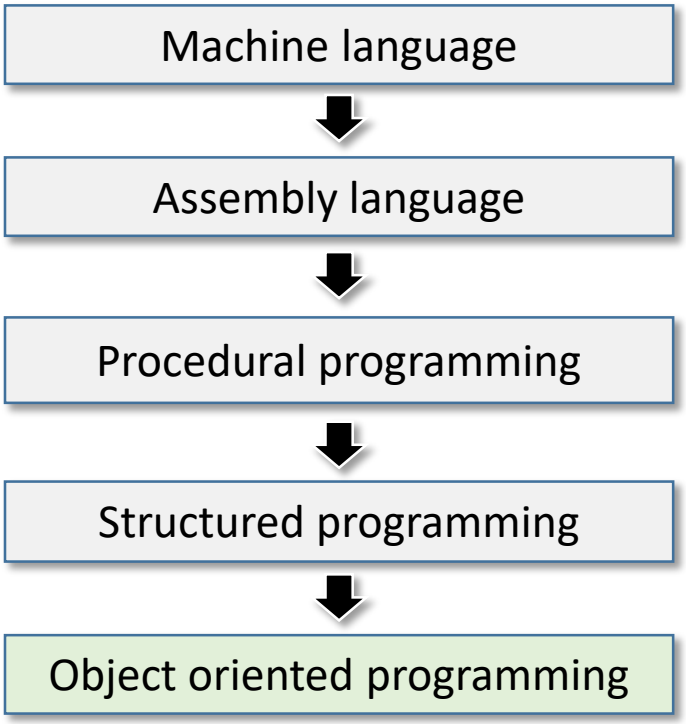
Understanding Object Oriented Design Concepts

Jinsoo Jang

Object-oriented Programming

- Object-oriented programming
  - Overcome the drawback of other methodologies, which is not closed to real world applications
  - Enable real-world modeling



Concepts - Class

❖ Class serves as a blueprint

- Define all the common properties of the different objects that belong to it
- Class can be divided into subclasses

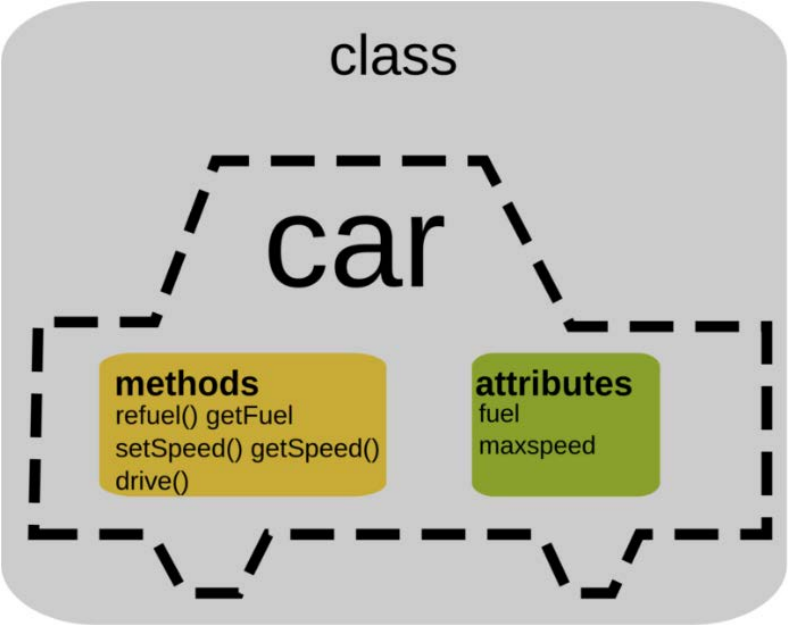


Figure from: https://brilliant.org/wiki/classes-oop/

Class in JAVA

```java
class Message {
  private String name = "Omar";
  private int age = 22;
  private double marks = 96.65;

  public void display(){
    System.out.println(name + " is " + age + " year old");
    System.out.println("He got " + marks + " marks in his exam");
  }
}
```

Class in C++

❖ Structure

```
class class_name
{
    //Class Variables or Properties
    Access-specifier:
        Data-Type x;
        Data-Type y;

    //Class Methods or Functions:
    Access-specifier:
        Return-Type function_name()
        {
            //Function Body:
        }
}
```

Keyword for creating "class"

Public, Private, Protected

Specify the type of variable

Data type to be return by method

Class in C++

❖ Example code

```
class Message
{
    //Private Members:
    private:
    string name = "Omar";
    int age = 22;
    float marks = 95.65;

    //Public Members:
    public:
    void display()  //Function to display message:
    {
        cout<<name<<" is "<<age<<" year Old: "<<endl;
        cout<<"He got "<<marks<<"% marks in his exam: ";
    }
};
```

Concepts - Object

- Instance of class
  - Specialize the class
  - e.g., object “mini” is an instance of “car” class

- Contain data and functions
  - Instance variables
  - Instance methods

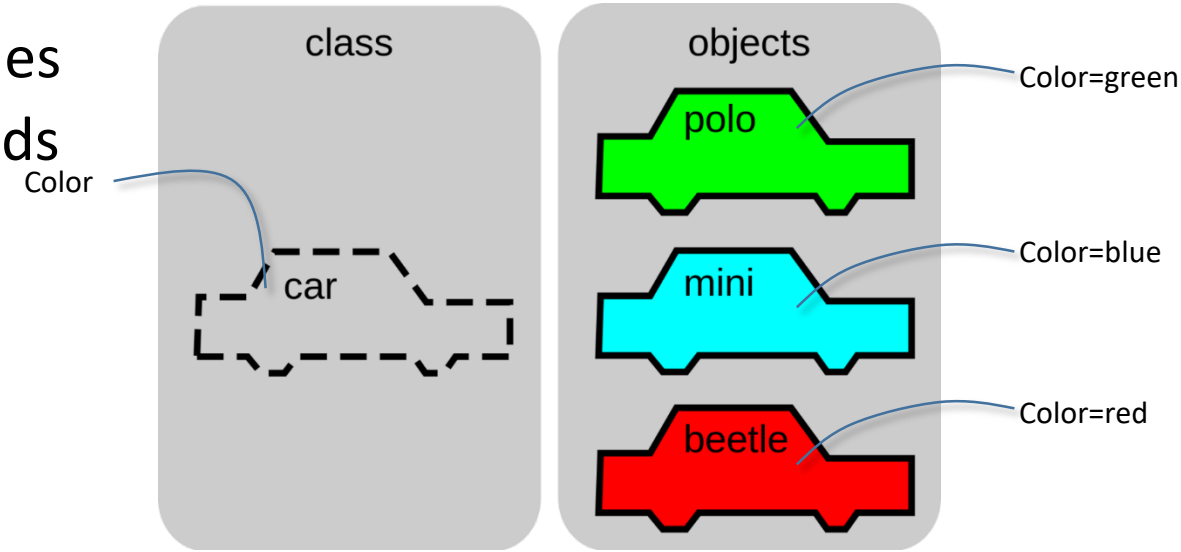


Figure from: https://brilliant.org/wiki/classes-oop/

Object in JAVA

```java
class Message {
  private String name = "Omar";
  private int age = 22;
  private double marks = 96.65;

  public void display(){
    System.out.println(name + " is " + age + " year old");
    System.out.println("He got " + marks + " marks in his exam");
  }
}
public class Main{
  public static void main(String args[]){
    Message msg = new Message();
    msg.display();
  }
}
```

Object in C++

❖ Syntax

```
 class Class-Name
{
    //Class Members:
};

main()
{
    //Creating object:
    Class-Name Class-Object;
}
```

Object in C++

❖ Example

- Create and use object

```
main()
{
    //Creating Object:
    Message msg;

    //Calling Class Public Member using Dot Operator:
    msg.display();

    return 0;
}
```

```
main() {
Message *msg = new Message();
msg->dis();
}
```

Compare!

Concepts – Abstraction

❖ Provide only essential (design) information to the outside world

- Example 1: we use a cell phone, but we don't know how the phone works
- Example 2: we use printf() function without knowing the internal working

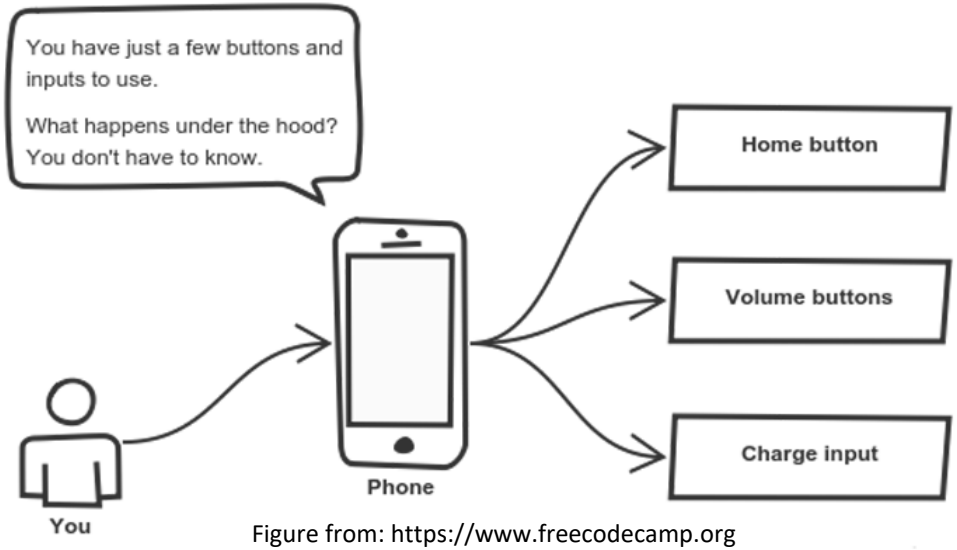


Figure from: https://www.freecodecamp.org

Concepts – Abstraction

❖ Advantage of abstraction

- Reduce the complexity of viewing the things
- Code duplication is avoided (reusability)
- Protection of internal implementation details (security)

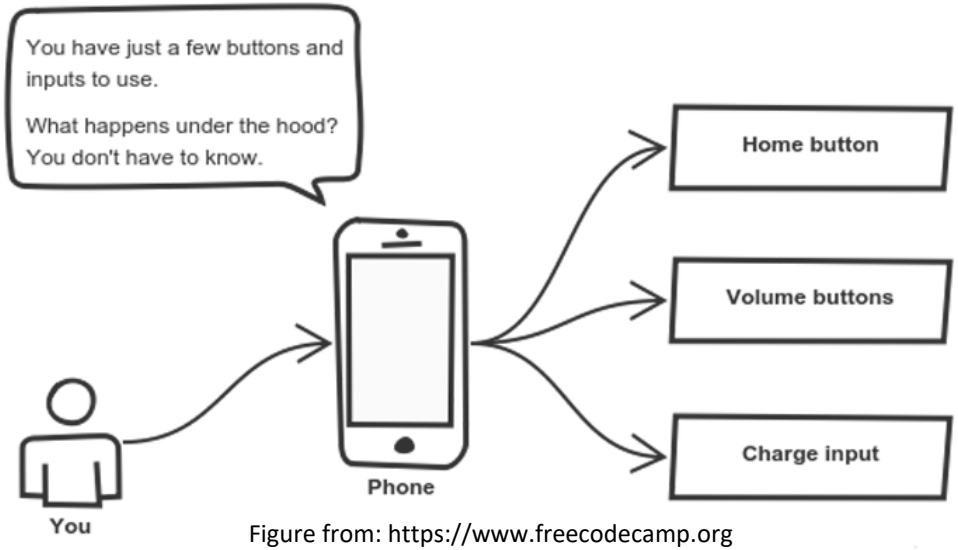


Figure from: https://www.freecodecamp.org

Abstraction in C++

```cpp
#include <iostream>
using namespace std;

class Summation {
private:
  // private variables
  int a, b, c;
public:
  void sum(int x, int y)
  {
    a = x;
    b = y;
    c = a + b;
    cout<<"Sum of the two number is : "<<c<<endl;
  }
};
int main()
{
  Summation s;
  s.sum(5, 4);
  return 0;
}
```

No need to know the details (e.g., variables - a, b, c) from outside!

Concepts- Encapsulation

- Data and functions are bound and isolated in a black box (class).
- Information hiding

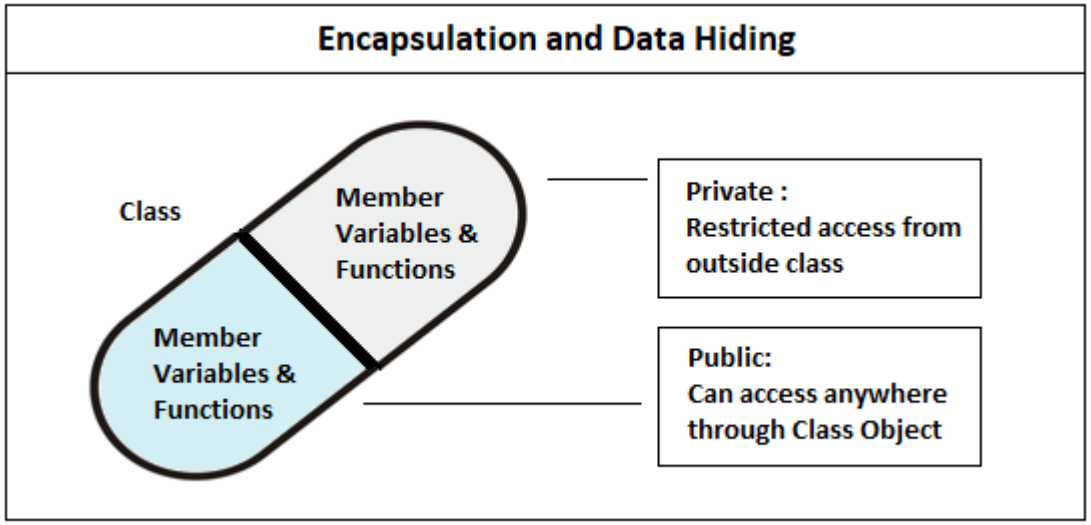


Figure from: ww.cpp.thiyagaraaj.com

Encapsulation in JAVA

```java
public class EncapTest {
    private String name;
    private String idNum;
    private int age;

    public int getAge() {
        return age;
    }
    public String getName() {
        return name;
    }
    public void setAge( int newAge) {
        age = newAge;
    }
    public void setName(String newName) {
        name = newName;
    }
}
```

```java
public class RunEncap {

    public static void main(String args[]) {
        EncapTest encap = new EncapTest();
        encap.setName("James");
        encap.setAge(20);

        System.out.print("Name : " + encap.getName() +
        " Age : " + encap.getAge());
    }
}
```

Encapsulation in C++

❖ Example

```
class addition //class begin
{
    private: //Private Members: (data Member)
        int UpperLimit;
        int sum;
        int Number;

    public: //Public Members: (Method)
       addition()
        {
            //body
        }

       void showResult()
       {
           //body
       }
};
```

Encapsulation in C++

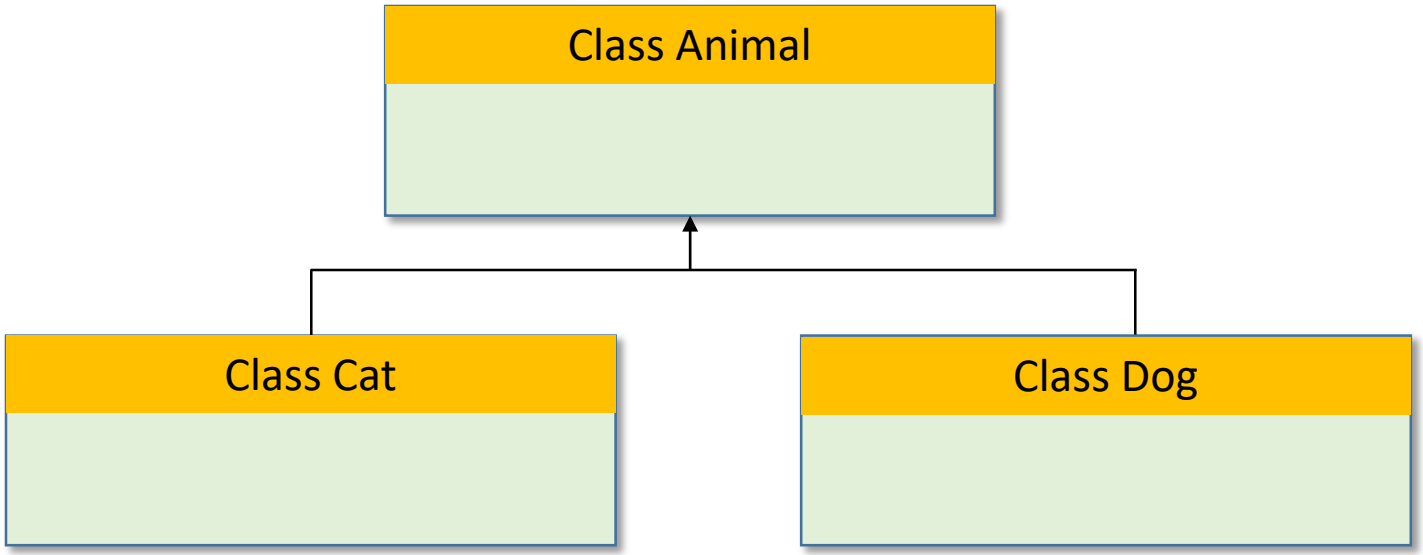
❖ Example

Encapsulation: a mechanism of bundling the data, and the functions that use them

```
/* Data Encapsulation: this Program can return you sum of all numbers till
UpperLimit. e.g. Upperlimit is 5, then it will return 5+4+3+2+1+0=15 */

#include<iostream>
using namespace std;

class addition
{
    private:
        int UpperLimit; //to get Upper Limit from User.
        int sum;        //will use to store the Sum of Numbers.
        int Number;     //all Numbers less than upperlimits.

    public:
        addition()
        {
            UpperLimit=0;
            sum=0;
            Number=0;
        }

        void showResult()
        {
            cout<<" Enter the Upper Limit to get sum: ";
            cin>>UpperLimit;

        /*Loop will be continue until reach at UpperLimit Number.*/
            while(Number<UpperLimit) //Number is less than UpperLimit.
              {
                Number=Number+1; //Increment number one by one.
                sum=sum+Number; /*All numbers will added to sum and will
                                  store in sum variable.*/
                               //Example. sum=0+1=1; sum=1+1=2 and so on.
              }

     //when the condition has gone false, mean Number will reach upperlimit.
             cout<<"The sum of first "<<UpperLimit
                                    <<" Number are: "<<sum;
        }
};

main()
{
    addition obj;
    obj.showResult();

    return 0;
}}
```

Concepts- Inheritance

- Obtain data and function from one class to another class
- Advantage
  - Save time and memory, less storage is required



Inheritance in JAVA

```java
class Vehicle {
  protected String brand = "Ford";
  public void honk() {
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";
  public static void main(String[] args) {
    Car myFastCar = new Car();
    myFastCar.honk();
    System.out.println(myFastCar.brand + " " + myFastCar.modelName);
  }
}
```

Inheritance in C++

❖ Syntax

Public
Private
protected

```
class Derived_Class: Access-Specifier Base_class
{
    //body of the Derived class.
}
```

Inheritance in C++

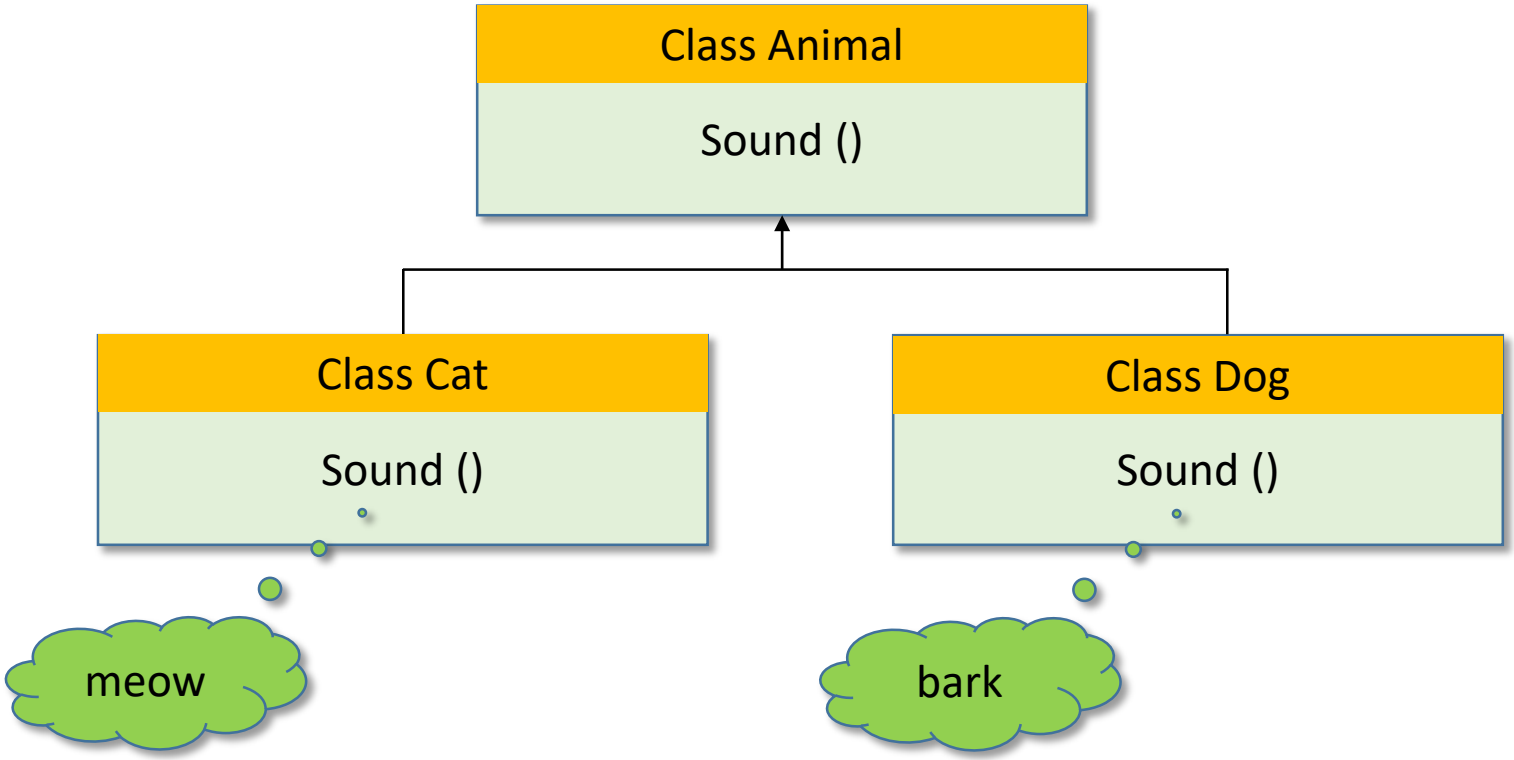
❖ Example

```
class TwoD_Shape //Base Class:
{
   public: //Public Members:
    double width;
    double height;

   //this function will Show dimensions:
    void showDim()
    {
            cout<<" Width and height are " <<width<< " and "
                                <<height<<endl;
    }
};
```

```
/*Base Class(i.e. TwoD_Shape) is Inherited by
  derived class(i.e. Triangle)*/
class Triangle: public TwoD_Shape
{
   public:
       char style[20];
       double area()
       {
         //using Public Members of Base Class.
           return width * height / 2;
       }

        void showStyle()
        {
            cout<<" Triangle is " <<style<<endl;
        }
};
```

Concepts- Polymorphism

❖ Allow one interface to access a general class of actions



Concepts- Polymorphism cont'd

- Two types of Polymorphism
  - Compile time polymorphism
    - Method Overloading
    - Method Overriding
  - Run-time polymorphism
    - Using virtual function

Polymorphism in JAVA

❖ Method overloading

```
1   class DisplayOverloading
2   {
3       public void disp(char c)
4       {
5           System.out.println(c);
6       }
7       public void disp(char c, int num)
8       {
9           System.out.println(c + " "+num);
10      }
11  }
12  class Sample
13  {
14      public static void main(String args[])
15      {
16          DisplayOverloading obj = new DisplayOverloading();
17          obj.disp('a');
18          obj.disp('a',10);
19      }
20  }
```

Polymorphism in JAVA

❖ Method overriding

```java
class Parent {
    void display() { System.out.println("super class display()"); }
}

class Child extends Parent {
    void display() { System.out.println("child class display()"); }
}

public class Main {
    public static void main(String[] args) {
        Parent pa = new Parent();
        pa.display();
        Child ch = new Child();
        ch.display();
        Parent pc = new Child();
        pc.display();
    }
}
```

```
super class display()
child class display()
child class display()
```

Polymorphism in C++

❖ Compile time polymorphism

- Method Overloading

```
#include<iostream>
using namespace std;

class math //class.
{

    public:
        void result(int x, int y) //Method to show result.
        {
            cout<<" The result is : "<<x+y;
        }

        void result(int x, int y, int z) //Method overloading with extra parameter.
        {
         cout<<" The result is: "<<x+y+z;
        }
};

main()
{
    math obj; //Object of type math.

    obj.result(10,15); //1st method Called.
    cout<<endl;

    obj.result(10,5,15); //overload method called.
    return 0;
}
```

Polymorphism in C++

❖ Compile time polymorphism

- Method Overriding

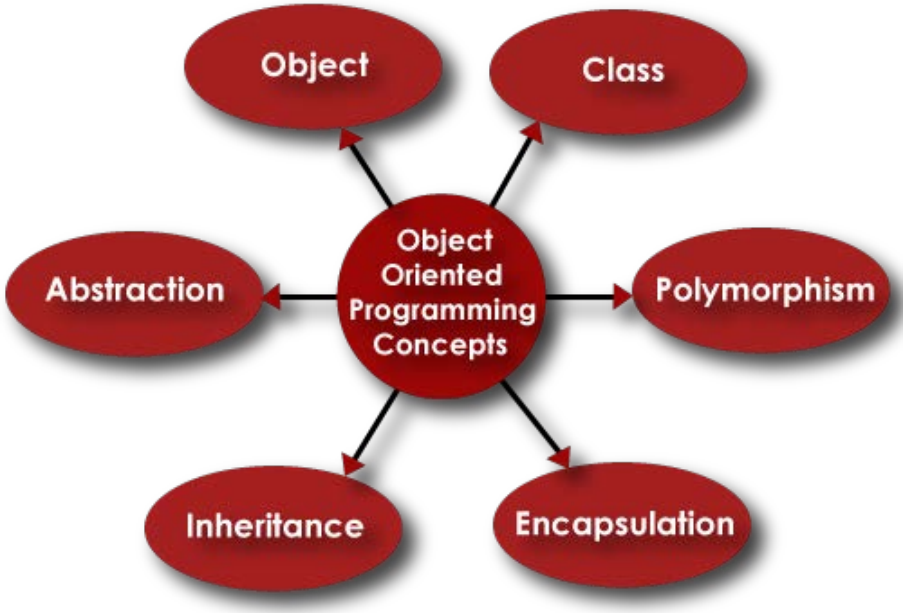
```
#include<iostream>
using namespace std;

class base //base class.
{
    public:
        void output() //simple method to print something.
        {
            cout<<" Its Base Class: "<<endl;
        }
};

class derived: public base //derived a class from class base.
{
    public:
        void output() //overridden method.
        {
            cout<<" Its Derived Class: "<<endl;
        }
};

main()
{
    base parent;   //object of type base.
    derived child; //Object of type derived.

    parent.output();   //1st method called.
    cout<<endl;

    child.output();   //overridden method called.

    return 0;
}
```

It's Base Class:

It's Derived Class:

Task

Let's find commonly used applications that might be developed with OOP concepts. Discuss which part of application can use the OOP.



https://blog.usejournal.com/object-oriented-programming-concepts-in-simple-english-3db22065d7d0?gi=c34d5b9c0acd

References

- *geeksforgeeks.org*
- *modoocode.com*
- *www.learncpp.com*
- *sourcemaking.com*
- www.infobrother.com
- *Lecture Slides for Programming in c++, google books*