CLASS WORK

Object Oriented Programming

Lecture 3

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

What is Object Oriented Programming?

CLASS WORK

An object is like a black box.

The internal details are hidden.

- Identifying *objects* and assigning *responsibilities* to these objects.
- Objects communicate to other objects by sending *messages*.
- Messages are received by the *methods* of an object

What is an object?

- Tangible Things as a car, printer, ...
- Roles as employee, boss, ...
- Interactions as contract, sale, ...
- Specifications as colour, shape, …

Tahir Nawaz
Assistant Professor
UOL Sargodha Campus

CLASS WORK

Object Oriented Programming

CLASS WORK

- Programmer who defines the attributes and behavior of objects
- Objects are modeled according to real-world entities
- Different approach from structure programming such as C language

Object Oriented Programming

- Object-oriented programming (OOP)
  - Encapsulates data (attributes) and functions (behavior) into packages named classes.

- Classes are user-defined types.
  - Data Members
  - Member Functions or Methods)

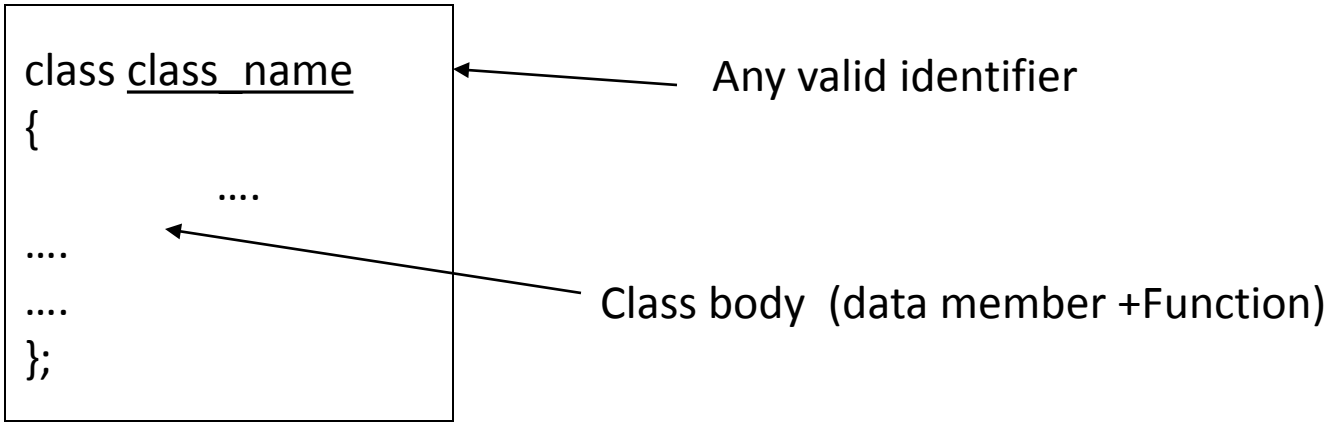
Reasons for OOP

1. Simplify programming
2. Interfaces
   - Information hiding:
     - Implementation details hidden within classes themselves
3. Software reuse
   - Class objects included as members of other classes

CLASS WORK

Classes in C++

- A class definition begins with the keyword *class*.
- The body of the class is contained within a set of braces, { } ; (notice the semi-colon).



Classes C++

- Within the body, the keywords *private:* and *public:* specify the access level of the members of the class.
  - Default is private

- Usually, the data members of a class are declared in the *private:* section of the class and the member functions are in *public:* section

CLASS WORK

Classes in C++

CLASS WORK

```
class class_name
{
    private:
            …
            …
            …
    public:
            …
            …
            …
};
```

private members or methods

Public members or methods

Classes in C++

- Member Access Specifiers
  - public:
    - Can be accessed outside the class directly
      - The public is used for *interface*
  - private:
    - Accessible only to member functions of class
    - Private members, methods are for internal use only

CLASS WORK

Class Example

- This class example shows how we can encapsulate (gather) a circle information under single name (unit or class)

```
class Circle
{
    private:
            double radius;
    public:
            void setRadius(double r);
            double getDiameter();
            double getArea();
            double getCircumference();
};
```

No need for others classes to access and retrieve its value directly. The class methods are responsible for that only.

They are accessible from outside the class, and they can access the member (radius)

Declaring an object of a Class

CLASS WORK

- Declaring a variable of a class type creates an object. You can have many variables of the same type (class).
  - *Instance*
- Once an object of a certain class is instantiated, a new memory location is created for it to store its data members and code
- You can instantiate many objects from a class type.

  e.g
  - Circle obj1; Circle obj2;

Procedural Programming:
a sequence of 'Procedures'

```
int main()
{
  int x,y,z;
  int a,b,c;


        a=f1(x);
        b=f2(y);
        c=f3(z);
        …

}

int f1()
{
}

int f2()
{
}

int f3()
{
}
```

VS

Object Oriented Programming:
a sequence of 'Objects'

```
int main()
{
  A a;
  B b;
  C c;

  a.f1();
  b.f2();
  c.f3();
   …
}

Class A
{
Int x;
Int f1();
}

Class B
{
Int y;
Int f2()
}

Class C
{
Int z;
Int f3();
}
```

CLASS WORK

Constructors

- Same name as class
- No return type (not even void!)
- Can be overloaded
  - Same name, differ by arguments they take
  - One with no arguments is “default” constructor
- Use of constructors
  - Initialization
  - Sometimes objects need to have values or perform some operation before they can be used

CLASS WORK

Destructors

- Only one per class
- Class name with ~ in front
- Doesn’t take any arguments
- Controls what happens when object destroyed
- Called automatically

CLASS WORK

Use of destructor

- When is destructor useful?
  - Executed when object destroyed
  - Can do anything, but interesting when deallocate memory
    - Want to delete items created using new to free up memory

CLASS WORK