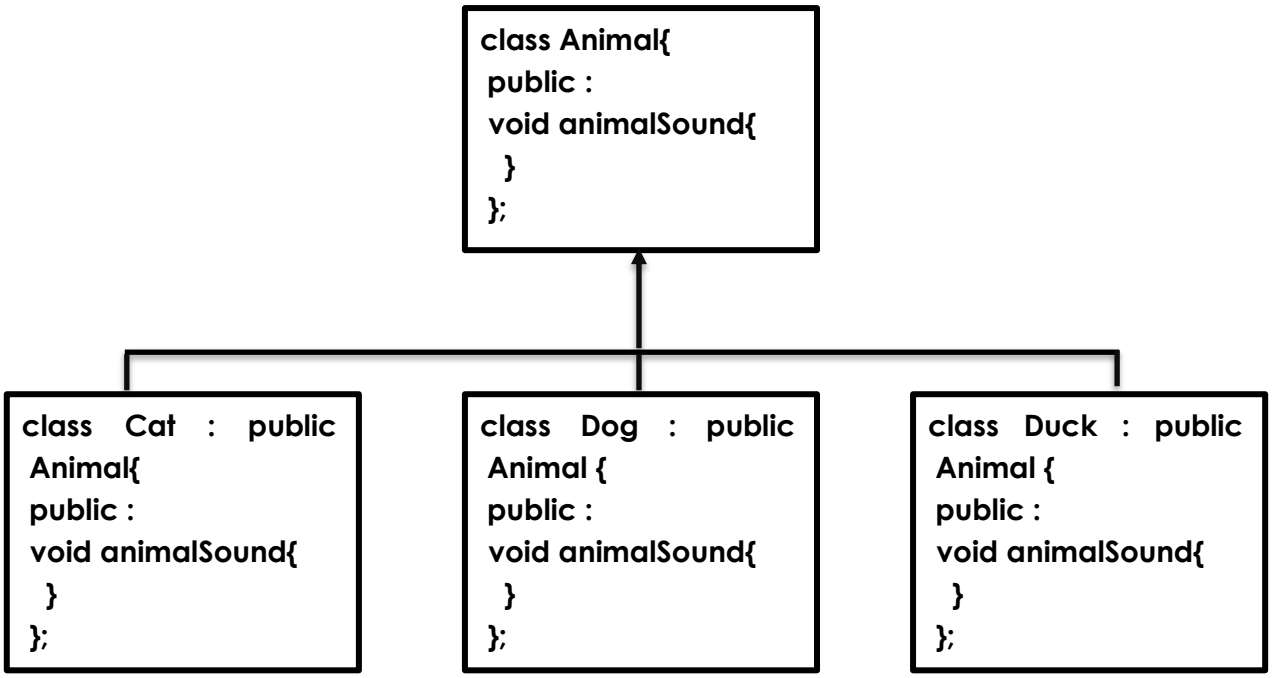
```
Function of Child Class
```

Note: In function overriding, the function in parent class is called the overridden function and function in child class is called overriding function.

LAB TASKS

Task - 01:

A school teacher is going to teach her students the different sounds of certain animals. You need to implement a program that does the following as shown in the figure:



Note the following:

- The animalSound function in the Animal class displays the text as follows: "The animal makes a sound."
- The animalSound function in the Cat class displays the text as follows: "The cat says meow."
- The animalSound function in the Dog class displays the text as follows: "The dog says bow wow."
- The animalSound function in the Duck class displays the text as follows: "The duck says quack quack."
- In the main function create objects for each of the classes and display the text for each of them.

Task - 02:

Create a base class called Car. It should have a few fields that would be appropriate for a generic car class. For example, engine, cylinders, wheels, etc. Constructor should initialize cylinders (number of) and name, and set wheels to 4 and engine to true. Cylinders and names would be passed parameters. Create appropriate getters. Create some methods like startEngine, accelerate, and brake show a message for each in the base class. Now create 3 sub classes for your favourite vehicles. Override the appropriate methods to demonstrate polymorphism in use.

Task - 03:

A company wants to calculate the bonuses of each of the employees that work in a particular department. Your services are required as a programmer to develop an automated program that will allow the company to perform their calculations. You need to create a base class called **Person** and derive two other classes named as **Admin** and **Accounts**.

- The base class will have the member functions getData, displayData and the derived class will have the member functions getData, displayData and bonus.
- The Person class will contain a data member that will store the Employee's ID.
- The Admin and Accounts contain data members that include the name of the employee and their monthly income. The bonus function will calculate the bonus of the employees. According to the company's policy each employee is awarded an annual bonus of 5%.
- Display each employee's information that includes the Employee's ID, their name, their monthly income and the bonus each one received.

Task - 04:

You are required to develop a program that allows students at a school to carry out multiplication operations with ease. To do that you need to create four different functions having the same name of your choice.

- The first function will take two integer values and return the result after multiplying them.
- The second function will take three integer values and return the result after multiplying them.
- The third function will take two decimal values and return the result after multiplying them.
- The fourth function will take three decimal values and return the result after multiplying them.
- Display the results for all the four functions to the user.

Task - 05:

Create a class called **ShapeDetails** with these private attributes:

- Area
- Parameter

Do not make any getters and setter for these, as we will be observing how friend classes will be used.

Create two classes called **Square** and **Circle** which will be friends of the class.

Class **Square** will have the following attributes:

- Side_length

Class **Circle** will have the following attributes:

- Radius_length

Both classes will have these two functions that take an object of **ShapeDetails** as input:

- Calculate_Perimeter
- Calculate_Area

In you main program you will have to make two objects for **ShapeDetails** and one object each of **Square** and **Circle**
Your task is to demonstrate how you will store the values of perimeter and area in the **ShapeDetails** object by using objects of **Square** and **Circles**