string builder

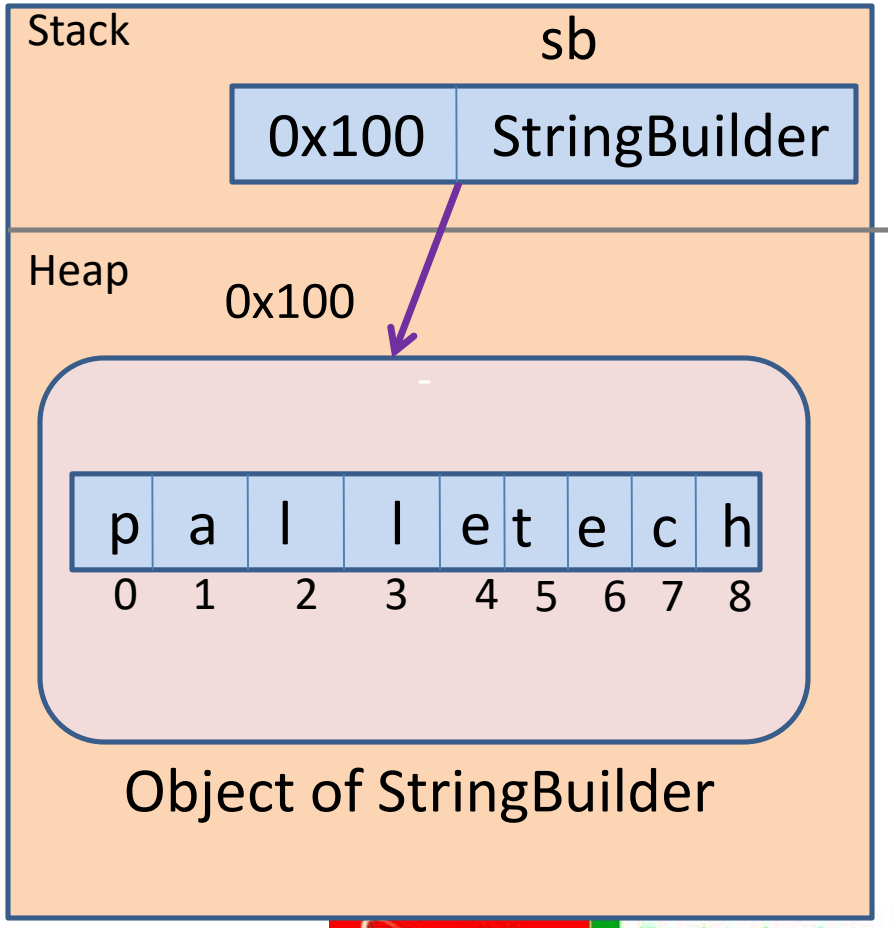
- string builder is an inbuilt class and its code is written by microsoft programmers
- using string builder we can modify the string data hence it is considered as mutable.

stringbuilder internals

LHS RHS

```
StringBuilder sb = new StringBuilder();
sb.Append("palle");
sb.Append("tech");
Console.WriteLine(sb);
Console.ReadLine();
```

Output:
palletech



Advantage:
1.using SB we can modify string data
2.Since SB are mutable, new strings are not created and hence memory wastage is reduced

predefined code in.net

- .net comes with lot of pre defined code.
- mostly predefined code contains classes & methods.

- Note: we can’t use a class present in .net without adding corresponding name space in the .cs file header

finding namespace while using class

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
            StreamReader streamReader = new StreamReader("C:\\students.txt");
        }
    }
}
```

Generate class 'StreamReader'

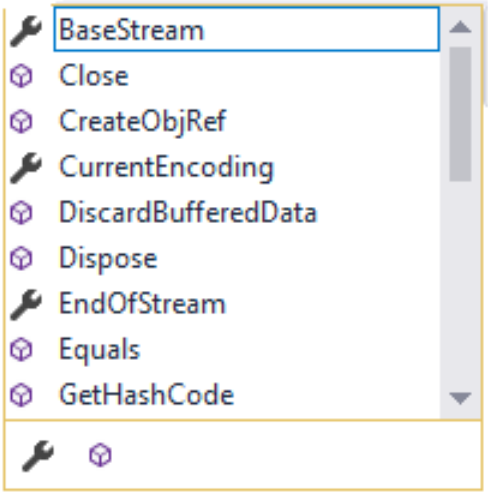
Generate nested class 'StreamReader'

Generate new type...

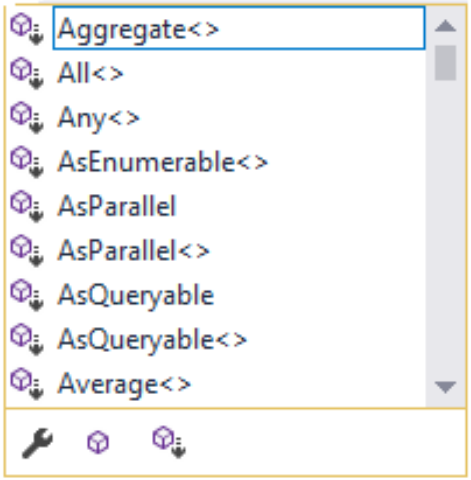
resent

finding methods in predefined class

```
StreamReader streamReader = new StreamReader("C:\\students.txt");
streamReader.
```




```
string name = "Dr.Proton";
name.
```



finding namespaces assignment

ListDictionary
StreamReader
SqlConnection
Stack
Calendar

constructor

by using constructors we can assign data into objects/we can initialize objects

Syntax :

```
public class class_name
{
    public class_name( dt v1,dt v2,..... )
    {

    }
}
```

1.Constructor name must be same as class name

2.Constructor must not contain return type

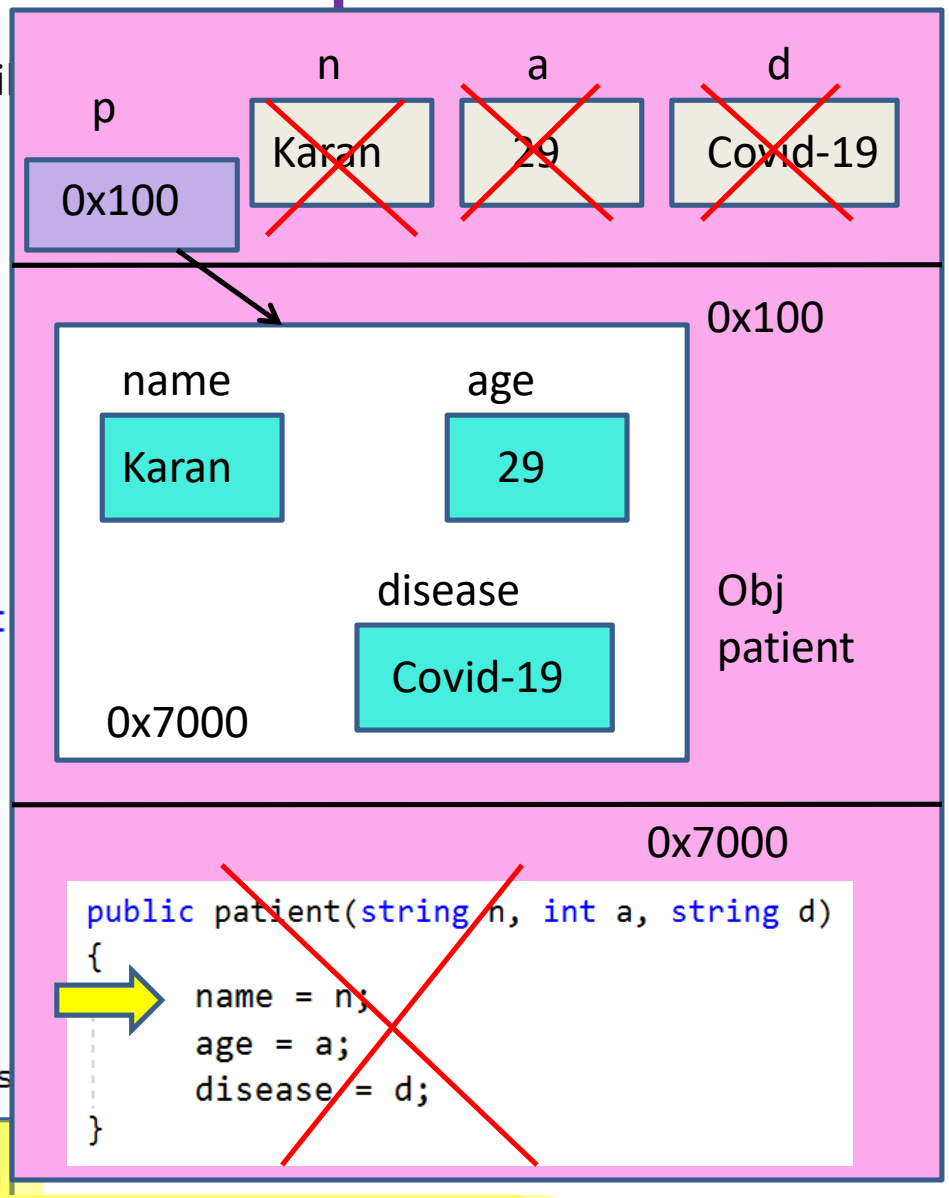
3.We can create any number of constuctors in a class

4.Constructors can contain 0 or more parameters

constructor sample

Req: I would like to store one patient details li
name, age and disease

```
public class Patient
{
    public string name;
    public int age;
    public string disease;

    public patient(string n, int a, st
    {
        name = n;
        age = a;
        disease = d;
    }
}
class Program
{
    static void Main(string[] args
    {
        patient p = new patient();
        patient p = new patient("karan",29,"covid-19");
```



Technologies

constructor sample

Req: I would like to store 3 patient details age and disease

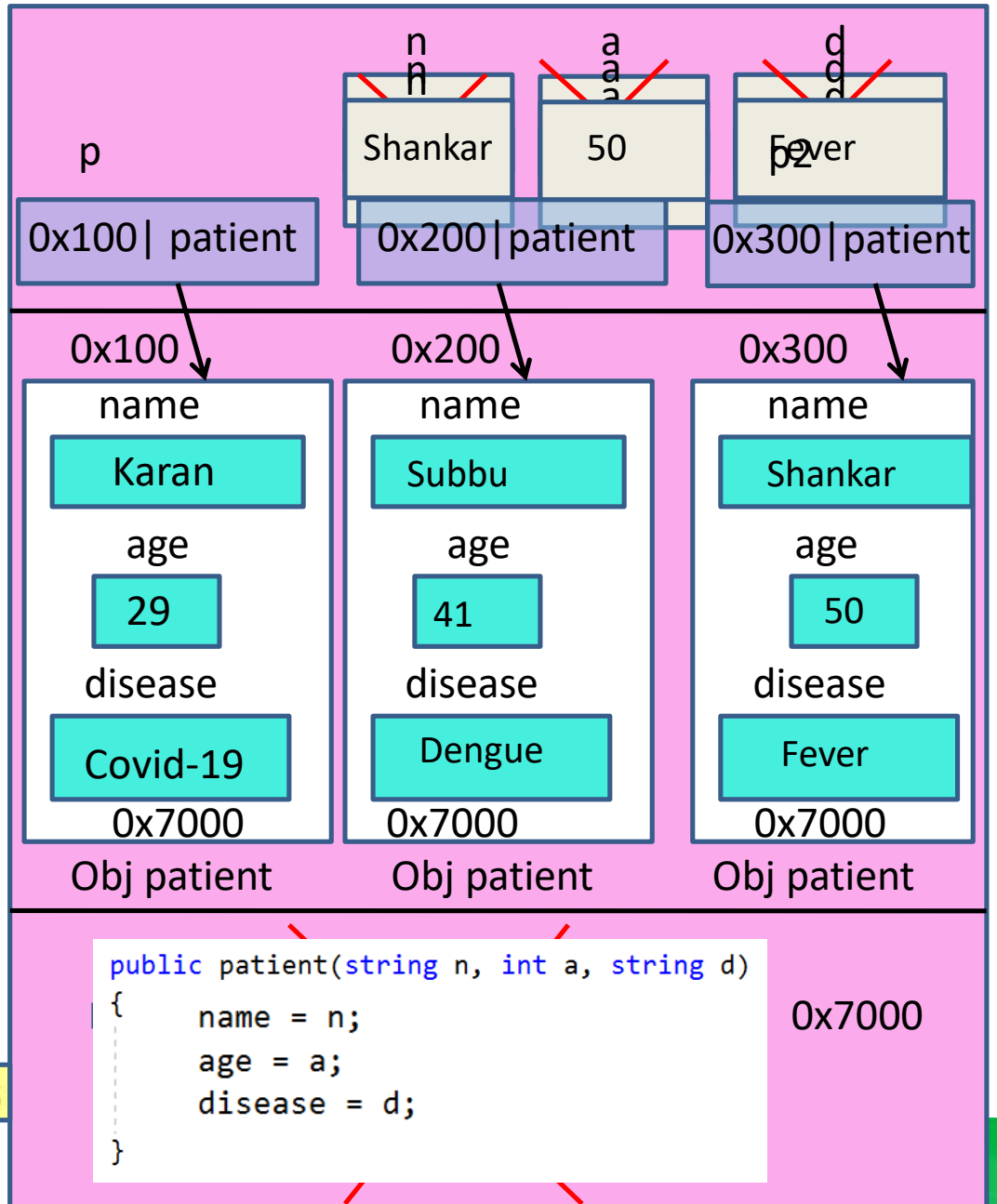
Stack

```
public class Patient
{
  public string name;
  public int age;
  public string disease;

  public patient(string n, int a, string d)
  {
    name = n;
    age = a;
    disease = d;
  }
}

class Program
{
    static void Main(string[] args)
    {
      patient p = new patient("karan",29,"covid-19");
      patient p1 = new patient("subbu", 41, "Dengue");
      patient p2 = new patient("Shankar", 50, "Fever");
    }
}
```

Heap

Code segment

RAM



OOP(Object Oriented Programming)

any language which supports following features called as OOPL

1. Encapsulation
2. Abstraction
3. Inheritance
4. polymorphism

These all are examples for OOPL

Encapsulation

Grouping related Variables and Methods in a common related place.

```
public class patient
{
  public string name = "Raj";
  public int exp = 10;
  public string sug_medicine(string disease)
  {
      return "paracetamol";
  }
}
```

```
public class Doctor
{
    public string name = "Ram";
    public int age = 40;
    public string disease = "Viral fever";
    public void Dosurgery(string disease)
    {

    }
}
```

Is it good to have suggest medicine method in a patient class?

Is it good to have exp variable in a patient class??

Is it good to have disease variable in a doctor class?

```
public class patient
{
  public string name = "Raj";
  public int age = 40;
  public string disease = "Viral fever";
}
```

```
public class Doctor
{
  public string name = "Ram";
  public int age = 40;
public string sug_medicine(string disease)
{
    return "paracetamol";
}
public void Dosurgery(string disease)
{

}
}
```

We are binding variables and methods in appropriate place

Abstraction

Def: Hiding implementation details and showing the required details to the users

Abstraction sample

```
string s = "safali balala";
```

Req: replace “la” with “na” in s

```
s.Replace("la", "na");
```

Who has written code for replace() method

Micro soft programmers has written code for replace method
We are just using the methodname,method parameters,
method return type

Is it possible to see the implementation details of replace method/
Is it possible to see the code present in replace method body

Not possible, and no need to see the implementation details

Right click on Replace and go to def

```
s.Replace("la", "na");
```

```
public String Replace(String oldValue, String newValue, bool ignoreCase, CultureInfo culture);
public String Replace(String oldValue, String newValue, StringComparison comparisonType);
public String Replace(String oldValue, String newValue);
public String Replace(char oldChar, char newChar);
```

Inheritance

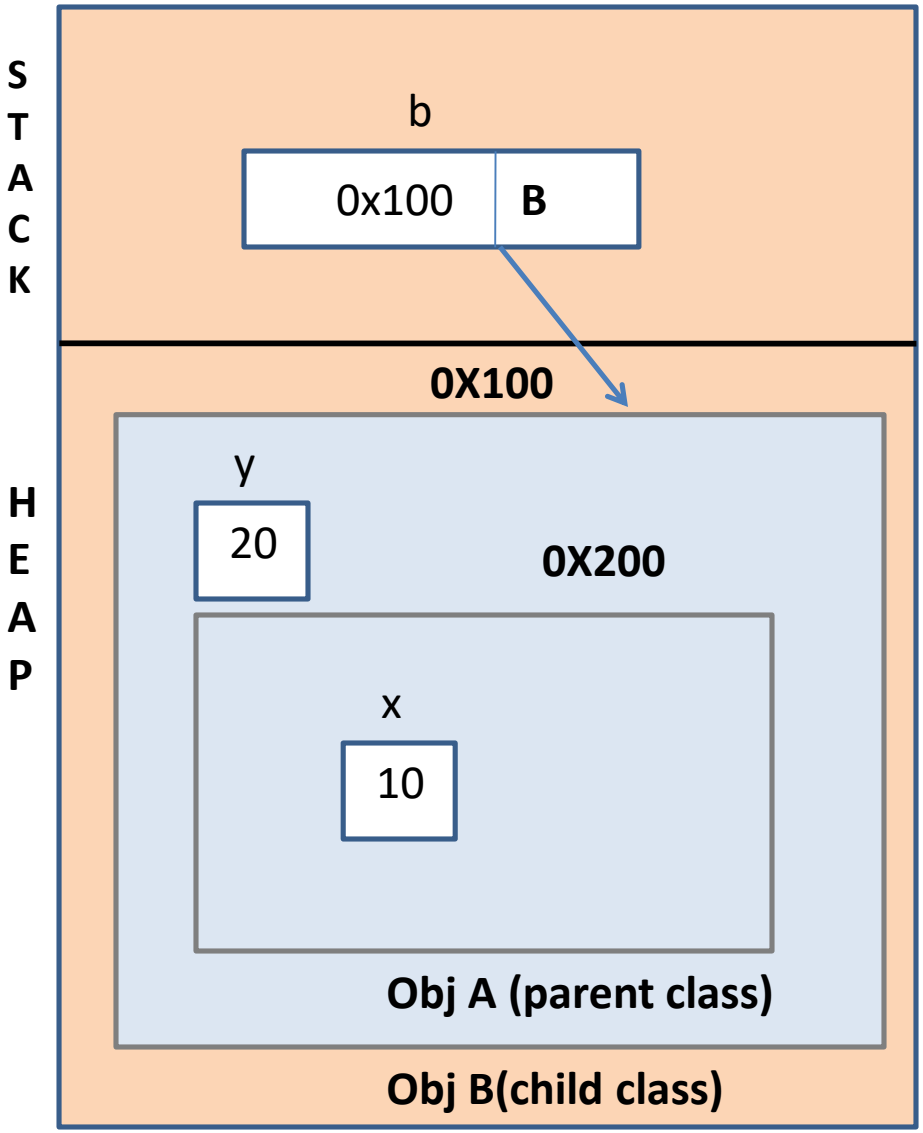
1. Inheritance reduce code duplication (code redundancy)
2. Inheritance improves maintainability.

Inheritance RAM architecture

```
class Program
{
    static void Main(string[] args)
    {
        B b = new B();
        Console.WriteLine(b.x);
        Console.WriteLine(b.y);
    }
}
public class A
{
    public int x = 10;
}
public class B:A
{
    public int y = 20;
}
```

output

10
20



Which class data will be loaded first????????

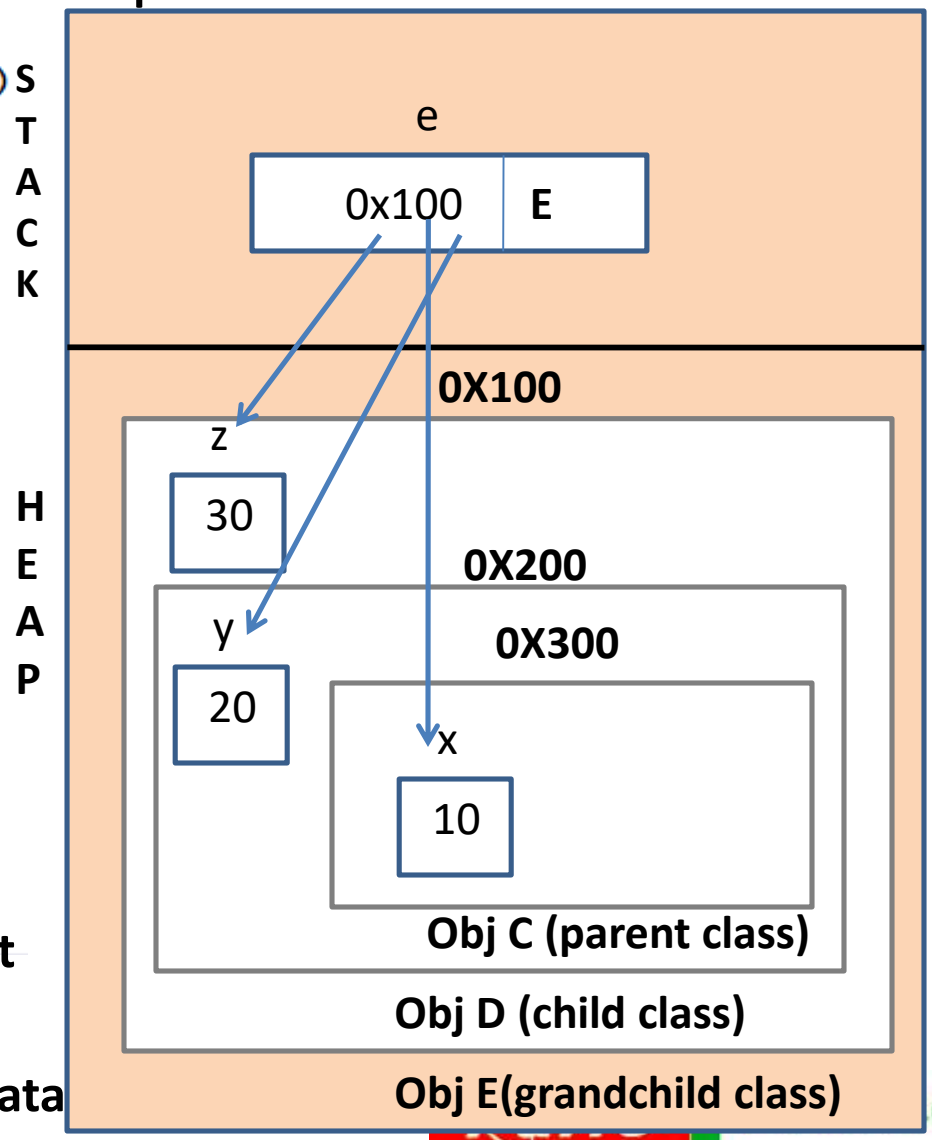
Parent class data will be loaded first

using child class object ref we can access child and parent class data

Inheritance Assignment 1

```
class Program
{
    static void Main(string[] args)
    {
        E e = new E();
        Console.WriteLine(e.x);
        Console.WriteLine(e.y);
        Console.WriteLine(e.z);
    }
}
public class C
{
    public int x = 10;
}
public class D:C
{
    public int y = 20;
}
public class E : D
{
    public int z = 30;
}
```

Req: draw Ram architecture



Parent class data will be loaded first

Note:

e can access parent and grand parent data

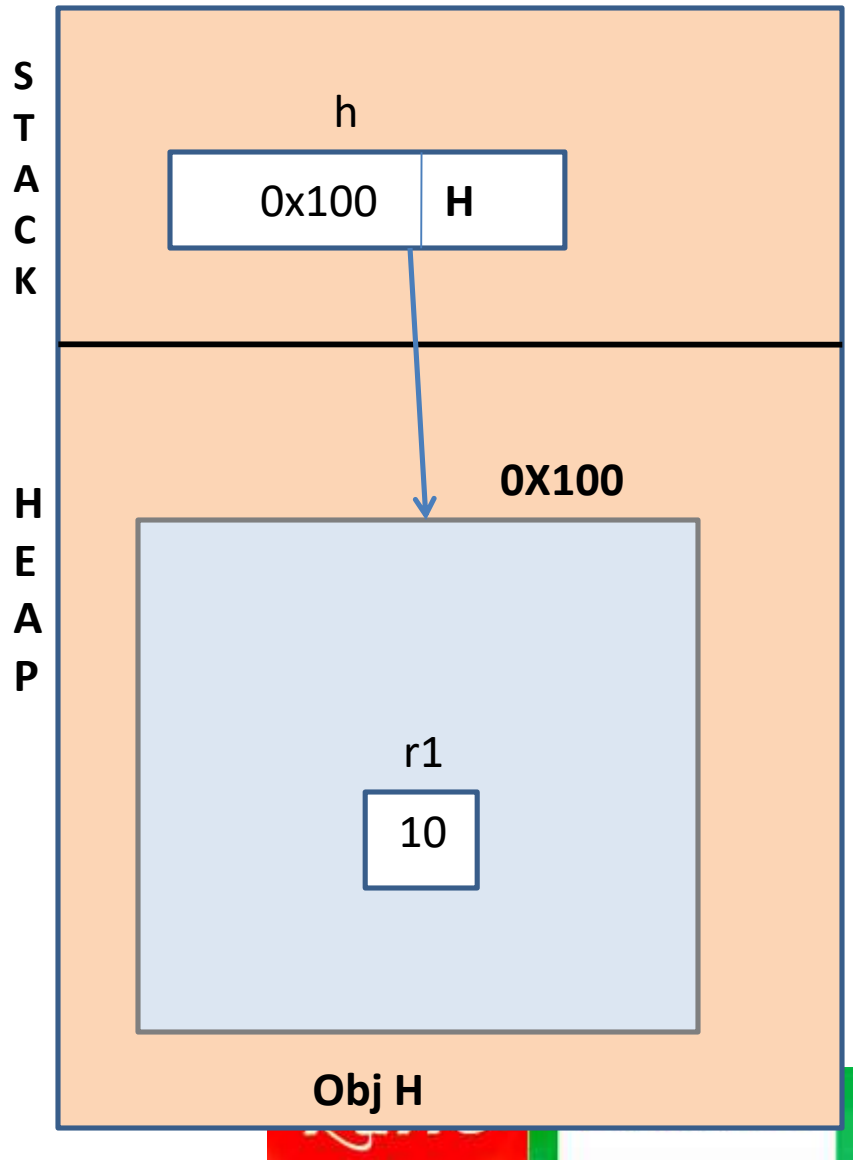
Inheritance Assignment 2

```
class Program
{
    static void Main(string[] args)
    {
        G g = new G();
    }
}
public class F
{
    public int r1 = 10;
}
public class G:F
{
    public int r2 = 20;
}
```

Req: write code for printing data
And show Ram architecture

Inheritance Assignment 3

```
public class H
{
    public int r1 = 10;
}
public class K : H
{
    public int r2 = 20;
}

static void Main(string[] args)
{
    H h = new H();
}
```



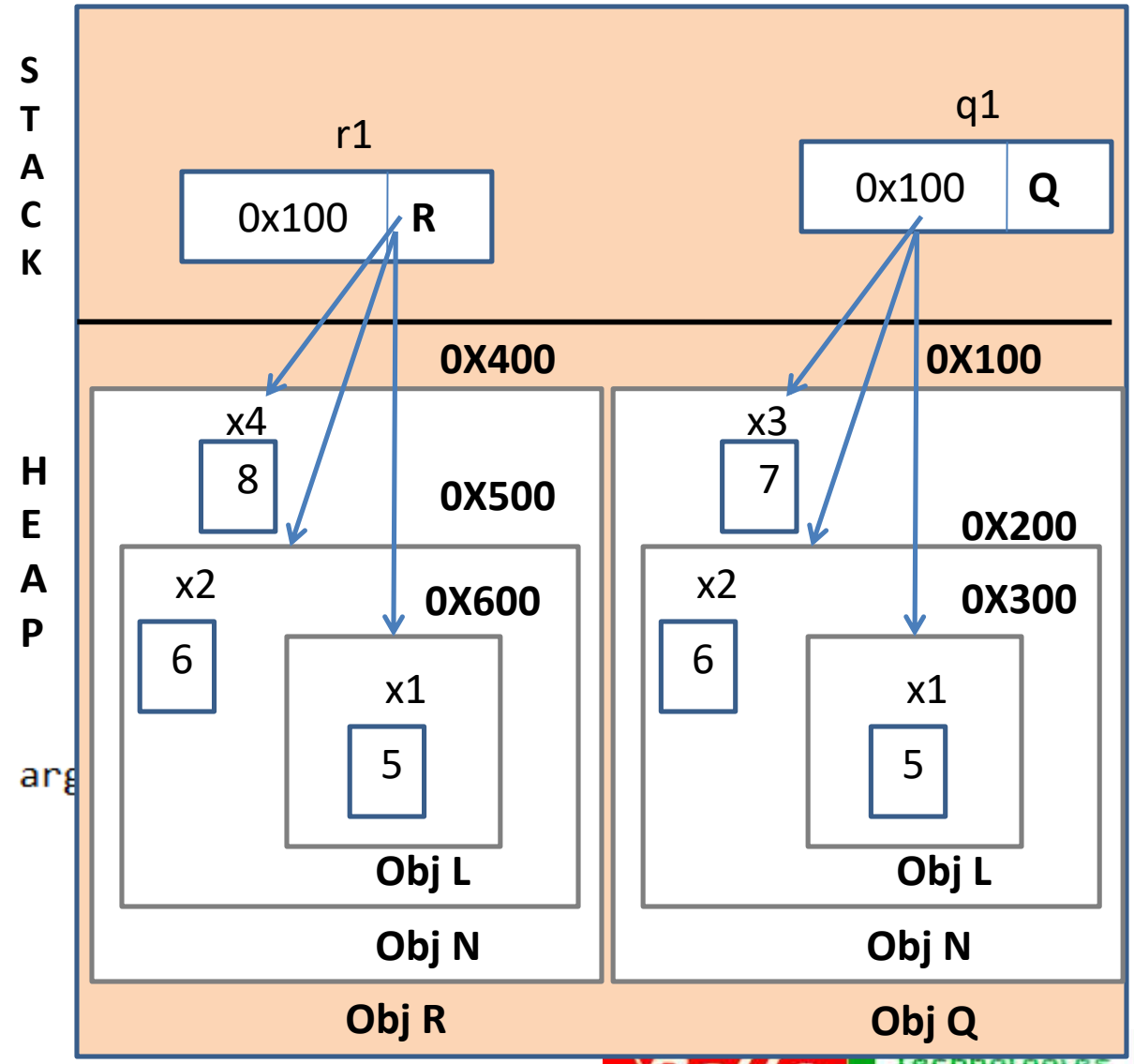
Req: draw Ram architecture

How many objects will be created

Inheritance Assignment 4

```
public class L
{
    public int x1 = 5;
}
public class N : L
{
    public int x2 = 6;
}
public class Q : N
{
    public int x3 = 7;
}
public class R : N
{
    public int x4 = 8;

}
 static void Main(string[] arg
 {
    Q q1 = new Q();
    R r1 = new R();

 }
```

Req: draw Ram architecture



Object class

as per c# language rules every class must contain a parent class.

```
public class A
{
    public int x = 10;
}
```

→ C# compiler →

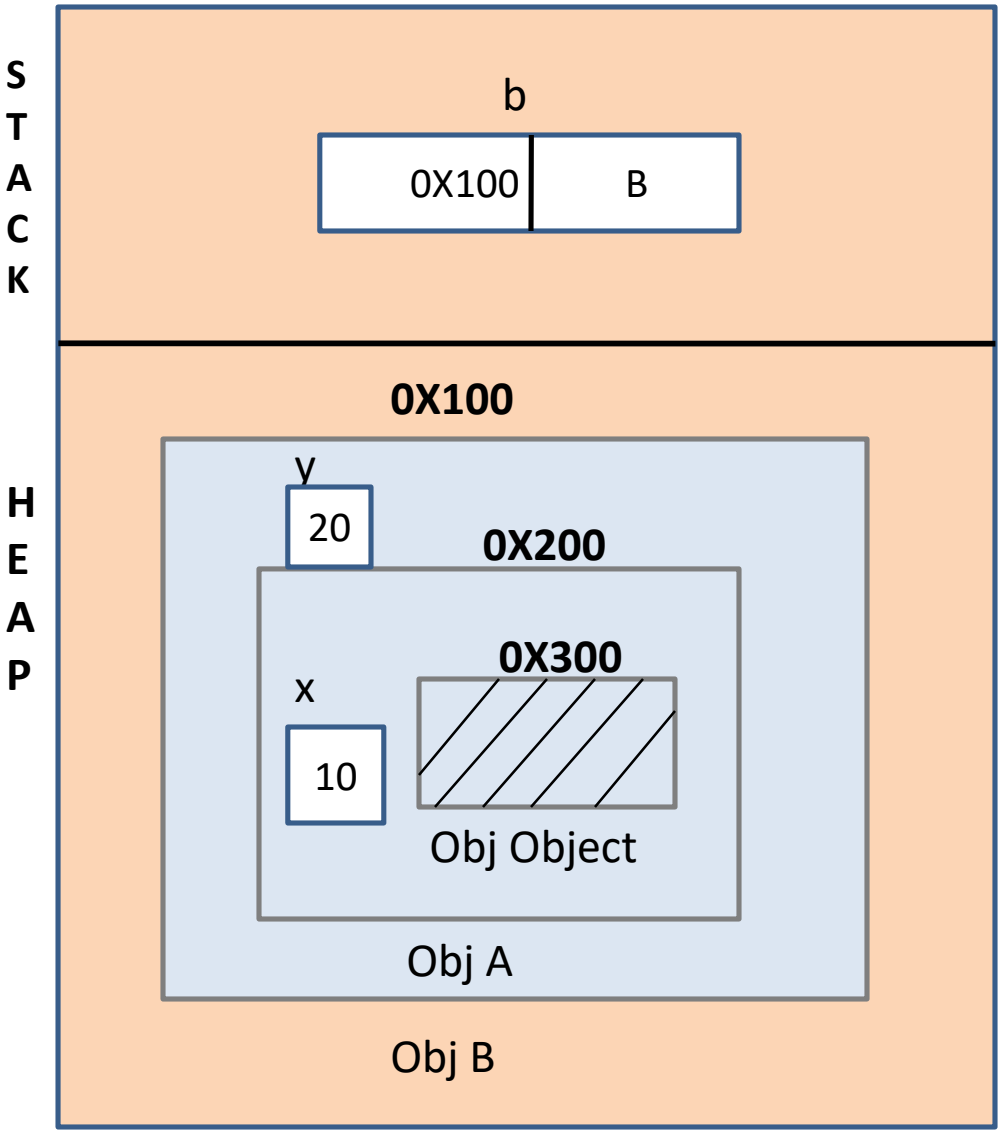
```
public class A :Object
{
    public int x = 10;
}
```

Is it possible to create the object for object class??

```
object o=new object();
```

During compilation time when a **class is not having parent** automatically **C# compiler will add Object class as Parent**.

Object class part-1




```
static void Main(string[] args)
{
    B b=new B();
}
```

CLR

```
public class A :Object
{
    public int x = 10;
}
```

```
public class B : A
{
    public int y = 20;
}
```

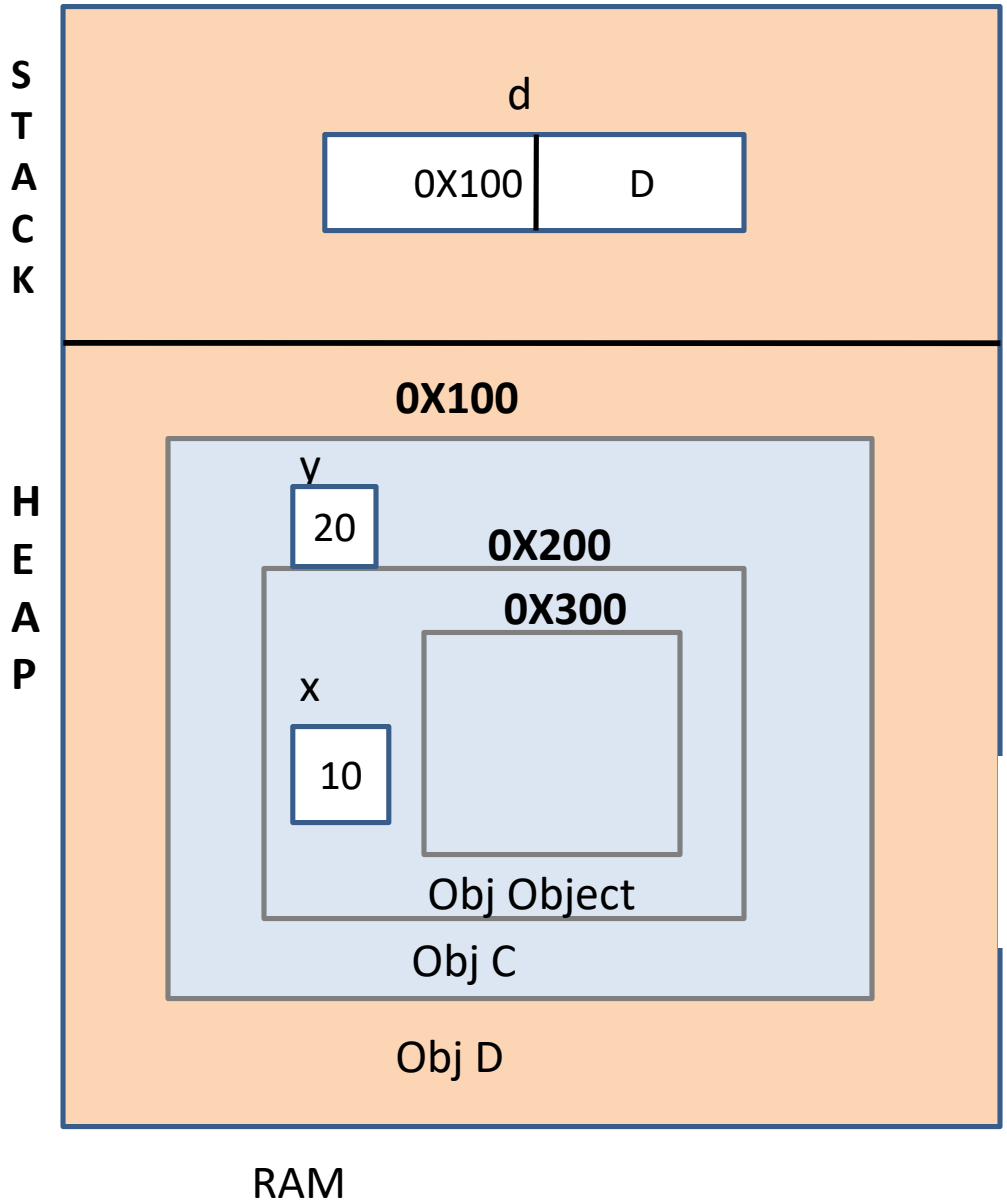
Object class part-2

```
public class C :Object
{
    public int x = 10;
    public int m1()
    {
        return 100;
    }
}
public class D : C
{
    public int y = 20;
    public int m2()
    {
        return 200;
    }
}
static void Main(string[] args)
{    D d = new D();
     d.
}
```

C# compiler

- Equals
- GetHashCode
- GetType
- m1
- m2
- ToString
- x
- y

object class part- 3



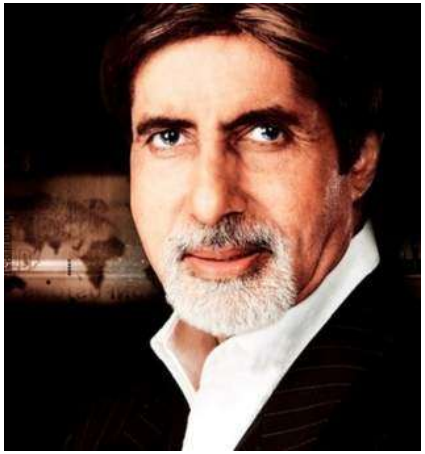

```
public class C
{
    public int x = 10;
    public int m1()
    {
        return 100;
    }
}
public class D : C
{
    public int y = 20;
    public int m2()
    {
        return 200;
    }
}
static void Main(string[] args)
{
    D d = new D();
}
```

CLR

Upcasting

Identifying parent class object reference by using child class object reference

Abhishek bhachan (child)



Amithab bhachan(parent)

Harivansha rai bhachan (grandparent)

Father of Abhishek bhachan???????

Grand Father of Abhishek bhachan???????

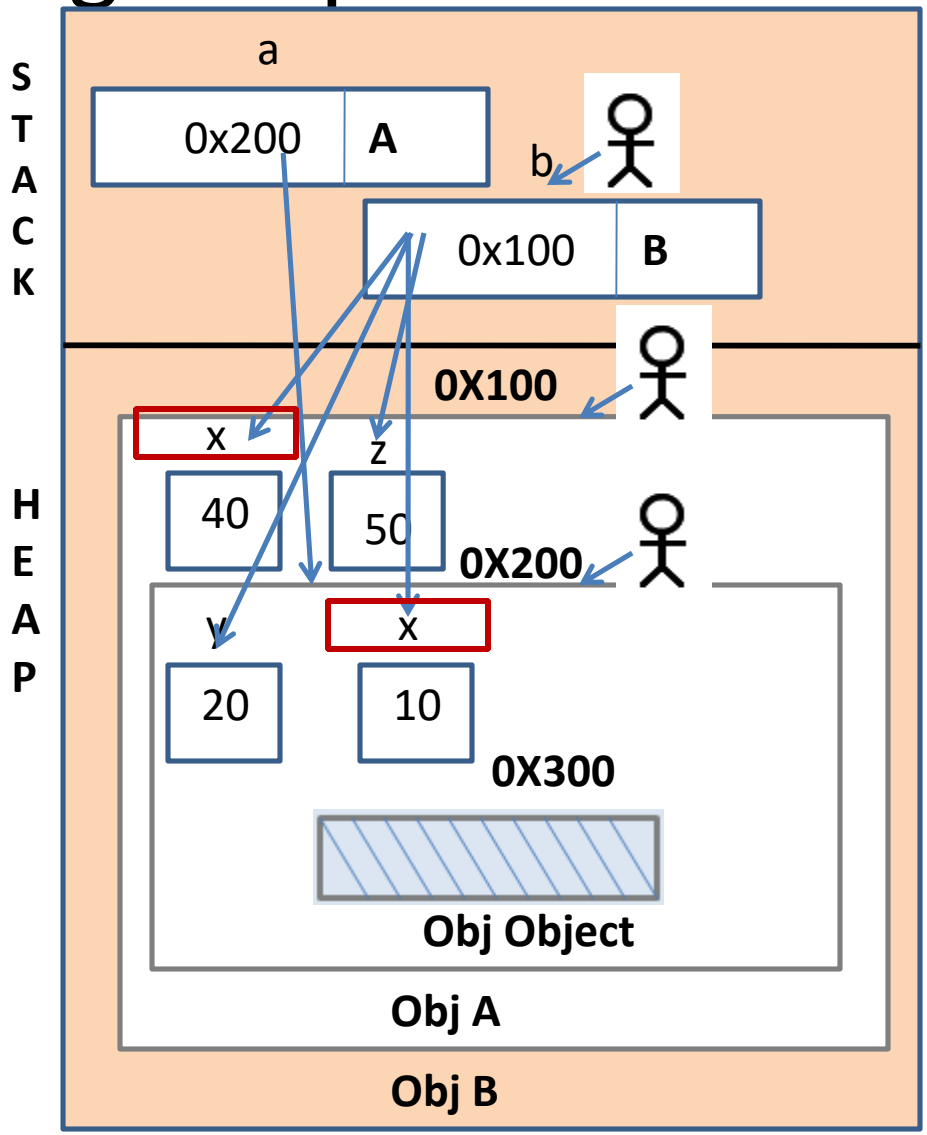
Father of Amithab bhachan???????

Using child class object reference we can identify direct or indirect parent class object reference

Upcasting sample

```
static void Main(string[] args)
{
    B b = new B();
    Console.WriteLine(b.x);//40
    Console.WriteLine(b.z);//50
    Console.WriteLine(b.y);//20
    A a = b;
    Console.WriteLine(a.x);//10
}
```

```
public class A
{
    public int x = 10;
    public int y = 20;
}
public class B : A
{
    public int x = 40;
    public int z = 50;
}
```



We are asking clr to find parent class object using child class object

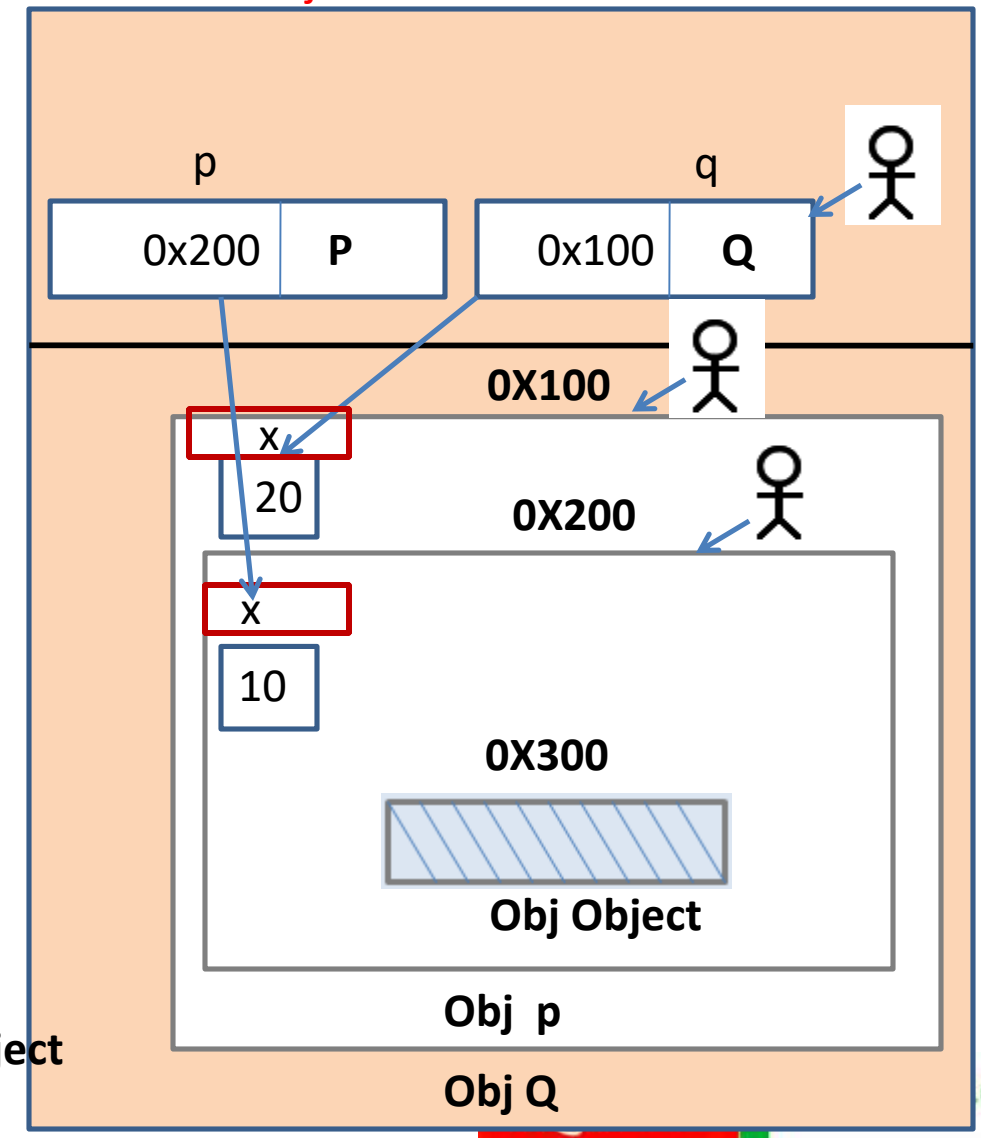
upcasting assignment

Req:1draw the ram architecture
2.print all the variable data present in all the created objects.

```
static void Main(string[] args)
{
    Q q = new Q();

    Console.WriteLine(q.x);//20

    P p = q;

    Console.WriteLine(p.x);//10
    Console.ReadLine();
}

public class P
{
    public int x = 10;

}
public class Q : P
{
    public int x = 20;
}
public class R : Q
{
    public int x = 30;
}
```

If child class and parent class Variable names are same use Upcasting

We are asking clr to find parent class object using child class object



downcasting

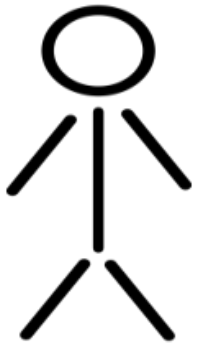
Identifying child class object reference using parent class object reference called as down casting

Abhishek bachan

Amitabh bachan

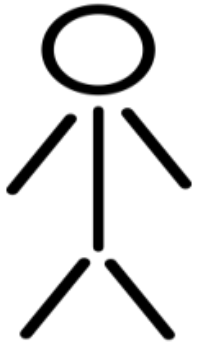
Who is Amitabh's child??

downcasting pre requisite



Child class

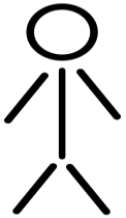
ccv = new ( );



Child class

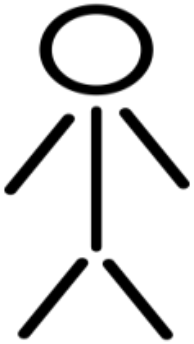
```
public class R
{

}
public class L:R
{

}
```

```
L l = new L();
R r = new L();
Object o = new L();
```



Parent class

pcv = new ( );

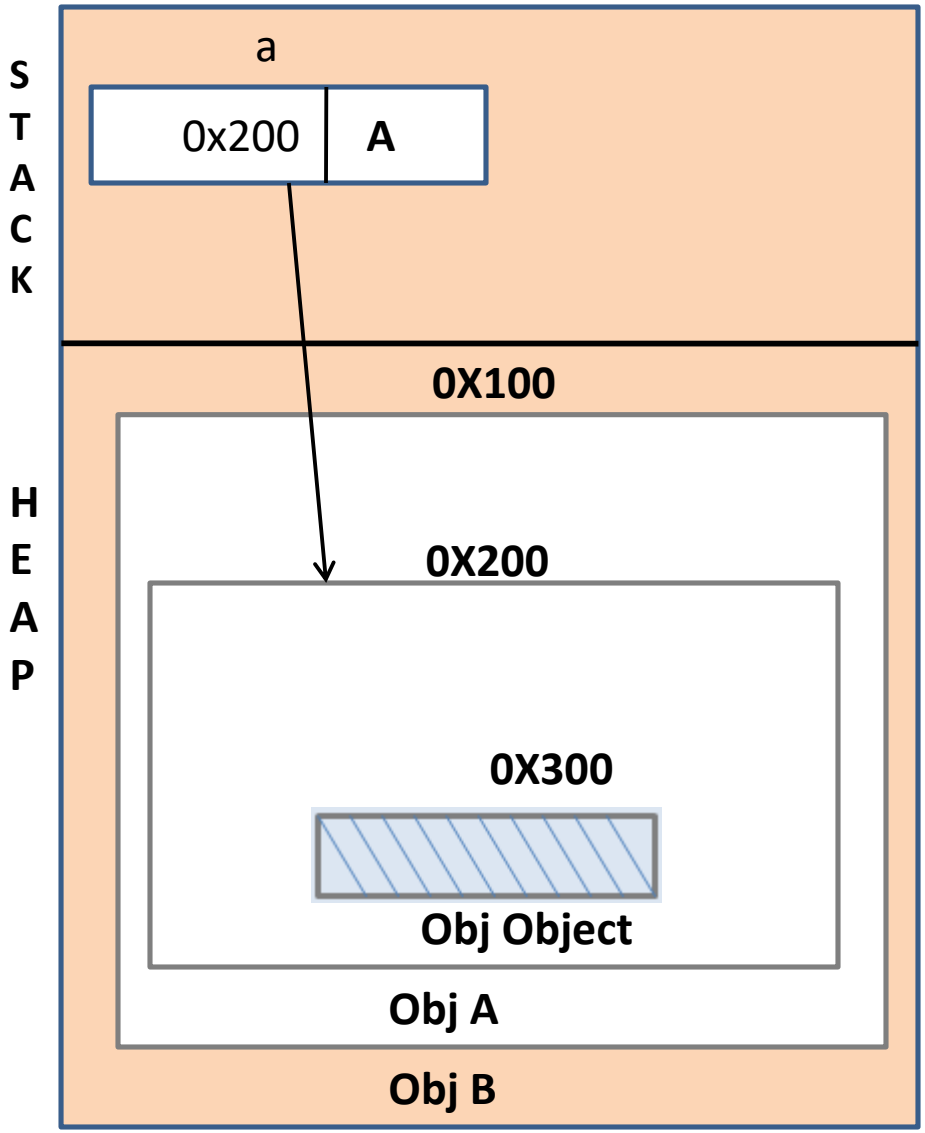


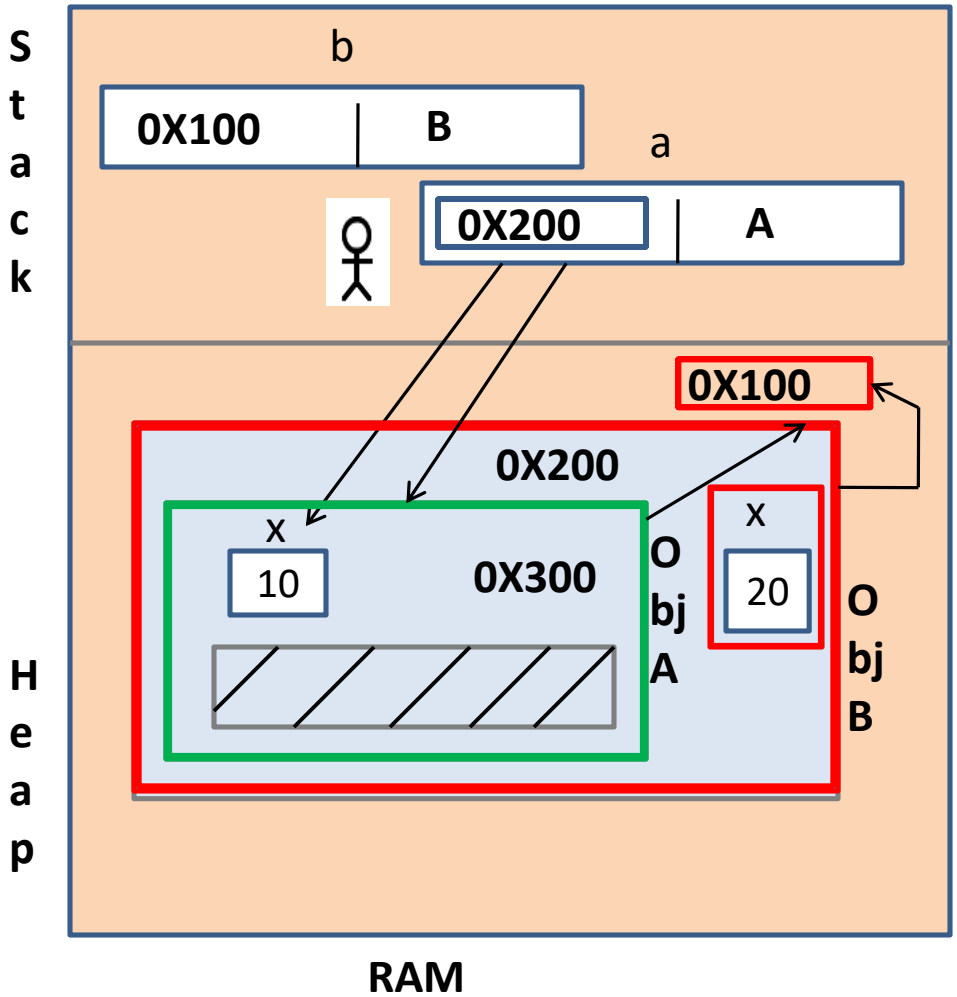
Child class

NOTE:

Since every child will be having a Parent

```
public class A
{

}
public class B:A
{

}
static void Main(string[] args)
{
 A a = new B();
}
```






```
static void Main(string[] args)
{
        A a = new B();   // 0X100

        B b = (B)a;

}
public class A
{
    public int x = 10;
}
public class B:A
{
    public int x = 20;
}
```

PCO

CCO

```
B b = (B)a;
```

CLR

Child class

Child class

base keyword

- base keyword is used for accessing base class members from derived class

```
P   C   A
{

}
```

```
P   C   B:A
{
    P   V   M1()
    {

    }
}
```

base Keyword

```
public class A
{
    public int x = 10;
}
public class B : A
{
    public int y = 20;
    public void m1()
    {
        Console.WriteLine(y);
        Console.WriteLine(base.x);
    }

}
```

base keyword assignment

```
public class D
{
    public void m2()
    {
        Console.WriteLine("Hai");
    }
}
```

```
public class E : D
{
    public void m3()
    {
        Console.WriteLine("Hello");
    }
    public void m4()
    {
        m3();
        base.m2();
    }
}
```

constructor execution sequence

- as per c# language rules

parent class constructor must be executed first and later derived class constructor

constructor execution sequence

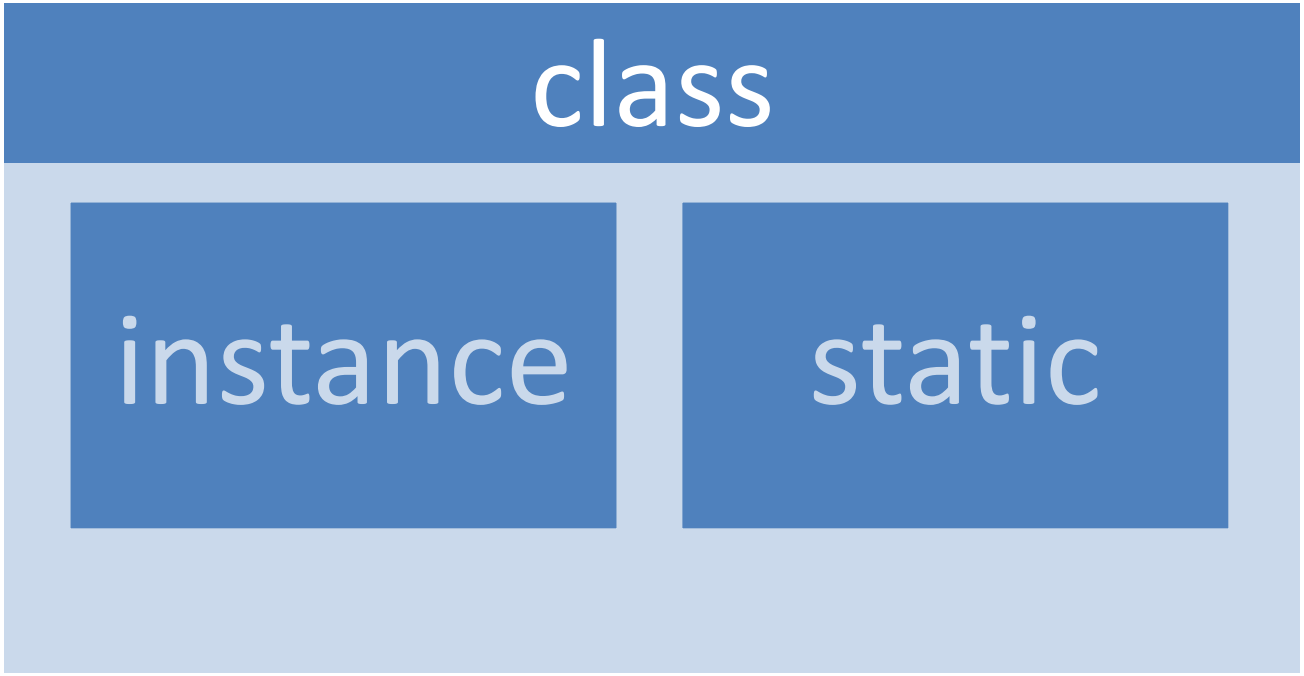
```
class Program
{
    static void Main(string[] args)
    {
        B b = new B();
    }
}
public class A
{
    public A()
    {
        Console.WriteLine("hello");
    }
}
public class B : A
{
    public B()
    {
        Console.WriteLine("hi");
    }
}
```

hello

hi

instance & static members

- usually class can contain 2 types of members



```
public class A
{
    public int x = 10;
    public static int y = 20;
    public int M1(int r)
    {
        return 17;
    }
    public static int M2(int i)
    {
        return i + 6;
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        A a = new A();
        Console.WriteLine(a.x);
        int r1=a.M1(5);
        Console.WriteLine(A.y);
        int r2=A.M2(3);
    }
}
```

instance & static member rules

same class instance method can access
same class instance & static member

same class static method can access
same class static members but not instance

```
public class A
{
    public int x = 10;
    public static int y = 20;
    public void M1()
    {
        Console.WriteLine(x);
        Console.WriteLine(A.y);
    }
    public static void M2()
    {
        Console.WriteLine(A.y);
        Console.WriteLine(x);
    }
}
```

this Keyword

1.In c# language **this** Keyword refers to current instance /object

2. usage of **this** keyword is mandatory for refering **instance variables** when the **instance variables** names are same as local variable

3.during program execution time **this keyword** will be **replaced** with **current object address**

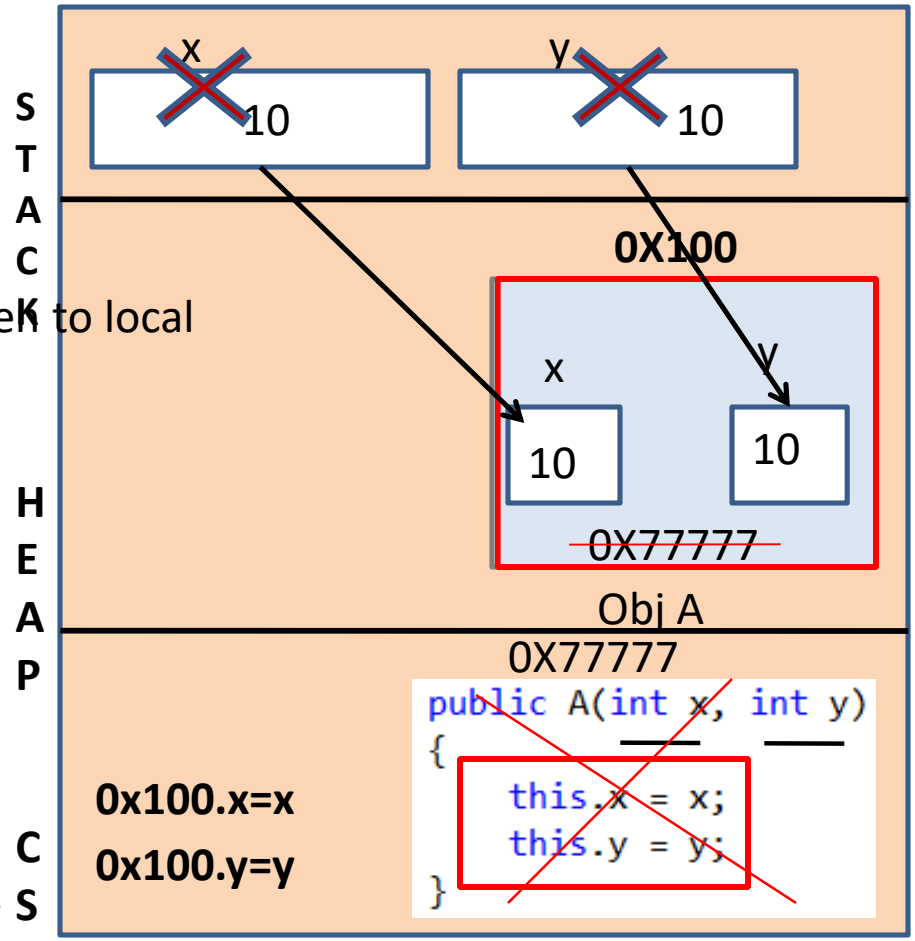
4. using this key word we can access only same class instance memebers.

this keyword-part2

Req:Using constructor assign values instance variables

```
public class A
{
 public int x;
 public int y;
 public A(int x, int y)
 {
  this.x = x;
  this.y = y;
 }

 }

static void Main(string[] args)
{
 A a1 = new A(10,10);

}
```

always priority will be given to local variables



Now CLR will search in object for x and y

So we will not get any effect

Whatever the data present in x and y will be copied into x,y in object

What is the meaning of **this keyword**?

Current object/Current Instance

Which is the current object?

How to point Instance variable?

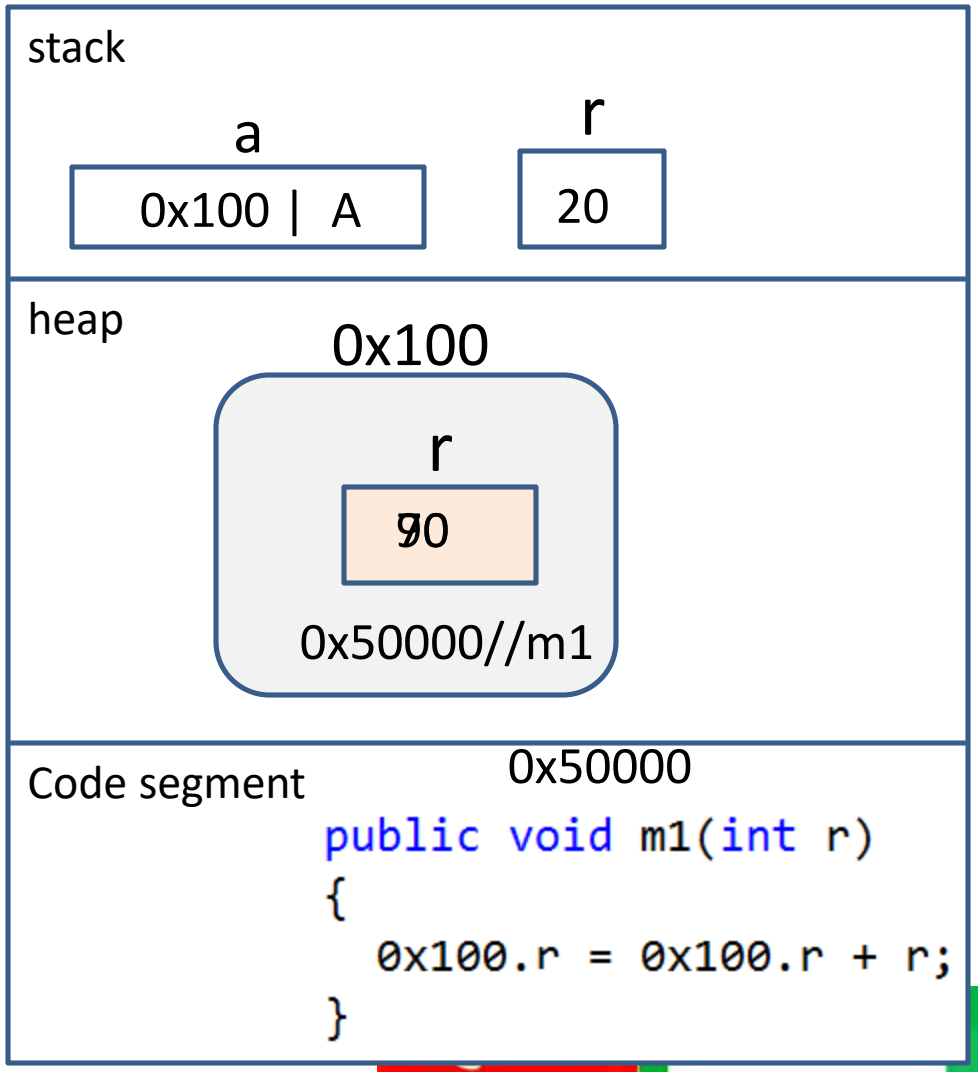
this keyword assignment.

```
public class A
{
    public int r = 70;
    public void m1(int r)
    {
        this.r = this.r + r;
    }
}

class Program
{
    static void Main(string[] args)
    {
        A a = new A();
        a.m1(20);
        Console.WriteLine(a.r);
        Console.ReadLine();
    }
}
```

// 90

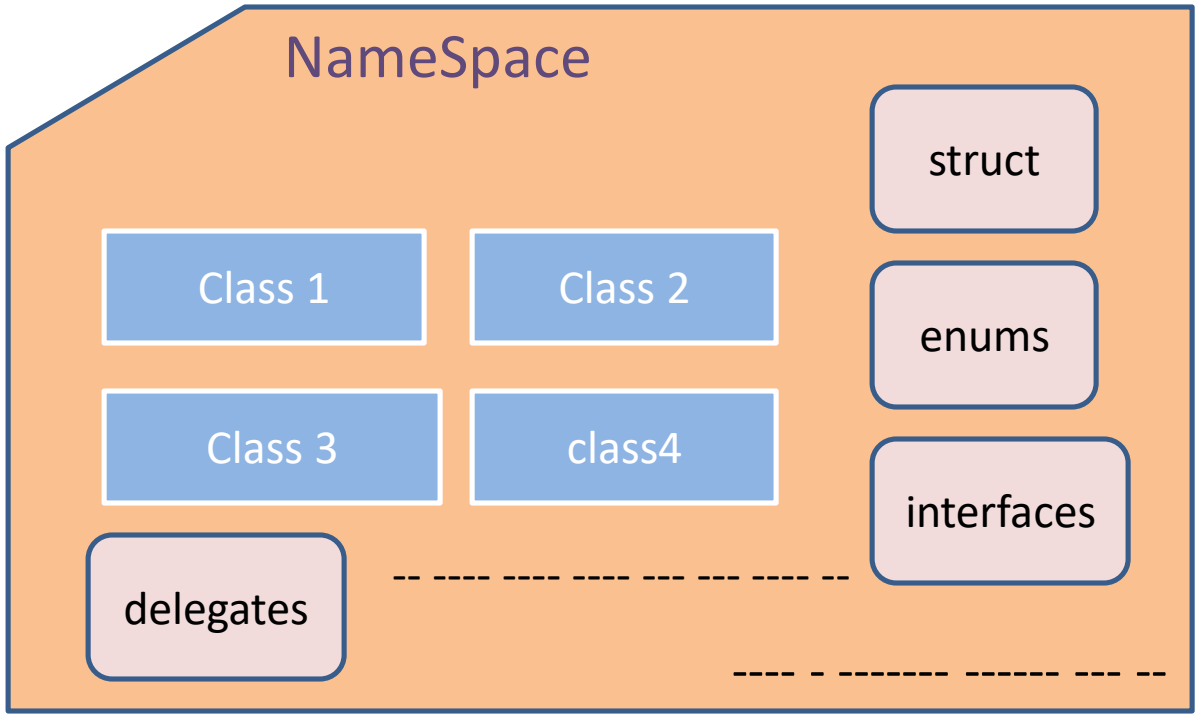
What is the program output?

We will see the execution sequence



namespaces

namespace is a logical container for grouping logically related classes,structs,enums,interfaces,delegates etc



namespace internals

```
namespace N1
{
    public class A
    {

    }
}
```


```
namespace N1
{
    public class N1.A
    {

    }
}
```

using one namespace member in the other

```
namespace N1
{
    public class A
    {

    }
    public class B
    {
        public void m1()
        {
            A a = new A();
        }
    }
}
```

```
using N1;
namespace N2
{

    public class C
    {
        public void m2()
        {
            A a = new A();
        }
    }
}
```

for resolving this error add namespace to the header using the using statement

compilation error: since class A is not present in N2 namespace

Do you think this code work

Yes as class A and class B are in same namespace

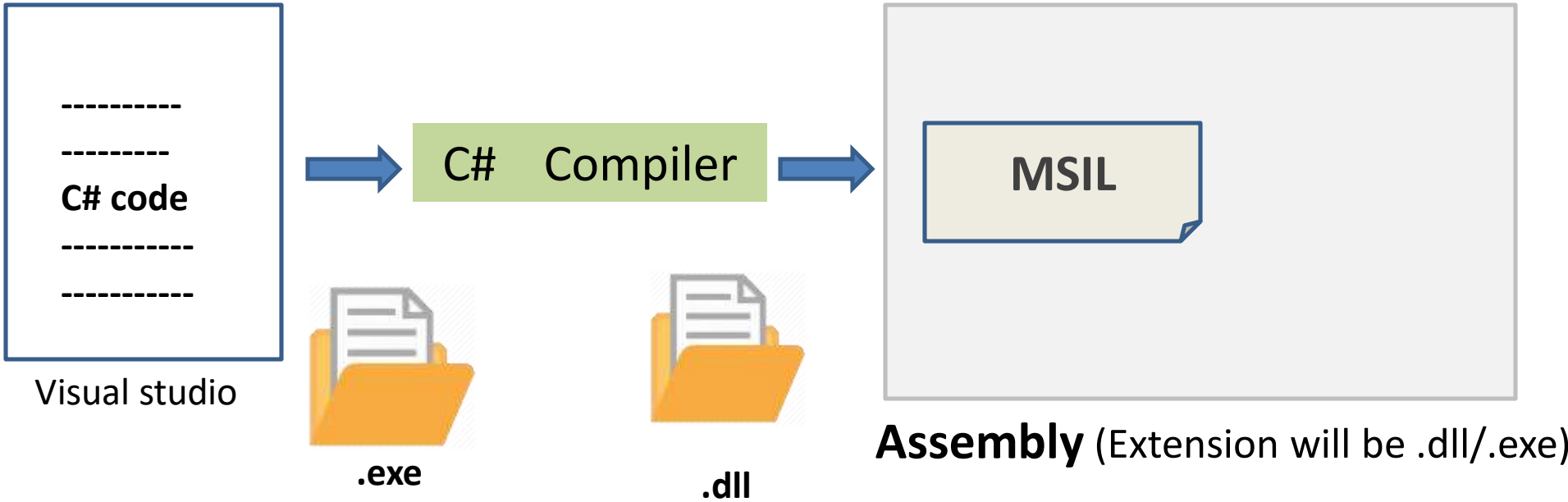
naming ambiguity

```
using N100;
using N200;

namespace N300
{   class Program
    {
      static void Main(string[] args)
      {
        N100.A a = new N100.A();
        B b = new B();
      }
    }
}
```

I would like to create objects for A class prsent in N100 & B class Present in N200

```
namespace N100
{
  public class A
  {

  }
}

namespace N200
{
  public class B
  {

  }
  public class A
  {

  }
}
```

assemblies

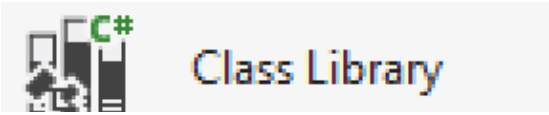


D:\xyz\project1\bin\debug

Console Application

```
static void Main(string[] args)
{


}
```

D:\xyz\project1\bin\Debug



Class Library

```
static void Main(string[] args)
{


}
```

using one project code in the other

D:\MyClassLibrary

```
namespace MyClassLibrary
{
    public class A
    {
        public int m1(int x, int y)
        {
            return 10;
        }

    }
}
```

Class Library

C# Compiler

MyClassLibrary.dll

D:\MyClassLibrary\bin\Debug\MyClassLibrary.dll

```
namespace MyConsoleApplication
{
  class Program
  {
      static void Main(string[] args)
      {


      }
  }
}
```

Console Application

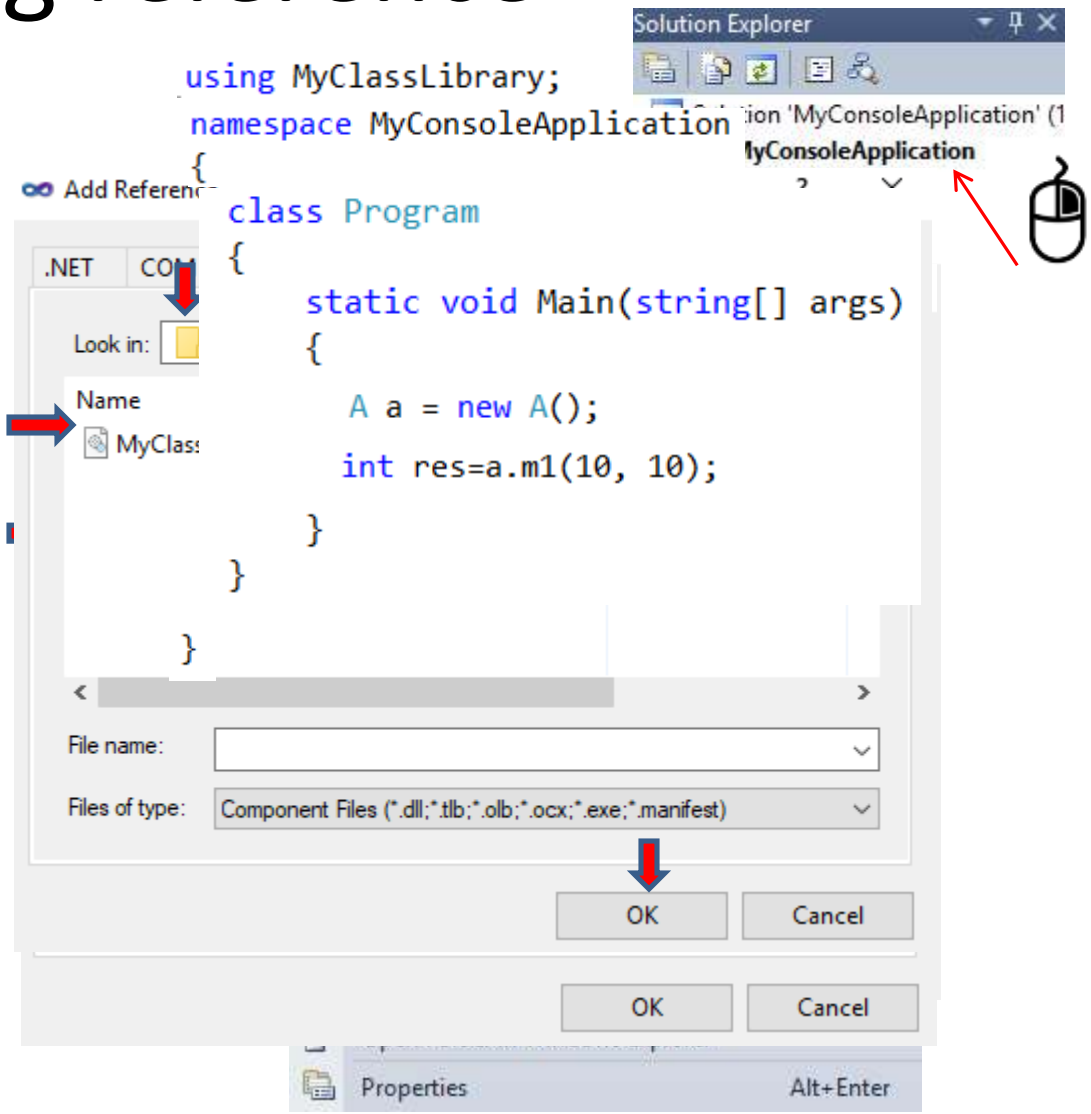
By Adding Reference

Req:Call m1() method in main()

adding reference

```
namespace MyClassLibrary
{
 public class A
 {
     public int m1(int x, int y)
     {
         return 10;
     }

 }
}
```

```
using MyClassLibrary;
namespace MyConsoleApplication
{
 class Program
 {
     static void Main(string[] args)
     {
       A a = new A();
       int res=a.m1(10, 10);
     }
 }

}
```



access modifiers.

Access modifiers are used to provide security for the program members

Program members can be classes , methods, constructors, fields ,etc,.

Types of Access Modifiers.

public → Is accessible anywhere , no security.

private → Is accessible only within the declared context.

protected → Is accessible in declared context and within the derived context.

internal → Is accessible only within the current Assembly.

protected internal

→These members are accessible within same assembly or in the Declared class also from Derived class

Access Modifiers Part1

```
namespace N100
{
    public class A
    {
        public int i = 1;
        private int j = 2;
        protected int k = 3;
        internal int l = 4;
        protected internal int n = 5;
        public void M1()
        {
            C.WL(i);    C.WL(j);    C.WL(k);

            C.WL(l);    C.WL(n);

        }
    }
}
```

```
public class B
{
    public void M2()
    {
        A a = new A();
        C.WL(a.i);    C.WL(a.j);

        C.WL(a.k);    C.WL(a.n);

        C.WL(a.l);
    }
}
```

Access Modifiers Part2

```
namespace N100
{
    public class A
    {
        public int i = 1;
        private int j = 2;
        protected int k = 3;
        internal int l = 4;
        protected internal int n = 5;
        public void M1()
        {
            C.WL(i);    C.WL(j);    C.WL(k);

            C.WL(l);    C.WL(n);

        }
    }
}
```

```
using N100;
namespace N200
{
    public class D
    {
        public void M3()
        {
            A a = new A();
            C.WL(a.i);    C.WL(a.j);

            C.WL(a.k);    C.WL(a.n);

            C.WL(a.l);
        }
    }
    public class E : A
    {
        public void M4()
        {
            C.WL(i);    C.WL(j);    C.WL(k);

            C.WL(l);    C.WL(n);
        }
    }
}
```

Project1

```
namespace N100
{
    public class A
    {
        public int i = 1;
        private int j = 2;
        protected int k = 3;
        internal int l = 4;
        protected internal int n = 5;
        public void M1()
        {
            C.WL(i);    C.WL(j);    C.WL(k);

            C.WL(l);    C.WL(n);

        }
    }
}
```

Project2

```
using N100;
namespace N300
{
    public class F
    {
        public void M5()
        {
            A a = new A();
            C.WL(a.i);    C.WL(a.j);
            C.WL(a.l);
            C.WL(a.k);    C.WL(a.n);
        }
    }
    public class G : A
    {
        public void M6()
        {
            C.WL(i);    C.WL(j);    C.WL(k);

                        C.WL(l);    C.WL(n);
        }
    }
}
```

allowed access modifiers in namespaces

```
namespace N2
{
  public class A
  {
    ------------
  } ------------
  internal class B
  {
      ------------
      ------------
  }
  private class D
  {
    ------------
  } ------------
}
```

```
namespace N2
{
    public
    internal
    protected
    private
    Protected internal
}
```

default access modifiers

```
namespace N2
{
    class A
    {
        ------------
        ------------
    }
}
public class E
{
    int r = 20;
}
```

C# compiler

C# compiler

```
namespace N2
{
    internal class A
    {
        ------------
        ------------
    }
}
public class E
{
    private int r = 20;
}
static void Main(string[] args)
{
    E e = new E();
    e.
}
```

- Equals
- GetHashCode
- GetType
- ToString

Variable 'r' not avaialable

polymorphism

- When an entity is appearing with the same name in different forms then that entity is said to exhibiting polymorphism

Types of polymorphism:

➢ Compile time polymorphism/Static/Early binding

overloading

Method Overloading | **Constructor Overloading** | **Operator Overloading**

➢ Runtime polymorphism/Dynamic/Late binding

overriding

↓

method overriding

Method Overloading

When a method is appearing with the same name and with the different signatures in a given context then the method is said to be overloaded

```
public class A
{
    public int m1()
    {
        return 10;
    }
    public int m1(int x)
    {
        return 100;
    }
    public int m1(string y)
    {
        return 100;
    }
    public int m1(int z,int r)
    {
        return 1000;
    }
}
```

Same method name with different parameters i.e. method m1 is overloaded

Different Parameters :

1)Different types of parameters

2)Different number of parameters

3)Different order of parameters

Method overloading using multiple classes

- We can overload a method using multiple classes which are participating in inheritance

```
public class B
{
    public int m2(int x,int y)
    {
        return 2 * (x - y); //20
    }
}
public class C:B
{
    public int m2(int x, int y,int z)
    {
        return 2 * (x + y + z);//120
    }
}
```

```
static void Main(string[] args)
{
    C c = new C();
    int r1 = c.m2(20, 10);
    int r2 = c.m2(30, 20, 10);
}
```

How many methods are present in class C

Constructor overloading

```
static void Main(string[] args)
{
  A a = new A();
  A a2 ▲ 2 of 2 ▼ A.A(int x1)

}
public class A
{
  public A()
  {

  }
  public A(int x1)
  {

  }
}
```

Same Constructor name with different signatures.

Constructor overloading

method overriding

Before understanding overriding we have to understand virtual methods

virtual methods

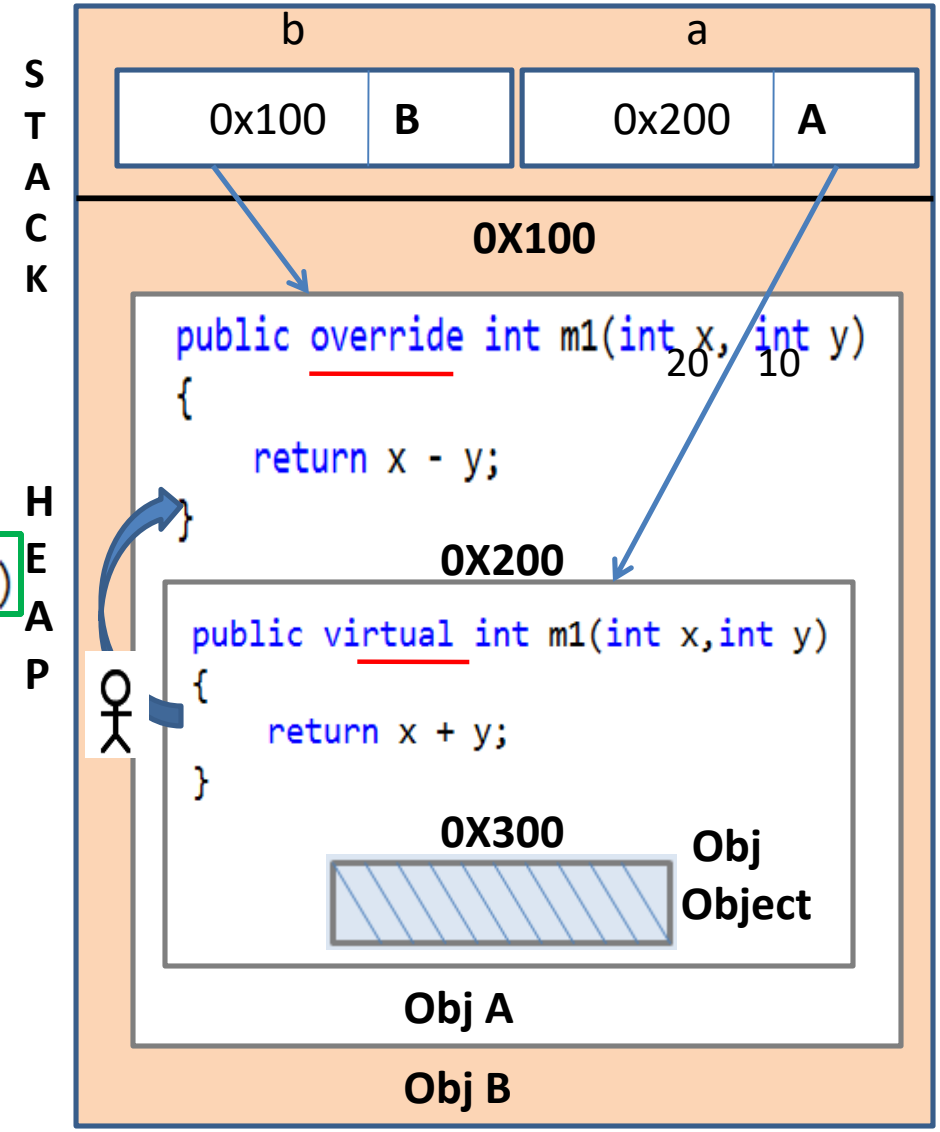
- virtual methods are considered as low priority methods
- We can change the behavior of virtual methods in derived class

override methods

- overriding methods are considered as high priority methods
- We can change the behavior of virtual methods using override methods
- We can override the override methods

overriding sample

```
public class A
{
    public virtual int m1(int x,int y)
    {
        return x + y;
    }
}
public class B:A
{
    public override int m1(int x, int y)
    {
        return x - y;
    }
}

static void Main(string[] args)
{
    B b = new B();
    int r1 = b.m1(20, 10);  //10
    A a = b;  //upcasting
    int r2 = a.m1(20, 10);  //10
}
```



method 1 in class A

Whenever an object serves given keyword it checks for overrided method

Always priority will be given to behavior of virtual method m1 using overriding

overriding an overrided method

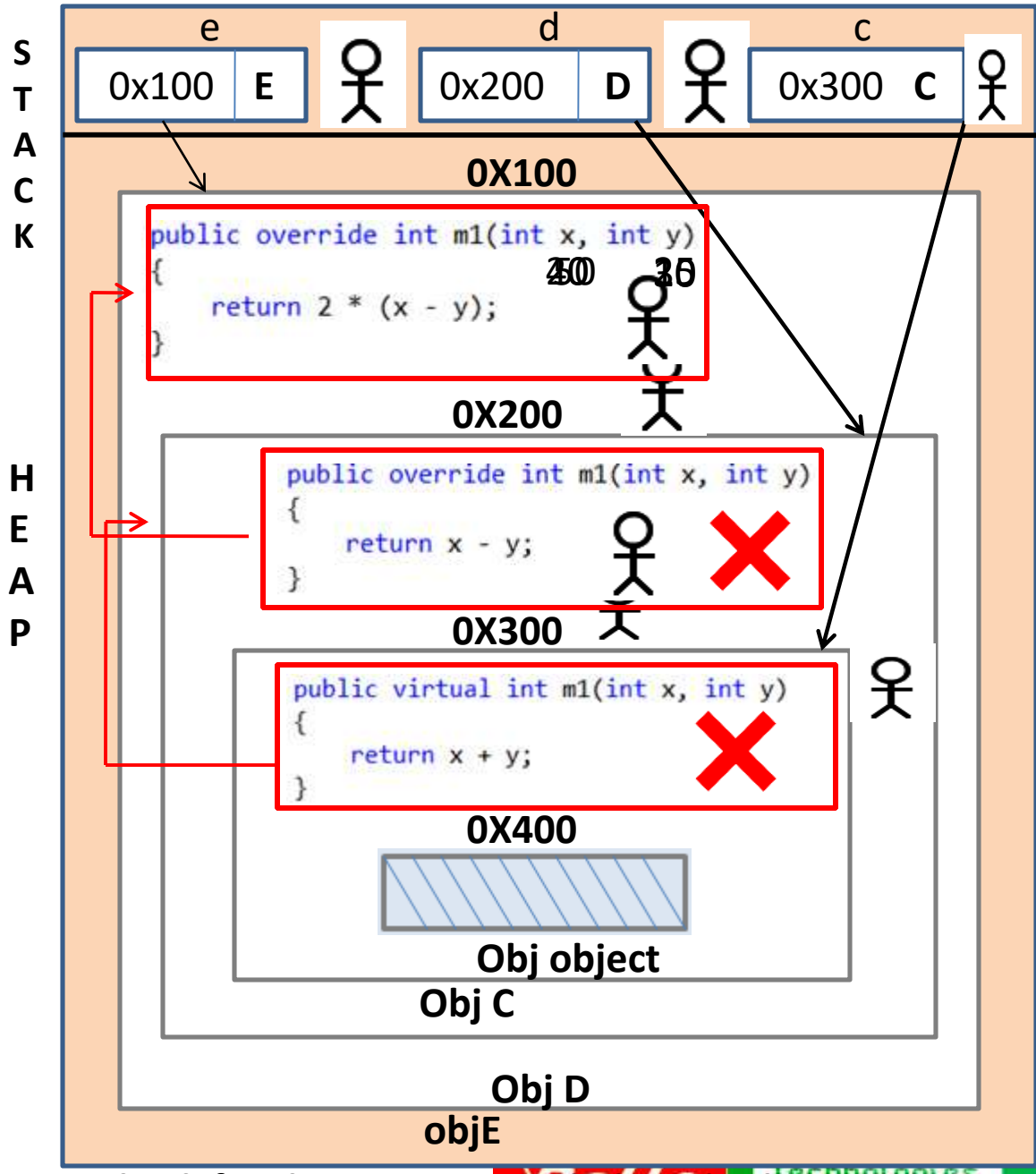
We are allowed to override already overrided method

```
public class C
{
    public virtual int m1(int x, int y)
    {
        return x + y;
    }

}
public class D : C
{
 public override int m1(int x, int y)
 {
     return x - y;
 }
}
public class E : D
{
 public override int m1(int x, int y)
 {
     return 2 * (x - y);
 }
}
```

```
static void Main(string[] args)
{
   E e = new E();
   D d = e;
   C c = d;
  int r1 = e.m1(20,10);
  int r2 = d.m1(40, 30);
  int r3 = c.m1(50, 25);
}
```

We Will see the RAM architecture

```
public class C
{
    public virtual int m1(int x, int y)
    {
        return x + y;
    }

}
public class D : C
{
 public override int m1(int x, int y)
 {
     return x - y;
 }
}
 public class E : D
 {
 public override int m1(int x, int y)
 {
     return 2 * (x - y);
 }
}
 static void Main(string[] args)
 {
    E e = new E();
    D d = e;
    C c = d;
    int r1 = e.m1(20,10);  //20
    int r2 = d.m1(40, 30);  //20
    int r3 = c.m1(50, 25);  //50
 }
```



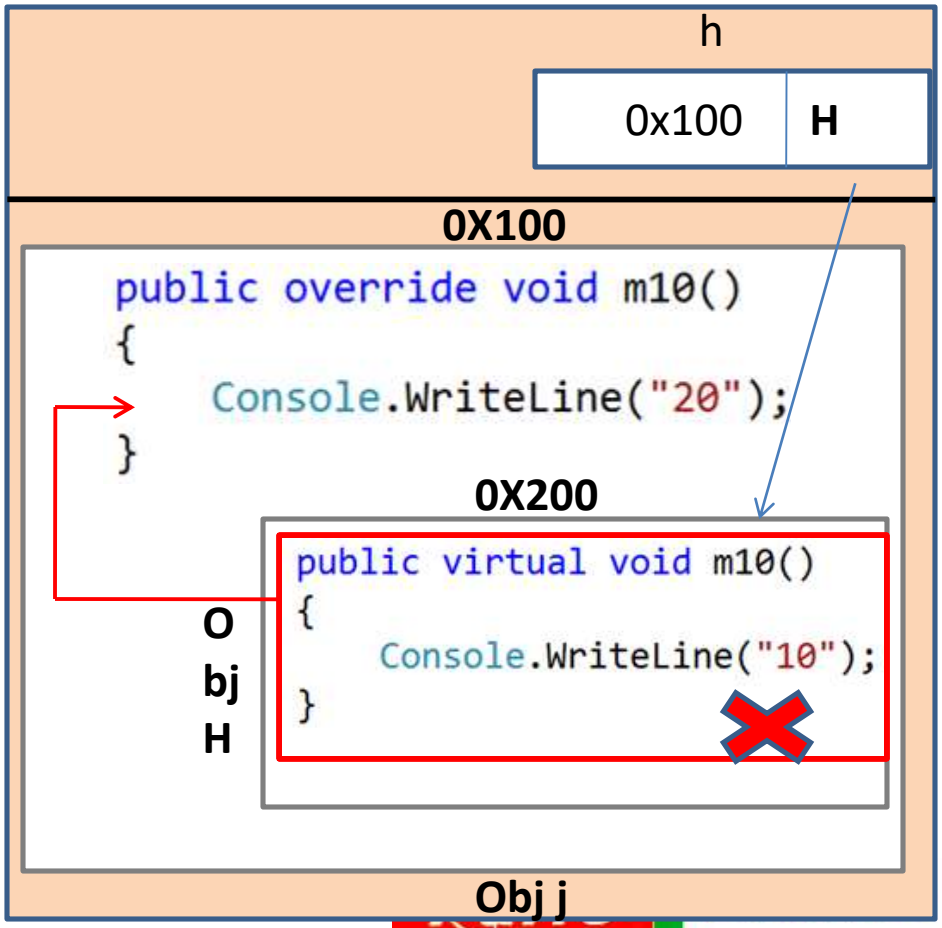
CLR will always check for the super overrided method

override assignment

```
public class H
{
    public virtual void m10()
    {
        Console.WriteLine("10");
    }
}
public class J : H
{
    public override void m10()
    {
        Console.WriteLine("20");
    }
}
public class K : J
{
    public override void m10()
    {
        Console.WriteLine("70");
    }
}
```

```
static void Main(string[] args)
{
    H h = new J();
    h.m10();
}
```

what is the output?



sealed methods

by using sealed methods we can stop behavior overriding or behavior changing

```
public class A
{
    public virtual int m1(int x,int y)
    {
        return x + y;
    }
}
public class B:A
{
    public override int m1(int x, int y)
    {
        return x - y;
    }
}
public class C:B
{
    public sealed override int m1(int x, int y)
    {
        return 2*(x + y);
    }
}
```

```
public class D : C
{
    public  override int m1(int x, int y)
    {
        return 3 * (x + y);
    }
}
```

Compilation error

If we are declaring any method as sealed method ,we cannot ove[illegible] method

static constructor

- Static constructors are used for initializing static variable
- Static constructors must not contain parameters and access modifiers
- A static constructor is executed only once in the program life time
- We cannot call a static constructor explicitly

static constructor sample

```
public class A
{
    public int x;
    public A(int x)
    {
        this.x = 20;
    }
    public static int y;
    static A()
    {
        y = 20;
    }
}
```

Instance Variable

Instance Constructor

Static Variable

Static Constructor

Note: always static constructor will be executed first and only once

static constructor execution seq

```
public class D
{
  public D()
  {                    3
   Console.WriteLine("one");
  }
  static D()
  {                    2
   Console.WriteLine("Two");
  }
}
public class E:D
{
   public E()
   {                   4
       Console.WriteLine("three");
   }
   static E()
   {                   1
       Console.WriteLine("four");
   }
}
```

```
static void Main(string[] args)
{
    E e = new E();
    Console.ReadLine();
}
```

What is the output??

file:///c:/us

```
four
Two
one
```

```
E e = new E(20);
```

Types of classes

Instance	Sealed
Static	nested
Partial	abstract

instance & sealed class

instance class : a class which allows object creation

sealed class : a class which can't have child classes

we are allowed to created objects for sealed class

```
class Program
{
    static void Main(string[] args)
    {
        A a = new A();
    }
}
public class A
{

}
```

```
class Program
{
    static void Main(string[] args)
    {
        B b = new B();
    }
}
public sealed class B
{

}
```

static class

static class can't participate in inheritance

static class can have only static members

we can't create object of static class

syntax :

```
access_specifier  static class   <classname>
{
    static variable + static method+

    static constructor + static property

}
```

static class

```
public static class A
{

}
public class B : A
{
    Static class cannot act as a base class
}
```

```
static void Main(string[] args)
{
   A a1 = new A();
}
```

Not allowed to create objects in static class

```
public class c
{

}
public static class D : c
{
  Static class cant act as a child class
}
```

nested class

nested class is a class which is placed inside another class

syntax :

```
public class  A
{
    public int x = 10;
    public class B
    {
        public int y = 20;
    }
}
```

Outer class

Inner/nested class

we can declare one namespace with in another namespace

nested class part1

```
public class D
{
  public int x = 10;
  public class E
  {
    public int y = 30;
  }
}
```


```
public class D
{
  public int x = 10;
}
public class D.E
{
  public int y = 30;
}
```

```
D d1 = new D();
D.E e1 = new D.E();
E e2= new E();
```

partial class

using partial class we can split single class definition into multiple files.

during compilation time all partial classes definitions which are present inside same name space will merge into single class.

Same project

File1.cs

```
namespace N1
{
 public partial class A
 {
      2V+2M
 }
}
```

File2.cs

```
namespace N1
{
  public partial class A
  {
    3V+4M
  }
 }
```

C# Compiler

```
public class N1.A
{
  5V+6M
}
```

abstract class

- abstract class is a class which contains **zero or more abstract members**(methods, properties, indexer or events)
- we can’t create an object for abstract class.
- no Inheritance restrictions for abstract class (It can act as parent or child class)

abstract class Part1

```
public abstract class A
{
    IV+IM+SV+SM
    IC+SC+VM+AM+OM+PROPERTIES
    +INDEXES
}
static void Main(string[] args)
{
    A a = new A();
}
```

Compilation error

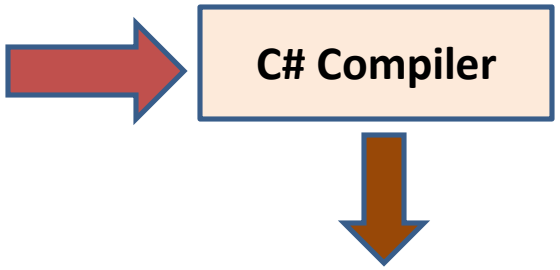
NOT ALLOWED TO CREATE OBJECT FOR ABSTRACT CLASS

abstract method part1

- abstract method is a method which will be having only method header without body.
- a method which is not having body technically called as un-implemented methods.
- during compilation time all abstract methods are converted to pure virtual methods.
- a pure virtual method is a method which is having only header without body.

abstract method part 2

```
public abstract class B
{
 public abstract int m1();
}
```

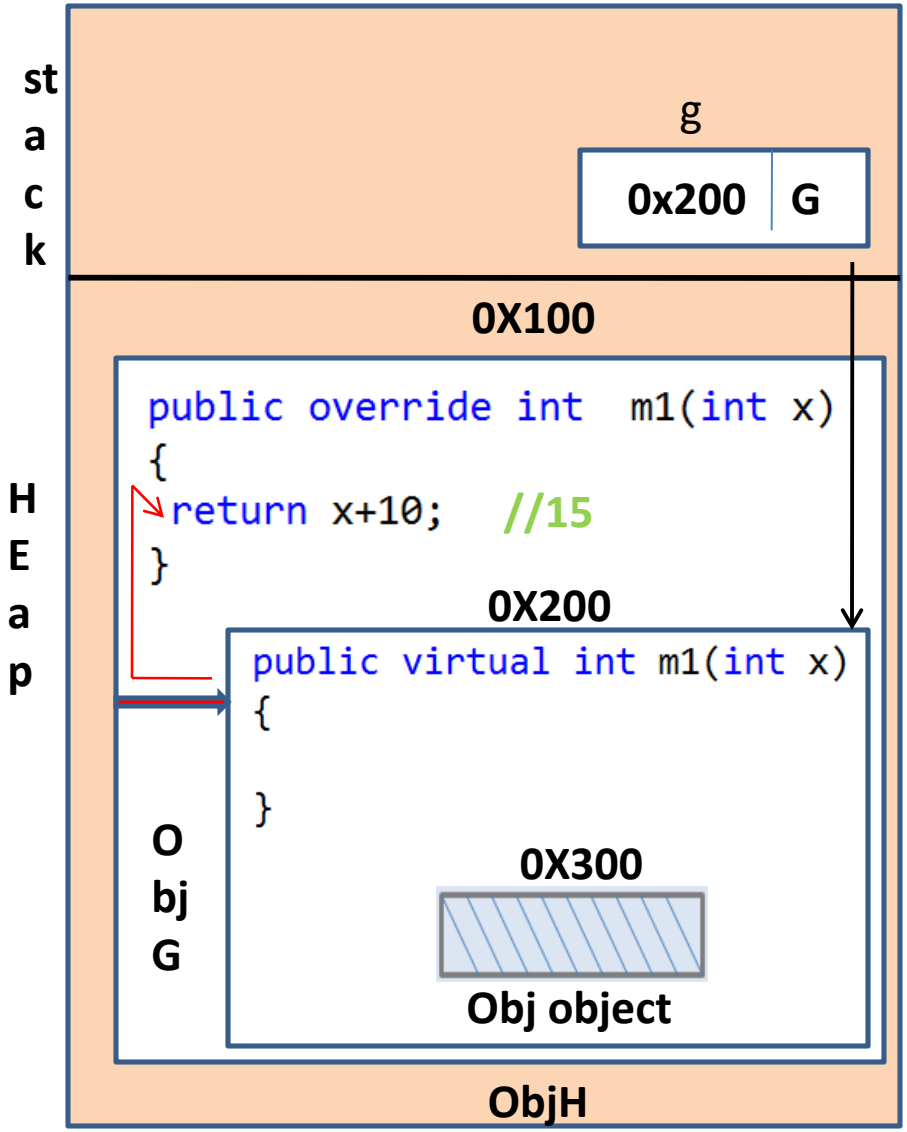



```
public  virtual   int m1();
```

pure virtual method: a virtual method without body called as pure virtual

abstract class sample

```
public abstract class G
{
    public abstract int m1(int x);
}
public class H:G
{
 public override int  m1(int x)
 {
  return x+10;
 }
}
static void Main(string[] args)
{
 G g = new H();
 int r1=g.m1(5); //15
}
```



What is the output?Draw Ram architecture

abstract class part2

```
public abstract class E
{
  public abstract int m2();
  public abstract int m3(int x, int y);
}
public class F : E
{
  public override int m2()
  {
    return 70;
  }
}
```

Compilation error ✗

Code will work or not?

as per c# rule derived class must give implementation for all abstract members present in parent class other wise we will get compilation error

How to escape from this error???

abstract class part 3

```
public abstract class H
{
    public abstract int m4();
    public abstract void m5(int x, int y);
}
public class J : H
{
    public override int m4()
    {
        return 60;
    }
}
public abstract class K : H
{
    public override void m5(int x, int y)
    {

    }
}
```

in general every child class must give implementation for all abstract members present in parent class

by declaring child class as abstract class you can escape from above rule

How many compilation error??

interface

1. Interface can contain only methods without body.
2. Interface can’t contain i.v/s.v
3. we can’t create object for Interface
4. Interface members can’t contain access modifier

Interface Part2

```
public interface I1
{
```

IV+ IC+ IM+ SV+ SC+ SM+ VM+ OM ✗

Unimplemented methods ✓

```
}
```

```
I1 i1 = new I1();
```

✗

```
I1 i2;
```

✓

Run time instance creation

Compile time instance creation

NOTE:

1.Run time instance creation is not possible in **abstract class** and in **static class**

2.Compile time instance creation and runtime creation is not possible in **static class**

Interface part3

```
public interface I1
{
  public int x = 10;
  public static int y = 20;
  public Const int z = 30;
  public ReadOnly int k = 40;
  public int m1(int x1,int y1)
  {
      return x1 + y1;
  }
  public Static int m2(int x2,int y2)
  {
      return x2 - y2;
  }
  public int m3(int x3, int y3);
  int m4(int x4, int y4);
  string m4(string s1, string s2);
}
```

Compilation e

Co

Compil

Compilation error

Compilation error

Compilation error

Compilation error

```
static void Main(string[] args)
{
  I1 i1 = new I1();
  I1 i2;
}
```

Compilation error

interface implementation

```
public interface I2
{
  string m3(int x);
}
public class B : I2
{

   string I2.m3(int x1)
    {
     return "given no is" + x1;
    }
}
```

```
static void Main(string[] args)
{
   B b1 = new B();
   I2 i2 = b1;
   string s1 = b1.m3(20);
   string s3=i2.m3(20);
}
```

Compilation error

NOTE:

all interface members must be called by using interface variable ,but not by using object name.

implementing multiple interfaces

```
public interface I3
{
   void m3();
}
public interface I4
{
   int m4();
}
```

```
public class E : I3, I4
{
  void I3.m3()
  {
   Console.WriteLine("Hello");
  }
  int I4.m4()
  {
    return 77;
  }
}
```

calling Implemented methods

```
E   e = new  E( );
I3  i3 = e;
i3.m3();
I4  i4 = e;
i4.m4();
```

inheritance & implementation at a time

```
public interface I20
{
 int m1(int x, int y);
}
```

```
public class A
{

}
```

```
public class B : I20, A
{

}
```

Compilation error

```
public class D : A,I20
{

}
```

Note: As per C# language rules we must give first priority to inheritance and last priority to Implementation.

interface assignment

```
public interface I30
{
    string m2();
}

public class H : I30
{

}
```

```
static void Main(string[] args)
{

}
```

Call the method m2()

Implement m2( ) in class H

interface assignment-solution

```
public interface I30
{
    string m2();
}

public class H : I30
{
  string I30.m2()
  {
      return "hai";
  }
}
```

```
static void Main(string[] args)
{
    H h = new H();
    I30 i30 = h;
    string str=i30.m2();
}
```

when to use abstract class

- It is recommended to **use** abstract class when we know implementation for some methods & when we don't know implementation for some methods

when to use interface

- interface is useful for avoiding code dependency between two developers.

OR

- interface acts as a contract/agreement between two developers

interface vs abstract class

abstract class	interface
supports implemented & un-implemented methods	supports only un-implemented methods
will support **versioning**	will not support **versioning**

note : versioning refers to the back ward compatibility (if we add new implemented method in abstract class the class which is implementing it will not be affected)

note : if we add new un- implemented method in interface the implementing class will be affected)

properties

- properties are useful for reading & modifying data present in private variables
- a getter portion of a property will be useful for reading data present in a private variable
- a setter portion of a property will be useful for assigning data into private variable.

properties

```
public class A
{
    private int _x;
    public int X
    {
        get
        {
            return _x;
        }
        set
        {
            _x = value;
        }
    }
}
```

90

90

90

90

90

```
static void Main(string[] args)
{
    A a = new A();
    a._x = 20;
    int r = a._x;
    a.X = 90;
    int r1 = a.X;
}
```

90

types of properties

1. Read only properties (property which is having only get)
2. Write only properties (property which is having only set)
3. Read Write properties (property which is having get & set)

```
public class A
{
    private int _x;
    public int X
    {
     get { return _x; }
     set { _x = value;}
    }
}
```

Read only properties

write only properties

Read write Properties

access modifiers for get & set

Q) Is it possible to change access modifiers for both get & set in a read-write property?

answer→ no (in a read write property we can change access modifier either for getter or for setter not for both).

- Note: in a read write property either get or set must get access modifier from property definition.

access modifiers for get & set – part2

```
public class A
{
    private int _x;
    public int X
    {
        private get
        {
            return _x;
        }
        public set
        {
            _x = value;
        }
    }
}
```

```
public class A
{
    private int _x;
    public int X
    {
        private get
        {
            return _x;
        }
        protected set
        {
            _x = value;
        }
    }
}
```

properties assignment 1

```
public class F
{
  private int _r = 60;
  public int R
  {
   get { return _r; }
  }
}
```

```
static void Main(string[] args)
{
  F f = new F();
  int y = f.R;
}
```

Req: store _r variable data in y variable?

properties assignment 2

```
static void Main(string[] args)
{
  D d1 = new D();
  d1.X = 20;
  d1.Y = "palle";
  d1.Y = "palle";
  int r1 = d1.X; Return //40
  String r2 = d1.Y; Return string except first and last character// all
}
```

property assignment solution

```
static void Main(string[] args)
{
   D d1 = new D();
   d1.X = 20;
   d1.Y = "palle";
   int r1 = d1.X;
   String r2 = d1.Y;
}
public class D
{
  private int _x;
  private string _y;
  public int X
  {
      get { return _x * 2; }
      set { _x = value; }
  }
}
```

auto implemented properties

compiler will automatically create private variable and the body for getter and setter incase of auto implemented properties

auto implemented properties

```
class A
{
    private int _x;
    public int X
    {
        set;

    }
}
```

```
public class A
{
    private string _ ?;
    public string S
    {
        get;
        set;
    }
}
```

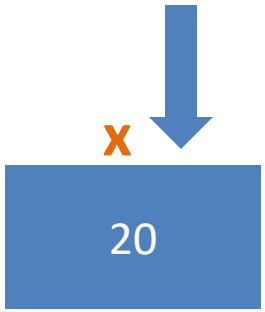
C# compiler

```
public String S
{
    get
    {
        return ?;
    }
    set
    {
        ?; = value;
    }
}
```

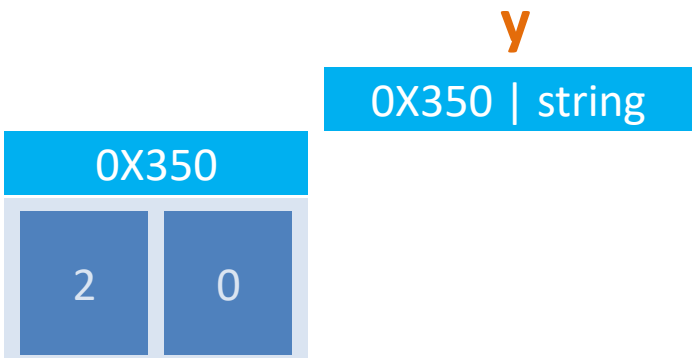
value type and reference type

value type	reference type
variable will be directly holding data both variable name and data are stored in same memory location all fixed size data types + struct + enum are value types	variable will be directly holding address both variable name and data are stored In different memory location all varying size data types are reference type except struct and enum

int x = 20;



string y = “20”;



structures

- structures are similar to classes
- structures will not support initialized instance variables
- structures will not support parameter less constructors
- structures will not support inheritance
- structures will support interface implementation
- structures are value types & classes are reference types

difference between structures & class

classes supports

initialized + un initialized instance variables
+
methods
+
Constructors
+
properties

structures supports

uninitialized instance variables
+
methods
+
parameterized Constructors
+
properties

structure inheritance

```
public struct A
 {
 }
 public class B : A
 {
 }
 public struct D:A
 {
 }
 public class E
 {
 }
 public struct F : E
 {
 }
```

Inheritance in any way is not possible

structure assignment

```
public struct A
{
    public int x = 20;
    public int y;
    public static int z = 40;
    public A()
    {
        y = 80;
    }
    public A(int y1)
    {
        this.y = y1;
    }
    public int m1(int i1)
    {
        return 20;
    }
    public static int m2()
    {
        return 40;
    }
}
```

How many mistakes are present in the code?

creating structure object

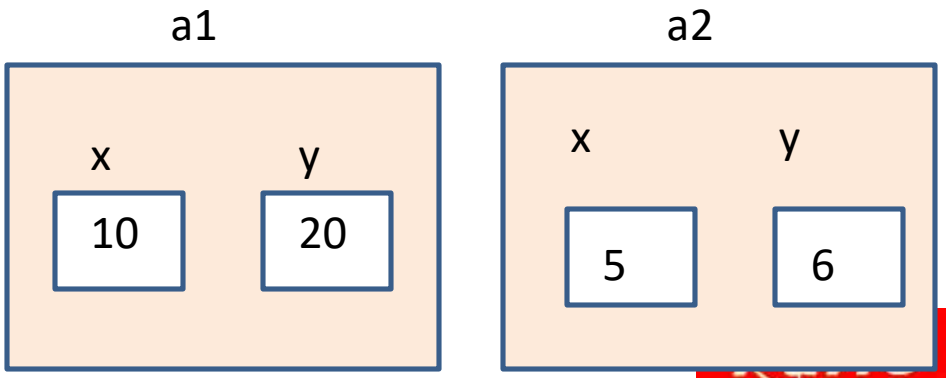
c# supports 2 ways for creating struct object

1.we can create structure object by using new keyword

2. without using new keyword

```
public struct A
{
  public int x;
  public int y;
  public A(int x, int y)
   {
    this.x = x;
    this.y = y;
   }
}
```

```
static void Main(string[] args)
{
  A a1 = new A(10, 20);
  A a2;
  a2.x = 5;
  a2.y = 6;
}
```



structures and interfaces

```
public interface I1
{
    int m10(int x, int y);
}

public struct E:I1
{

    int I1.m10(int x, int y)
    {
        return 10;
    }

}
```

Structure can implement one or more interfaces

constant

- a constant must be declared by using const keyword
- we can't modify the data present in constant
- we must assign data into the constant variables wherever we are declaring constant variable
- constant members must be accessed by using classname dot (since all const members converted to static members during compilation)

constant lab (find mistakes)

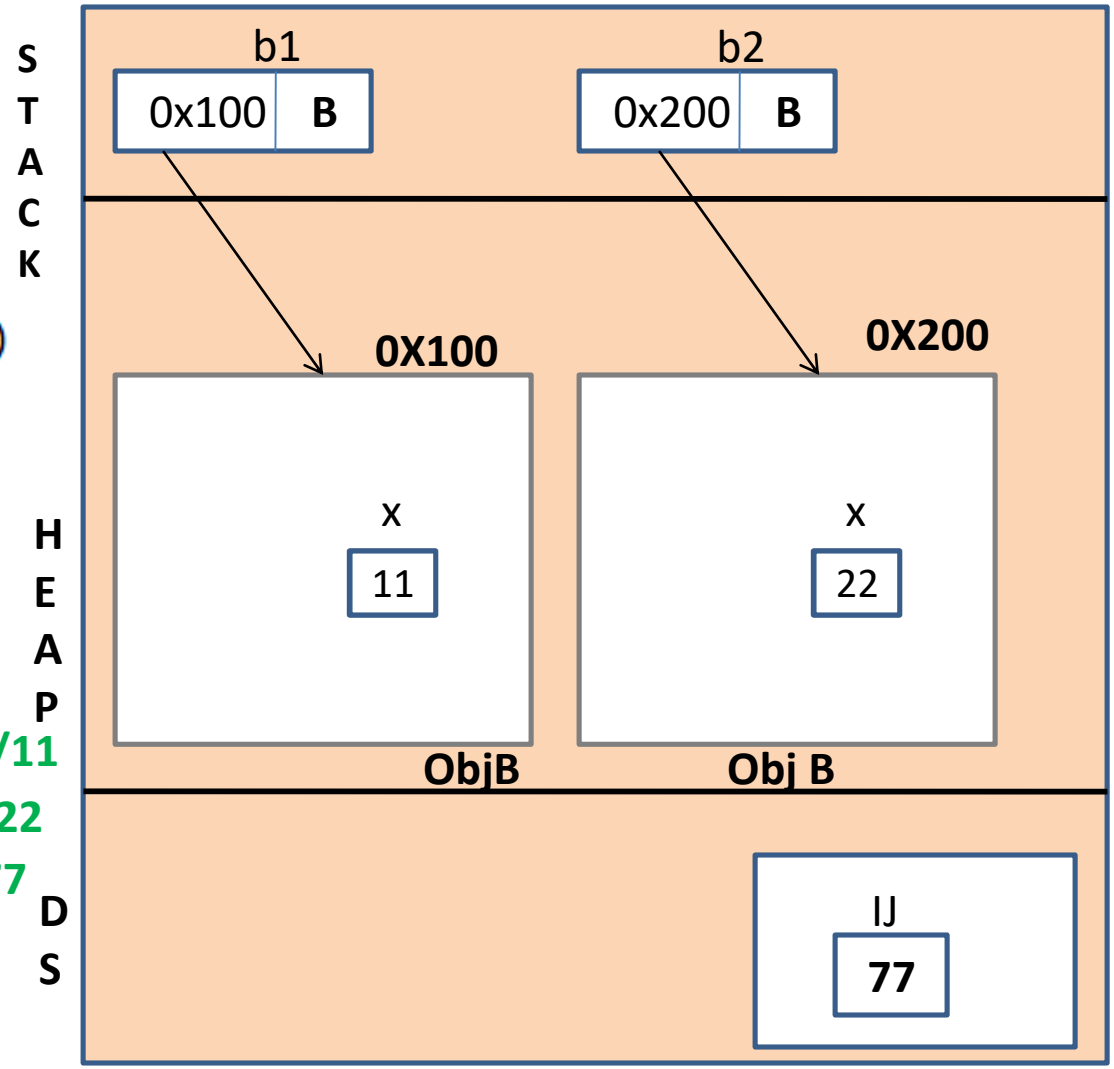
```
public class A
{
    public int x;
    public int y = 20;
    public static int r;
    public static int j = 70;
    public const int I;
    public const int L = 70;
    public A()
    {
        x = 5;
        y = 46;
        A.r = 9;
        A.j = 65;
        A.I = 4;
        A.L = 8;
    }
}
```

Constant variable has to be initialized

Constant variable cannot be modified

constant sample

```
public class B
{
    public int x = 6;
    public const int IJ = 77;
}

static void Main(string[] args)
{
    B b1 = new B();
    B b2 = new B();
    b1.x = 11;
    b2.x = 22;
    Console.WriteLine(b1.x);  //11
    Console.WriteLine(b2.x);  //22
    Console.WriteLine(B.IJ);  //77
}
```



read only variable

- readonly variable is a object specific constant
- we can assign data into read only variables while declaring the variable or in the same class constructor.
- we can re assign data into read only variable in the same class constructor
- read only variables are instance variables and hence must be accessed by using object name [dot]

readonly variable sample

```
public class A
{
  public readonly int x = 10;
  public readonly int y;
  public readonly int z = 30;
  public A(int r1, int r2)
  {
    this.y = r1;
    this.z = r2;
  }
  public void m3()
  {
    x = 11;
  }
}
static void Main(string[] args)
{  A a = new A(10,20);
   Console.WriteLine(a.x);  //10
   Console.WriteLine(a.y);  //10
   Console.WriteLine(a.z);  //20
}
```

You can assign data into readonly variables in same place,but not mandatory

Req:Whatever comes in r1,r2 I would like to assign y,z variables

You can reassign the data into same variable inside constructor

Compilation error

Whenever you declare readonly variables there you can assign data or else in constructor.But not outside any other place

read only-assignment

```
public class B
{
 public readonly int R= 80;
 public int J = 90;
 public static int K = 100;
 public const int L = 110;
 public B(int i1, int i2, int i3, int i4)
 {
   R = i1;
   J = i2;
   K = i3;
   L = i4;
 }
 public void m3()
 {
   R = R + 1;
   J = J + 1;
   K = K + 1;
   L = L + 1;
 }
}
```

```
static void Main(string[] args)
{
  B b1 = new B();
  b1.R = 12;
  b1.J = 13;
  B.K = 14;
  B.L = 15;
}
```

How many mistakes are present in the code?

enum

1. By using Enums we can group set of related constants
2. Enum wont support any other type of variables declarations and also enums will not support methods
3. Enum will not participate in '**inheritance**'
4. Enums will not support the object creation

enum sample1

Declaration part:

```
public enum A
{
  X,Y,R
}
static void Main(string[] args)
{
  Console.WriteLine(A.X); //X
  Console.WriteLine((int)A.X);//0
  Console.WriteLine((int)A.Y);//1
  Console.WriteLine((int)A.R);//2
}
```


```
public enum A
{
  public const int X=0;
  public const int y=1;
  public const int R=2;
}
```

enum sample2

```
public enum D
{
  X1=77,
  X2,
  X3,
  X4=96,
  X5,
  X6=77,
  X7
}
```

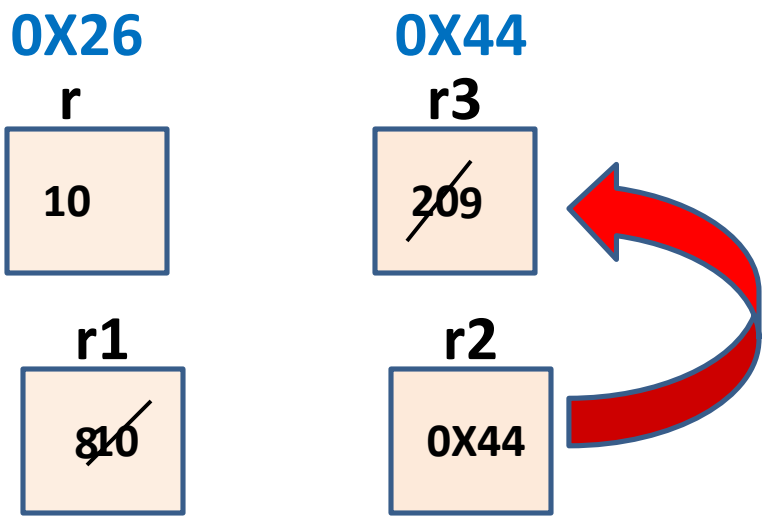
```
Console.WriteLine((int)D.X1);//77
Console.WriteLine((int)D.X2);//78
Console.WriteLine((int)D.X3);//79
Console.WriteLine((int)D.X4);//96
Console.WriteLine((int)D.X5);//97
Console.WriteLine((int)D.X6);//77
Console.WriteLine((int)D.X7);//78
```

ref Keyword

Using **ref** keyword we can pass address of a variable to another variable. When we pass a variable to a method usually data present in variable will be passed to the receiving variable.

For passing address of a variable to another variable we must use **ref** keyword

```
static void Main(string[] args)
{
  int r = 10;
  int r3 = 20;
  A a1 = new A();
  a1.m1(r, ref r3);
}
public class A
{
 public void m1(int r1,ref int r2)
  {
    r1 = 8;
    r2 = 9;
  }
}
```



out keyword

- using out keyword we can pass address of a variable
- we don’t need to initialize out variable before passing

out keyword and ref keyword

```
static void Main(string[] args)
{
    int i;
    A a = new A();
    a.m1(ref i);
}
```

No data present in i variable

Compilation error

```
public class A
{
    public void m1(ref int x)
    {

    }
}
```

Ref variable address

```
static void Main(string[] args)
{
    int j;
    B b = new B();
    b.m1(out j);
}
```

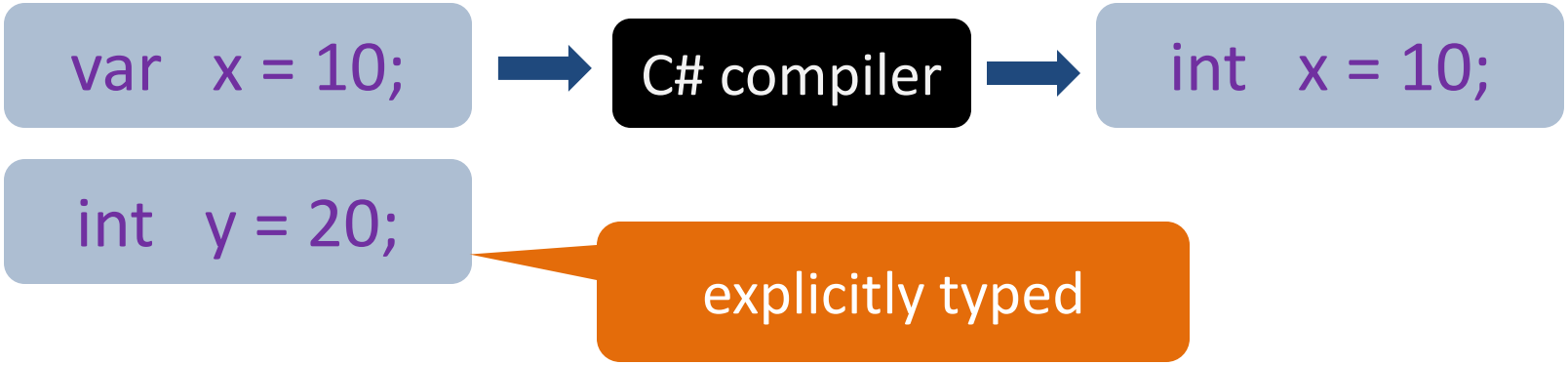
```
public class B
{
    public void m1(out int y)
    {

    }
}
```

Compilation error

the output parameter y must be assigned in the received function body

var keyword

- var can be used for declaring implicitly typed method scope local variables.



var keyword assignment

```
public class A
{

}
public class B
{
  public var i=10;
  public void m2(var x,int y)
  {
   int z=20;
   var a=new A();
   var b = 9;
  }
 }
```

How many mistakes are there in this code??

Compilation error

var type variables are allowed in method body

var type variables are not allowed in method header

C# compiler

C# compiler

```
A a = new A();
int b = 9;
```

for each loop

1. foreach loop used for reading data present in collection from beginning to end
2. foreach loop execution is faster then for loop execution

syntax

```
foreach(IndividualitemDataType  variablename  in  collectionname)
{
        -----
        C# code
}       -----
```

for each example

Req:Print the data present in array using foreach loop

```
static void Main(string[] args)
{
int[] x = new int[] { 10, 20, 60, 80 };
foreach (int j in x)
  {
    Console.WriteLine(j);
  }
    Console.ReadLine();
}
```

x

10	20	60	80

output

file:///

```
10
20
60
80
```

for each assignment

```
string[] s = new String[] { "ab", "cd", "ef", "gh" };

foreach (string i in s)
{
   Console.WriteLine(i);
}

   Console.ReadLine();
```

Req:Read the items present in string array

Iteration variable

Output

ab
cd
ef
gh

Is it possible to modify the data present in iterative Collection?

Is it possible to modify the data present in iterative variable?

```
foreach (string i in s)
{
   s[3] = "xz";
   if (i == "cd")
   {
     i = "xy";
   }
}
```

Compilation error

object class

all classes present in c# must directly or indirectly **inherit** from Object class.

Object class object can store any type of data and hence it is considered as universal datatype.

Eg: object o = 10;

int value

o

10

Object class object

object o1 = “palle”;

o1

0x300 | string

Object class object

0x300

p a l l e

object o2 = true;

o2

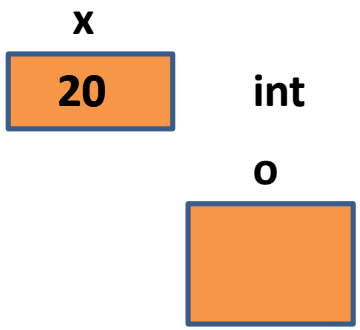
true

Object class object

boxing & unboxing

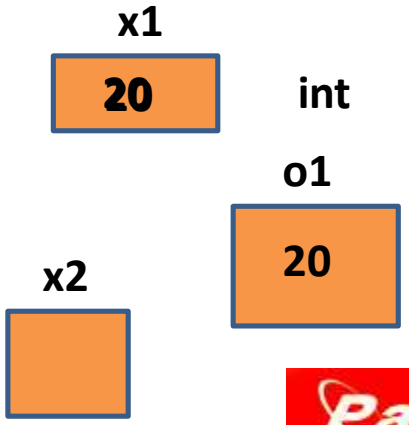
Boxing: Copying data from value type to object type.

```
int x = 20;

object o = x;  // Boxing
```



UnBoxing: Copying data from object type to value type.

```
int x1 = 20;

object o1 = x1;    // Boxing

int x2 = (int)o1;  // UnBoxing
```



comments

comments will make the code understandable to the other programmers

1. single line comment
2. multi line comment
3. xml comment/documentation comment

Single line comment:

//---------- single line comment

Multi line comment:

```
/*-----
  ------
   -------
*/
```

Multiline line comment

xml comment:

///.....
///.....
///.....

When you are creating a class/method.It will help you to describe what your method/classes doing

exceptions

- Exception is a runtime error.
- Exception will terminate the program abruptly if we don’t handle.

exception internals

```
static void Main(string[] args)
{
  -------- L1
  -------- L2
  -------- L3
  string data = File.ReadAllText("D:\\student.txt");
  Console.WriteLine(data);

  --------L6
  --------L7
}
```

0X100

0X200

CN=...

MN=....

Line=.... Time=..

Unknown situation	class
File not available in the specified location	FileNotFoundException
Denominator Zero	DivideByZeroException
.....	.....

```
FileNotFoundException ex = new FileNotFoundException();
```

How to stop abrupt termination

```
static void Main(string[] args)
{
  try
  {
  -------- L1 ✓
  -------- L2 ✓
  -------- L3 ✓
  string data = File.ReadAllText("D:\\student.txt");
  Console.WriteLine(data);

  --------L6
  --------L7
  }
  catch(FileNotFoundException ex)
  {

  }

}
```

0X100

0X200

CN=...

MN=....

Line=.... Time=..

try catch finally rules

```
try
{
    ------
    ------
    ------
}
```

```
try
{
  ------
  ------
}
catch (Exception ex)
{
  ------
  ------
}
finally
{
  ------
  ------
}
```

```
try
{
  ------
  ------
}
catch (Exception ex)
{
  ------
  ------
}
```

```
try
{
  ------
  ------
}
catch (Exception ex)
{
  ------
  ------
}
finally
{
  ------
  ------
}
```

```
try
{
  ------
  ------
}
finally
{
  ------
  ------
}
```

```
finally
{
  ------
  ------
}
```

1.You are allowed to write 0 or 1 finally block for a given try

2.You are allowed to write 0 or more catch blocks in a given try block

nested try catch blocks part1

Is it possible to write nested try

```
try
{
  try
  {
   }
   catch (Exception ex)
   {

   }
}
```

Is it possible to write nested catch

```
try
{

 }
catch (Exception ex)
{
    try
    {

     }
     catch (Exception ex)
     {

     }
  }
```

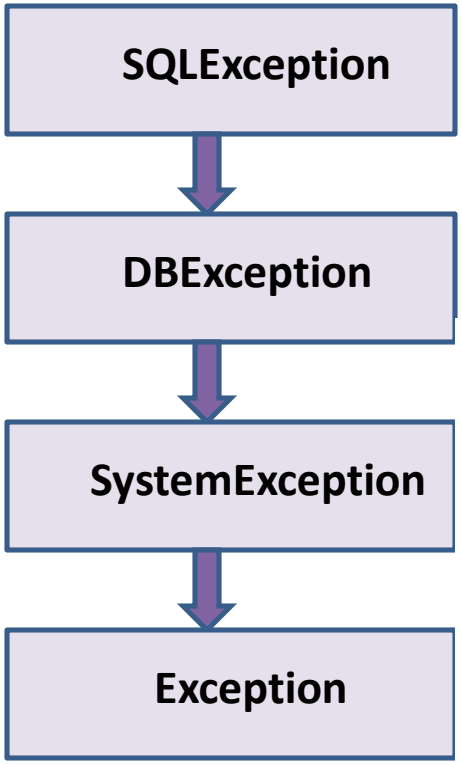
nested try catch blocks part2

```
try
{   ------
    ------
 }
 catch (Exception ex)
 {
   ------
   ------
 }
finally
{
   ------
   ------
   try
   {

   }
   catch (Exception ex)
   {

   }
}
```

multiple catch blocks

- we are allowed to create multiple catch blocks for a given try block.
- always catch block must flow from child to parent.

writing multiple catch blocks




```
try
{
  ------
  ------
}
catch (DBException ex1)
{
  ------
  ------
}
catch (SystemException ex2)
{
  ------
  ------
}
catch (Exception ex3)
{
  ------
  ------
}
catch (SQLException ex4)
{
  ------
  ------
}
```

```
try
{
  ------
  ------
}
catch (SQLException ex4)
{
  ------
  ------
}
catch (DBException ex1)
{
  ------
  ------
}
catch (SystemException ex2)
{
  ------
  ------
}
catch (Exception ex3)
{
  ------
  ------
}
```

finally

1. the code present in the finally block will be always executed (even in presence of return statement in try | catch blocks)
2. we can’t write return statement in finally

finally - 1

```
try
{
    Console.WriteLine("hi");
    File.ReadAllText("D:\\xyz.txt");
    Console.WriteLine("hello");
}

catch (Exception e)
{
    Console.WriteLine("handled exception");
}

Console.WriteLine("bye");
```

Exception generated

normally **the statements written after the catch block will be executed**

finally - 2

```
try
{
    Console.WriteLine("hi");
    File.ReadAllText("D:\\xyz.txt");
    Console.WriteLine("hello");
    return 10;
}
catch (Exception e)
{
    Console.WriteLine("handled exception");
    return 0;
}

Console.WriteLine("bye");
```

Exception generated

If the try catch block contains return statement , then the statements written after the catch block will not executed.

finally - 3

```
try
{
    Console.WriteLine("hi");
    File.ReadAllText("D:\\xyz.txt");
    Console.WriteLine("hello");
    return 10;
}
catch (Exception e)
{
    Console.WriteLine("handled exception");
    return 0;
}
finally
{
    Console.WriteLine("Bye");
}
```

Exception generated

the code present in finally will be always executed (even when return statements are present in try | catch)

Thank you