Run-time Polymorphism

Polymorphism

- Compile time polymorphism — Method overloading
- Run-time polymorphism — Method overriding

Binding

method overloading

```
int sum(int x, int y)
{
}
float sum(float x, float y)
{
}
```

Binding → compiler, Run-time

sum(10,20)

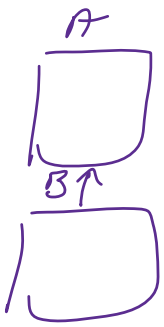
```
class A
{
    int sum(int x, int y)
    {
    }
    float sum(float x, float y)
    {
    }
}
class B extends A
{
    int sum(int x, int y, int z)
    {
    }
}
```

Compiles

sum(10,20,30)

```
class A
{
    int sum(int x, int y)
    {
    }
}
class B extends A
{
    int sum(int x, int y)
    {
    }
}
A x;
int ch = Scanner.nextInt();
if(ch == 1)
    x = new A();
else
    x = new B();
x.sum(10,20);
```

x = new A();

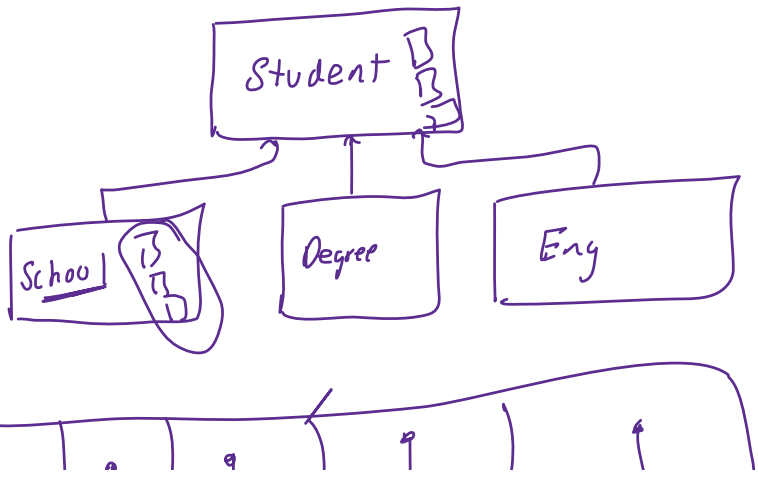


B x

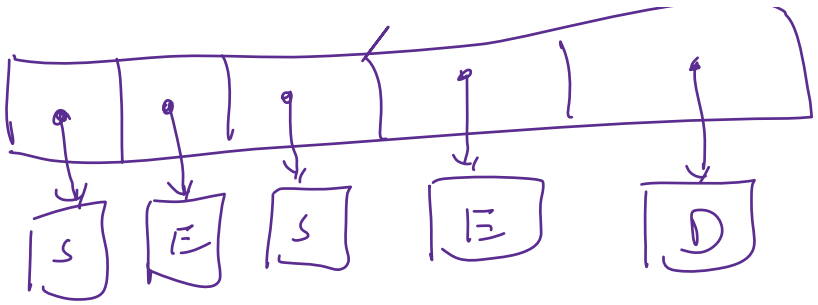
A x;
x = new A();
x = new B();

B x;
x = new B();

interface StudentRepository
{
}



S
=
S

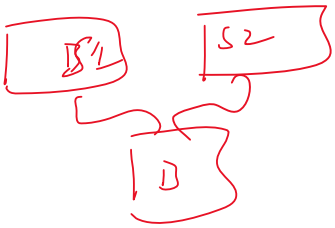


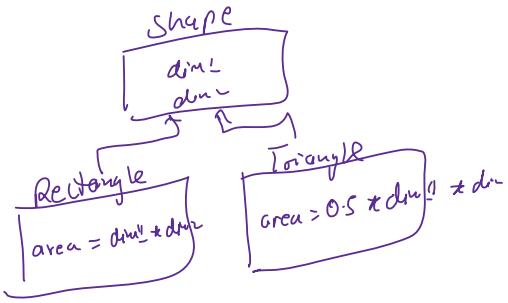
Student[] stud = new Student[5];

```
abstract class shape
{
    protected int dim1, dim2;
final public void setDim1(int dim1)
    { this.dim1 = dim1;
    }
final public int getDim1()
    { return dim1;
    }
final public void setDim2(int dim2)
    { this.dim2 = dim2;
    }
final public int getDim2()
    { return dim2;
    }
abstract public float getArea();
}
```

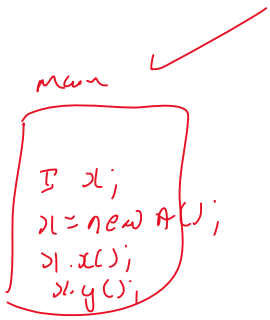
```
final class Rectangle extends Shape
{   private float area;
    public float getArea()
    {   area = dim1 * dim2;
        return area;
    }
}
class Triangle extends Shape
{   private float area;
    public float getArea()
    {   area = 0.5f * dim1 * dim2;
        return area;
    }
}
```

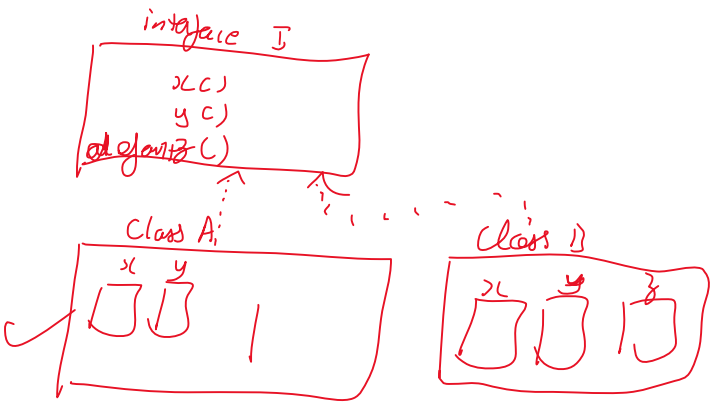
```
class Boc extend Rectangle
{

}
```

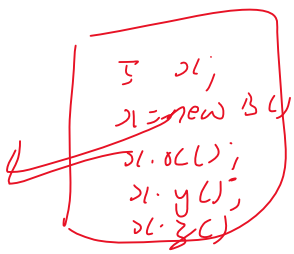


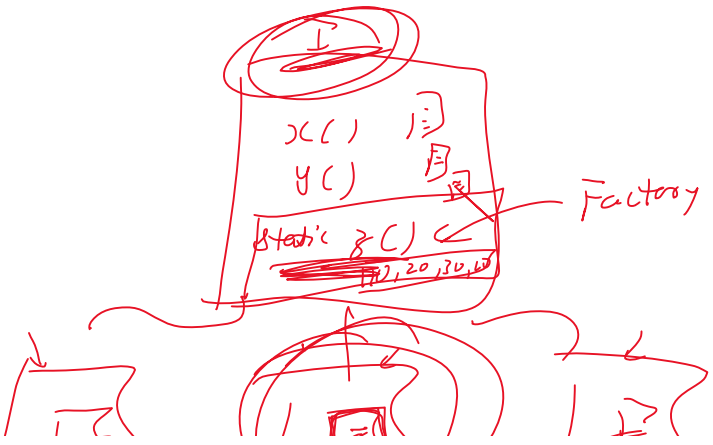



```
Class Main
{   PSVM(..)
    {   Shape shape;
        int ch = Scan.nextInt();
        if(ch == 1)
            shape = new Rectangle();
        else
            shape = new Triangle();
        shape.setDim1(10);
        shape.setDim2(20);
        S.O.P("Area:" + shape.getArea());
    }
}
```









Stream.of(10,20,30,40,

ICircle getPi

interface

ICircle

public float Pi

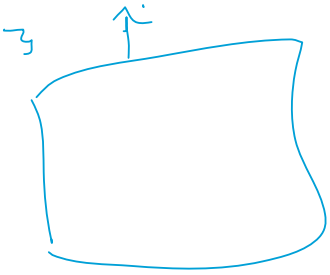
public static float getPi()

{

return (Pi);

}

}



```
interface IDisplay
{
    void displayMessage(String msg);
}
class Display implements IDisplay
{
    public void displayMessage(String msg)
    {
        S.O.P(msg);
    }
}
class Main
{
    P S V main(String args)
    {
        IDisplay ref = new Display();
        ref.displayMessage("Hello world");
    }
}
```

```
class
  interface IDisplay
  {
      void displayMessage(String msg);
  }
abstract
```

```
Class Main
{
    P S V main(..)
    {
        IDisplay ref = new IDisplay()
        {
            public void displayMessage(String msg)
            {
                S.O.P(msg);
            }
        };
```

class ☒ implements IDisplay

class ☒ extends IDisplay

@FunctionalInterface

```
interface IDisplay
{
    void displayMessage(String msg);
}
```

```
class XYZ
{
    public static void disp(String msg)
    {
        S.O.P("Message:" +msg);
    }
}
```

```
class Main
{
    P S V main(..)
    {
        IDisplay ref = 'XYZ::disp;
        ref.displayMessage("Hello world");
```

```
interface IDisplay
{
    void displayMessage(String msg);
}

class Main
{
    p s v m(-)
    {
        IDisplay ref = Display::disp;
        ref.displayMessage("Hello");

        IDisplay ref2 = Display2::mydisp;
        ref2.displayMessage("world");
    }
}
```

```
Class Display
{
  public static void disp(String msg)
  {
      S.O.P("Message :" +msg);
  }
}
```

```
class Display2
{
  public static void mydisp(String msg)
  {
      S.O.P("###" +msg+ "###");
  }
}
```

```
Class SimpleInterest
{
    public static float calculateSimpleInterest(int p, float t, float r)
    {
        return (p*t*r/100);
    }
}
Class CompoundInterest
{
    public static float calculateCompoundInterest(int p, float t, float r)
    {
        return p* Math.pow(1+r/100, t);
    }
}

interface Interest
{
    public float calcInterest(int p, float t, float r);
}

class Main
{
    p s v main(..)
    {
        Interest x = SimpleInterest :: calculateSimpleInterest;
        s.o.p("SI :" + x.calcInterest(10000, 2, 14.5f));
        Interest y = CompoundInterest :: calculateCompoundInterest;
        s.o.p("CI :" + y.calcInterest(10000, 2, 14.5f));
    }
}
```

```
packageorg.cgi2;


Class  FindingLS{
      public static int findLargest(int a,int b){
            if(a>b){
                  return a;
            }else{
                  return b;
            }
      }

      public static int findSmallest(int a,int b){
            if(a<b){
                  return a;
            }else{
                  return b;
            }
      }

}

Interface LargestSmallest{
      public int  find(int x,int y);
}

Class LargestSmallestDisplay{

      public void   display(LargestSmallest  ls,String    message,int
      value1,int   value2){
            System.out.println(message+":"+ls.find(value1,value2));
      }
}



public   class  FindLargestSmallest{
      Public static void main(String[] args){

            LargestSmallestDisplay obj = new LargestSmallestDisplay();

            obj.display(FindingLS::findLargest,"Largest ", 25,13);
            Obj.display(FindingLS::findSmallest,"Smallest ",25,13);

}
```

```
interface IArithmetic
{
    public int calculate(int x, int y);
}

class Main
{
    p s v main(String[] args)
    {
        IArithmetic arith1 = ArithmeticFun :: add;
        s.o.p("Sum:" + arith1.calculate(10,20));

        IArithmetic arith2 = ArithmeticFun :: subtract;
        s.o.p("Diff:" + arith2.calculate(10,10));
    }
}
```

```
Class ArithmeticFun
{
    public static int add(int x, int y)
    {
        int z;
        z = x+y;
        return(z);
    }
    public static int subtract(int x, int y)
    {
        int z;
        z = x-y;
        return(z);
    }
}
```

```
interface IArithmetic
{
    public int calculate(int x, int y);
}

class Main
{
    p s v main(String[] args)
    {
        IArithmetic arith1 = (x,y) -> { int z;
                                        z = x+y;
                                        return z;
                                      };

        s.o.p("Sum:" + arith1.calculate(10,20));

        IArithmetic arith2 = (x,y) -> x-y;

        s.o.p("Diff:" + arith2.calculate(10,10));
    }
}
```

parameter

Lambda Exp

```
Class ArithmeticFun
{
    public static int add(int x, int y)
    {
        int z;
        z = x+y;
        return(z);
    }
    public static int substract(int x, int y)
    {
        int z;
        z = x-y;
        return(z);
    }
}
```