Constructor

Mohammad Tasin

Agenda

- What is Constructor .?
- Parameterized Constructor.
- Constructor overloading.
- Default constructor.
- Copy constructor.
- Shallow copy vs Deep copy

What is Constructor .?

- Constructor is a special member of the class whose name is same as the name of the class.
- Constructor has no return type.
- Constructor is invoked at the time of object creation (automatically).
- Constructor is an instance member.
- Usually constructor is defined as public member but it can be private also.

```
class Item
{
    private:
        int a;
    public:
        Item() { }
};
```

→ constructor

Parameterized Constructor.

- You can make a constructor with arguments
- constructor arguments are passed at the object creation.

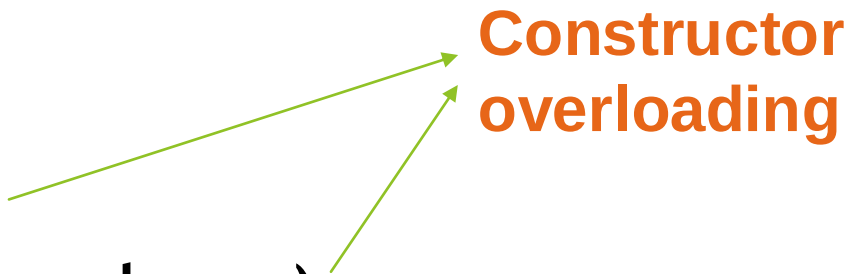
```
class Item
{
    private:
        int a;
    public:
        Item(int x) {a = x; }
};
int main()
{
    Item i1(10);
}
```

→ Parameterized Constructor

Constructor overloading

- Programmer can provide multiple Constructors in the class with different signatures

```
class Item
{
    private:
        int a;
        int b;
    public:
        Item(int x) {a = x; }
        Item(int x, int z) (a = x; b = z;)
};
int main()
{
    Item i1(10),i2(20,30);
}
```



Default constructor

- When programmer doesn't provide explicit constructor in the class, compiler create an empty body, no argument constructor in the Class

```
class Item
{
    private:
        int a;
    public:
        Item() {    }
};
```

Default constructor

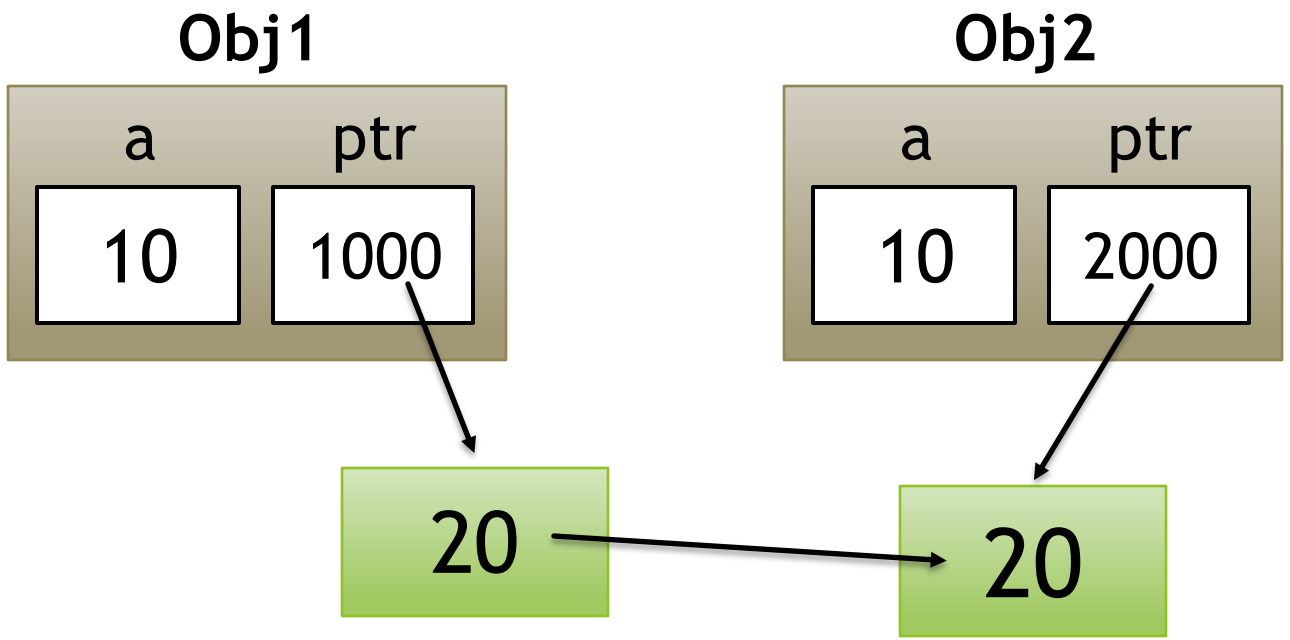
Copy constructor.

- Either programmer has to provide copy constructor in the class or compiler itself provides copy constructor.
- Copy constructor is invoked for newly created object which is initialized with the object of the same class
- Formal argument of copy constructor must be a reference variable of same class

```
ClassName(ClassName &obj);
```
→ Declaration

```
ClassName obj;
ClassName obj1 = obj;
```
→ Call

Shallow copy vs Deep copy




```
obj2 = obj1;
```

```
Item(Item &obj)     {
        a = obj.a;
        ptr = new int;
        *ptr = *(obj.ptr);
}
```