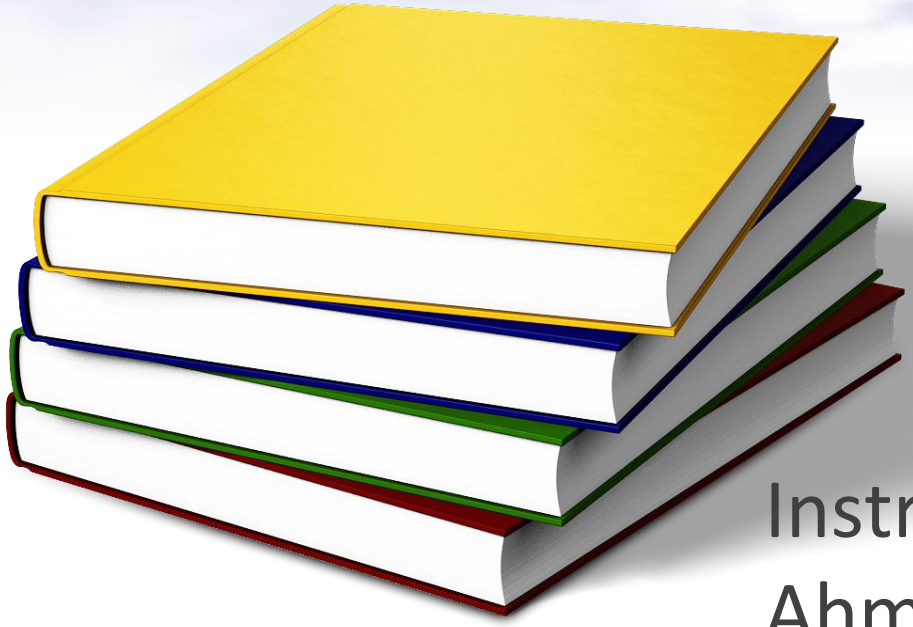
Chapter # 6
Objects & Classes



Instructor: Engr. Nauman
Ahmad Tariq

Objectives

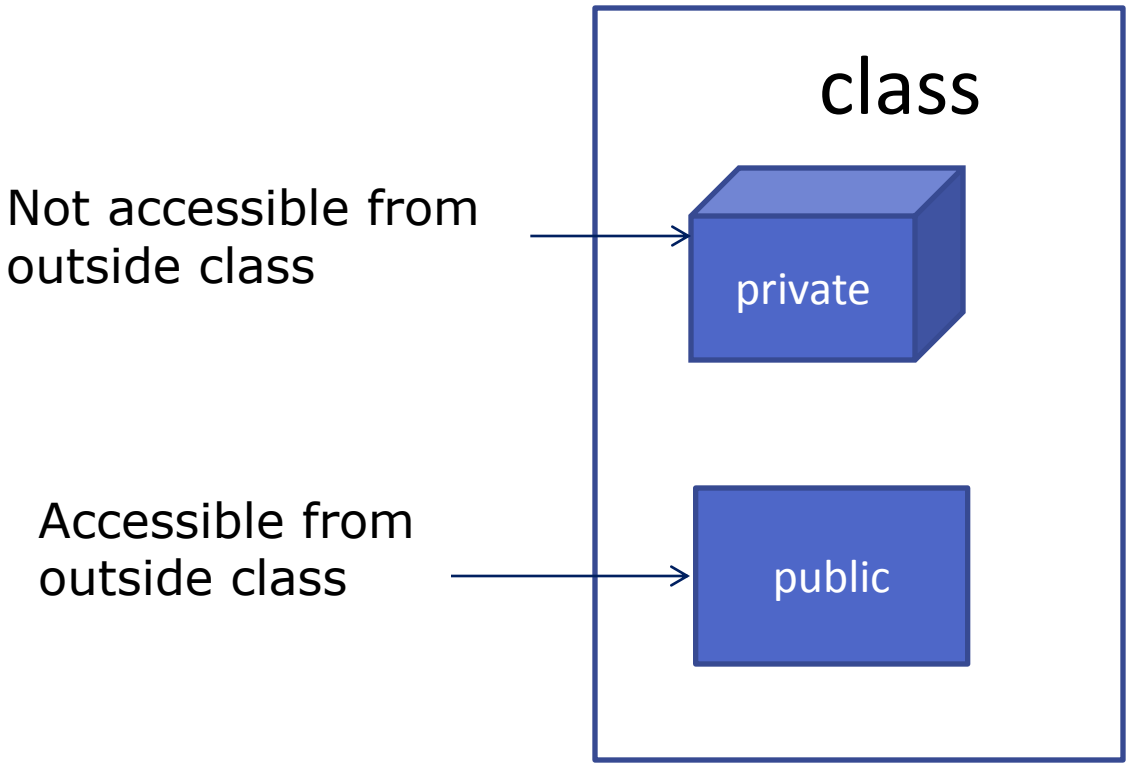
- Member functions & data.
- *private & public*
- Constructors & Destructors.
- Objects in the real world.
- When to use Objects

Class

Syntax

```
class class_name
{
private:
data1
data2
…
public:
function1()
function2()
…
}
};
```

public & private



Simple Class

```
#include<iostream>
#include<conio.h>
using namespace std;
class data //class declaration
{
int somedata;
public:
void setdata(int d)
{
somedata=d;
}
void showdata()
{
cout<<"data="<<somedata;
}
};
int main ()
{
data d1;  //objects
d1.setdata(100); //function call
d1.showdata();
getche();
return 0;
}
```

Objects as Data Types

```
#include<iostream>
#include<conio.h>
using namespace std;
class part
{
int partnumber;
int salary;
public:
void setpart(int n,int s)
{
partnumber=n;
salary=s;
}
void showpart()
{
cout<<"\npart
number="<<partnumber;
cout<<"\nSalary="<<salary;
}
};
int main ()
{
part p1;
p1.setpart(10,10);
p1.showpart();
getche();
return 0;
}
```

Constructors & Destructors

```
#include<iostream>
#include<conio.h>
using namespace std;
class counter
{
unsigned int count;
int salary;
public:
counter():count(0)
//constructor
{ }
counter(int n)  //overloaded
constructor
{
count=n;
}
~counter()  //destructor
{
cout<<"good bye";
}
void inc()
{
count++;
}
int getcount()
{
return count;
} };
```

Contd…

```
int main ()
{
counter c1;
counter c2(4);
cout<<"c1="<<c1.getcount();
cout<<"\nc2="<<c2.getcount();
c1.inc();
c1.inc();
c2.inc();
cout<<"\nc1="<<c1.getcount();
cout<<"\nc2="<<c2.getcount();
getche();
return 0;
}
```

Objects as Function Arguments

```
#include<iostream>
#include<conio.h>
using namespace std;
class Distance
{
int feet;
float inches;
public:
Distance():feet(0),inches(0.0)
{ }
Distance(int ft,float in):
feet(ft),inches(in)
{ }
void getdist()
{
cout<<"\nEnter feet:"; cin>>feet;
cout<<"\nEnter inches:";
cin>>inches;
}
void showdist()
{
cout<<feet<<"\'-"<< inches<<"\"";
}
void add(Distance,Distance);
//Objects as arguments
};
```

Contd…

```
void Distance::add(Distance d1,Distance
d2) {
inches=d1.inches+d2.inches;
feet=0;
if(inches>12)
{
inches-=12;
feet++;
}
feet+=d1.feet+d2.feet;
}
```

```
int main ()
{
Distance dist1(11,2.3);
Distance dist2,dist3;
dist2.getdist();
dist3.add(dist1,dist2);
cout<<"\ndist1=";dist1.showdist();
cout<<"\ndist2=";dist2.showdist();
cout<<"\ndist3=";dist3.showdist();
getche();
return 0;
}
```

The Default Copy Constructor

```
....
Distance(): feet(0),inches(0.0)
{ } //zero argument
Distance(int ft,float in)
{
feet =ft;
inches=in;
}  //two arguments
```

```
int main
{
Distance dist1;
Distance dist2(6,3.6);
Distance dist3=dist1; //one
argument
Distance dist4(dist2); //one
argumnet
```

Returning Objects From Function

```
Distance Distance::add(Distance d1)
{
Distance temp;
temp.inches=d1.inches+inches;
if(temp.inches>12)
{
temp.inches-=12;
temp.feet++;
}
temp.feet+=d1.feet+feet;
return temp;
}
```

```
int main ()
{
Distance dist1(11,2.3);
Distance dist2,dist3;
dist2.getdist();
dist3=dist1.add(dist2);
…
}
```

Static Class Data

```
#include<iostream>
#include<conio.h>
using namespace std;
class count
{
static int c;
public:
count()
{
c++;
}
int getcount()
{
return c;
}
 };
int count::c=0;
int main ()
{
count co1,co2,co3;
cout<<"\nCount="<<co1.getcount();
cout<<"\nCount="<<co2.getcount();
cout<<"\nCount="<<co3.getcount();
getche();
return 0;
}
```

const & Classes

```
#include<iostream>
#include<conio.h>
using namespace std;
class Distance
{
int feet;
float inches;
public:
Distance():feet(0),inches(0.0)
{ }
Distance(int ft,float in):
feet(ft),inches(in)
{ }
void getdist()
{
cout<<"\nEnter feet:"; cin>>feet;
cout<<"\nEnter inches:";
cin>>inches;
}
void showdist() const
{
cout<<feet<<"\'-"<< inches<<"\"";
}
void add(Distance,Distance) const;
//Objects as arguments
};
```

Contd…

```
void Distance::add(Distance d1,Distance d2) const
 {
inches=d1.inches+d2.inches;
//feet=0;   ERROR
if(inches>12)
{
inches-=12;
feet++;
}
feet+=d1.feet+d2.feet;
}
```

```
int main ()
{
Distance dist1(11,2.3);
Distance dist2,dist3;
const Distance dist4;
//dist4.getdist();   ERROR
dist2.getdist();
dist3.add(dist1,dist2);
cout<<"\ndist1=";dist1.showdist();
cout<<"\ndist2=";dist2.showdist();
cout<<"\ndist3=";dist3.showdist();
getche();
return 0;
}
```