Saraswati Vandana

या कुन्देन्दु तुषार हार धवला
या शुभ्र वस्त्रान्विता ।
या वीणा वर दंड मंडितकरा
या श्वेत पद्मासना ॥

या ब्रह्मा अच्युत शंकर प्रभ्रतिभिः
देवै सदा पूजिता ।
सा मां पातु सरस्वती भगवती
निःश्येश जाड्यापह ॥

Faculty of Engineering & Technology
Sankalchand Patel College of Engineering, Visnagar

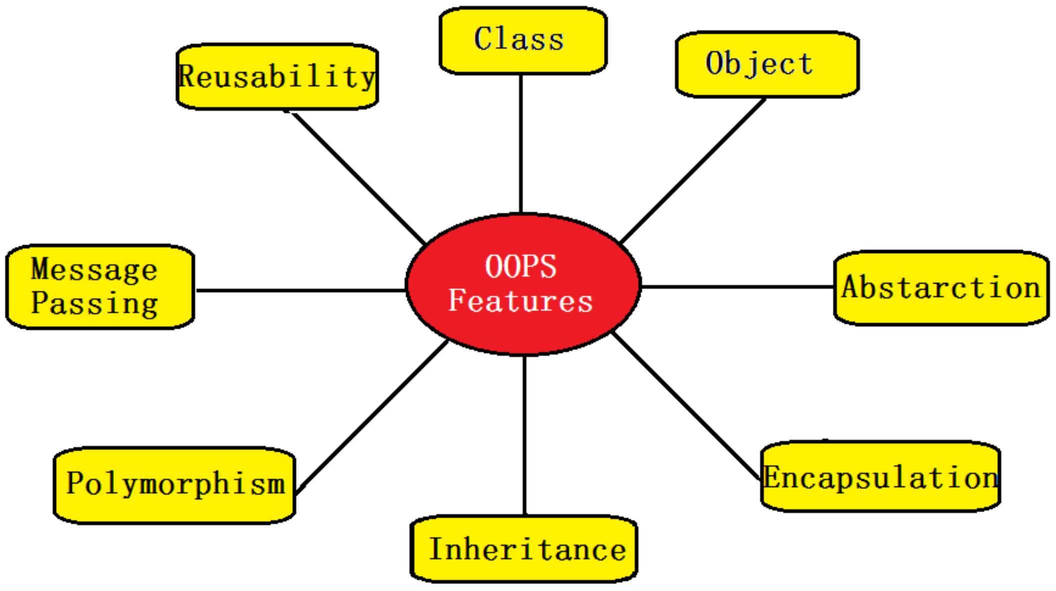
Java Programming
(1ET1030406)

Unit-3 : Classes, Objects and Methods

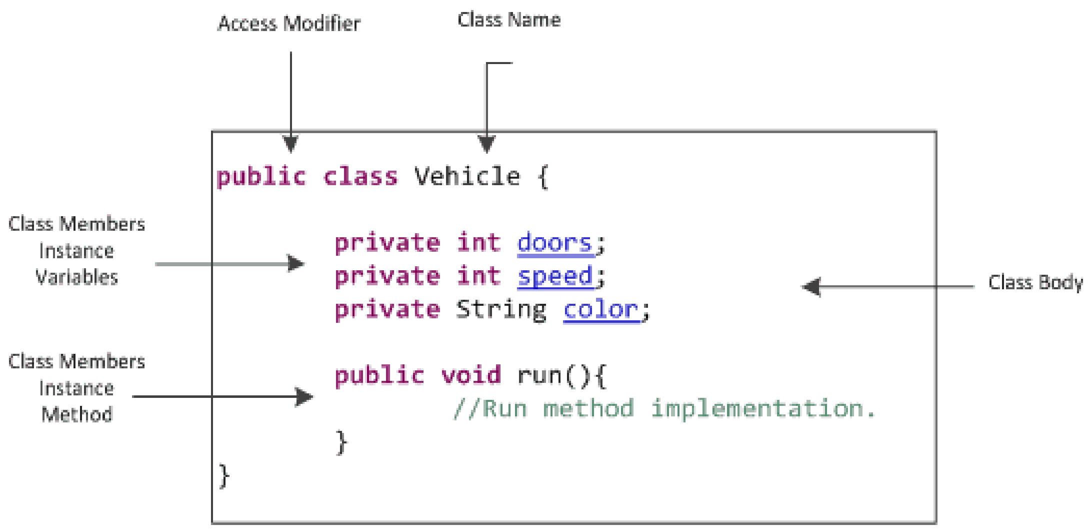
Prepared By
Mr. Mehul S. Patel
Department of Computer Engineering & Information Technology

Content

Feature of OOP

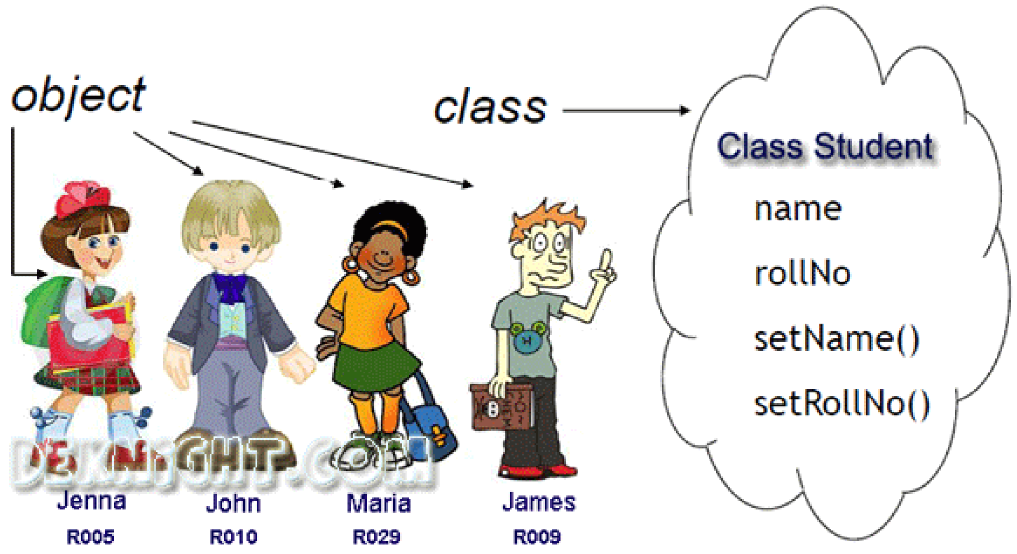


Class


Access Modifier
Class Name
Class Members Instance Variables
Class Members Instance Method
Class Body

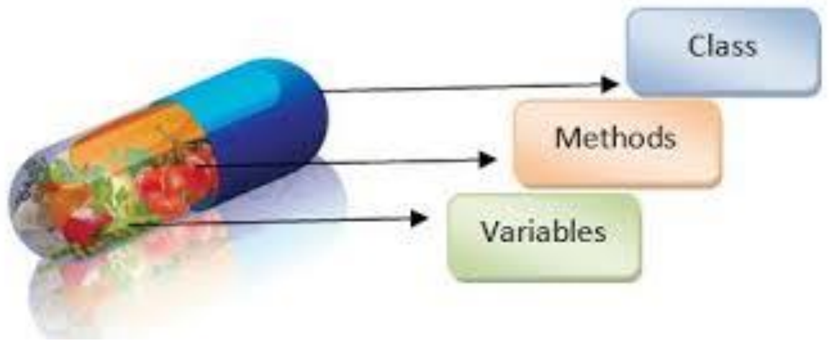
```java
public class Vehicle {

        private int doors;
        private int speed;
        private String color;

        public void run(){
                //Run method implementation.
        }
}
```

Object

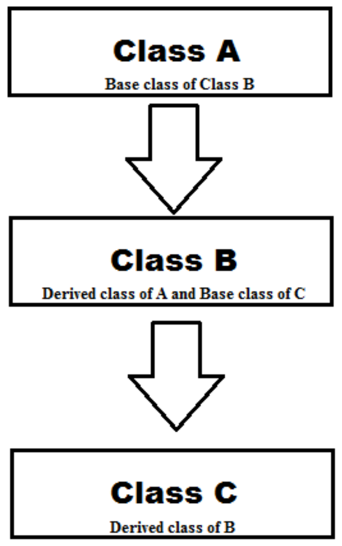

object
class
Class Student
name
rollNo
setName()
setRollNo()
Jenna
R005
John
R010
Maria
R029
James
R009

Abstraction

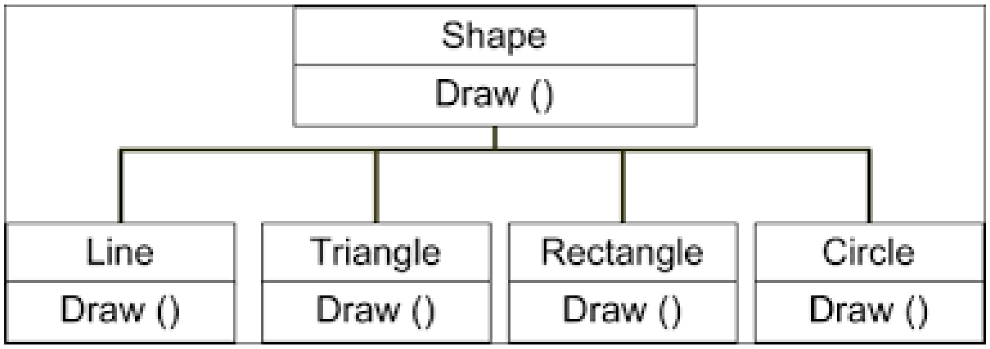
Encapsulation



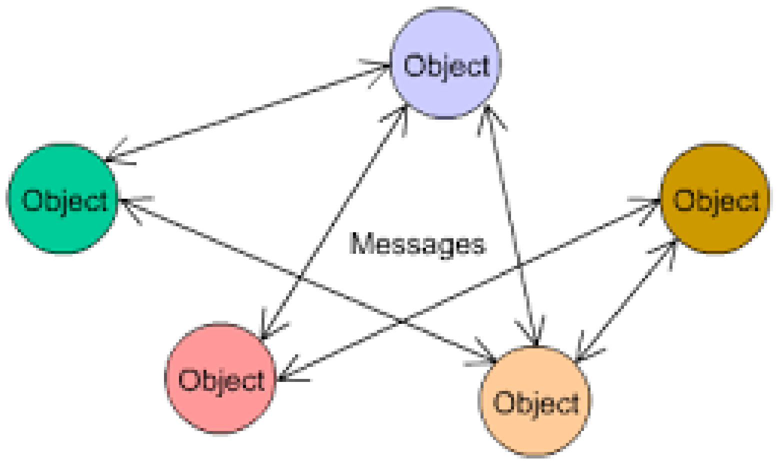
Inheritance



Polymorphism

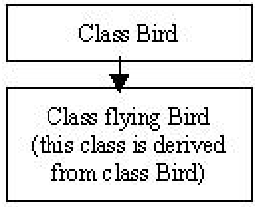


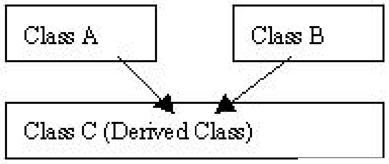
Message Passing

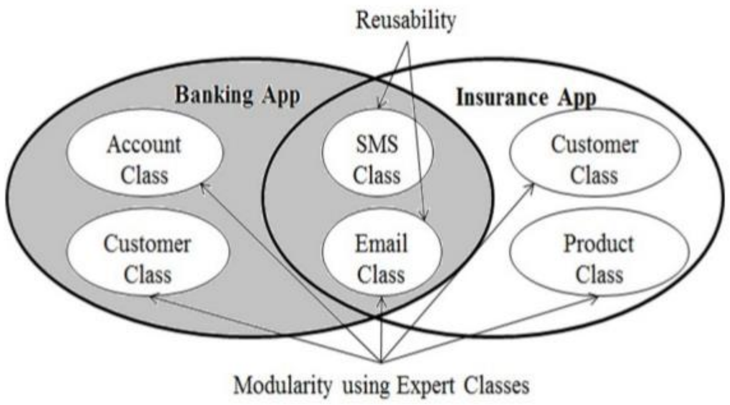


Interaction of objects via message passing

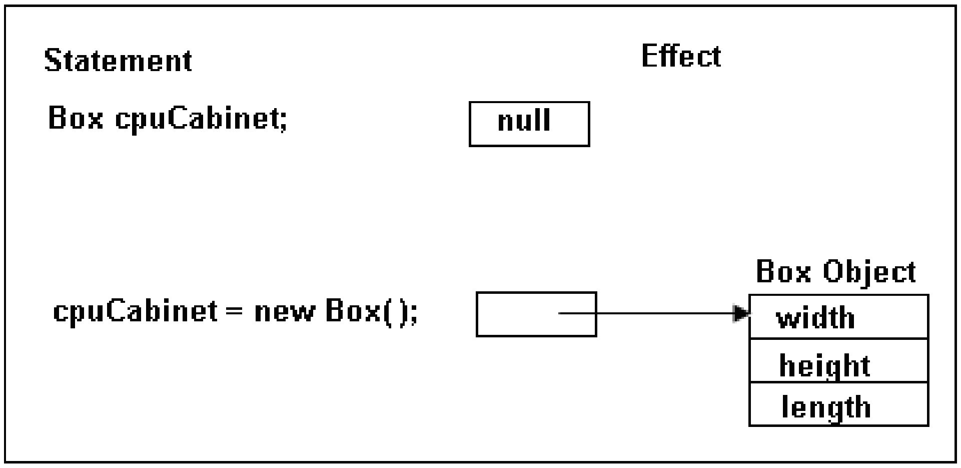
Reusability





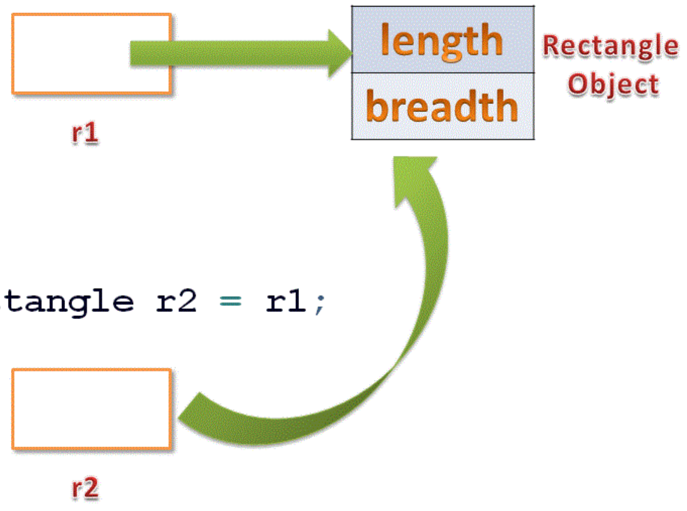


new Operator



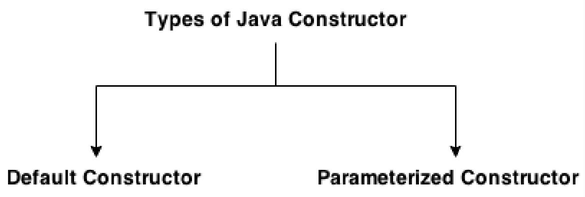
Object Reference

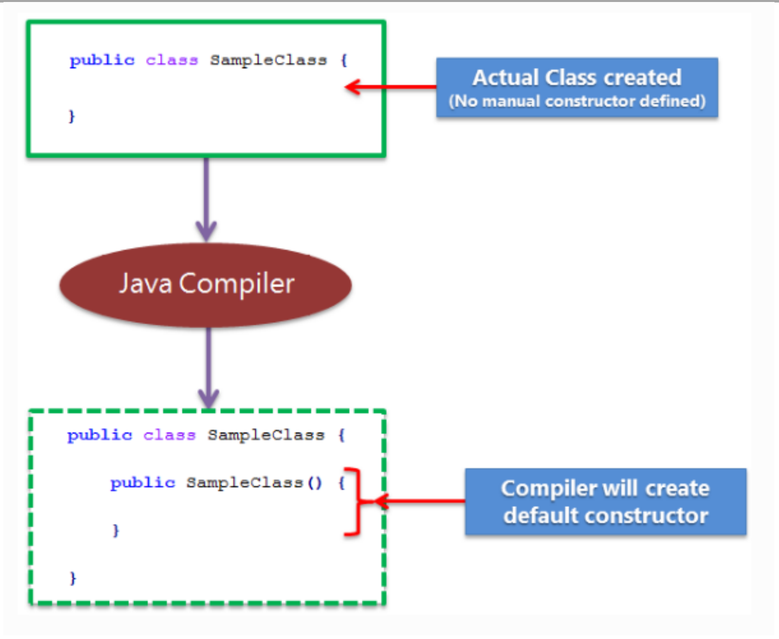
```
Rectangle r1 = new Rectangle();
```



```
Rectangle r2 = r1;
```

Default Constructor

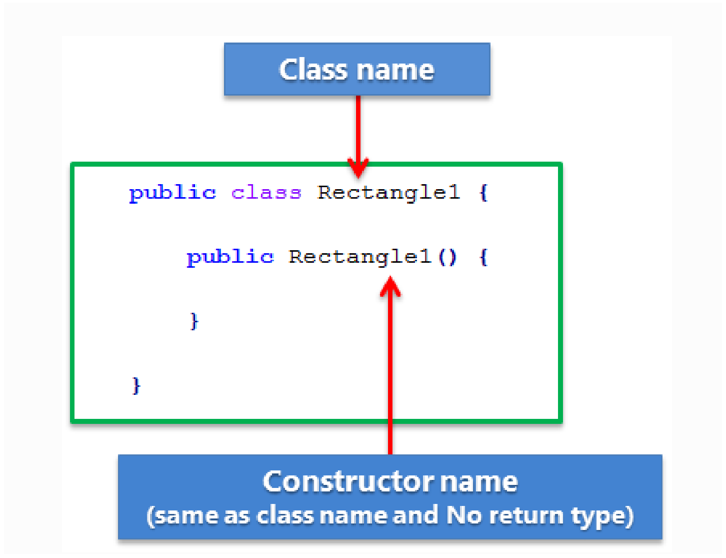



Actual Class created
(No manual constructor defined)
Java Compiler
Compiler will create default constructor

```
public class SampleClass {

}
```

```
public class SampleClass {

    public SampleClass() {

    }

}
```

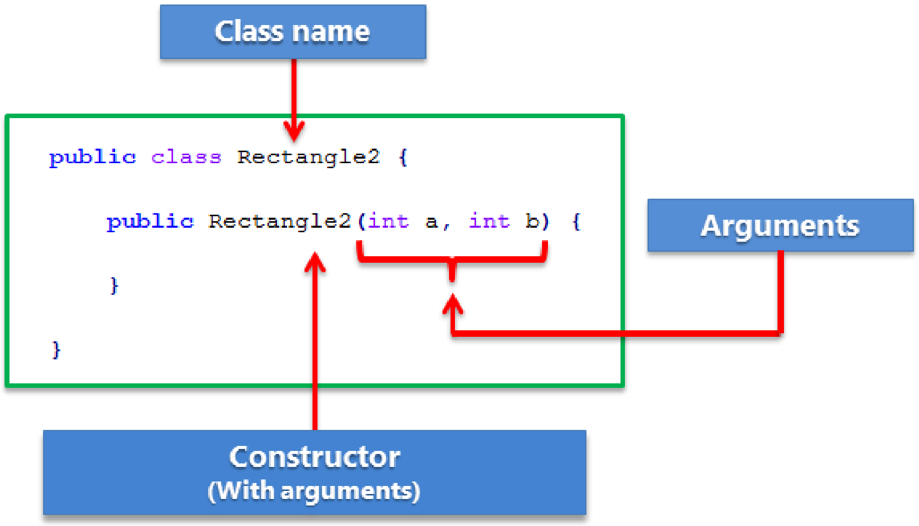
Without Parameter Constructor



```java
public class Rectangle1 {

    public Rectangle1() {

    }

}
```

```java
Rectangle1 obj=new Rectangle1();
```

Parameterize Constructor



```
public class Rectangle2 {

    public Rectangle2(int a, int b) {

    }

}
```

```
Rectangle2 obj = new Rectangle2(120, 80);
```

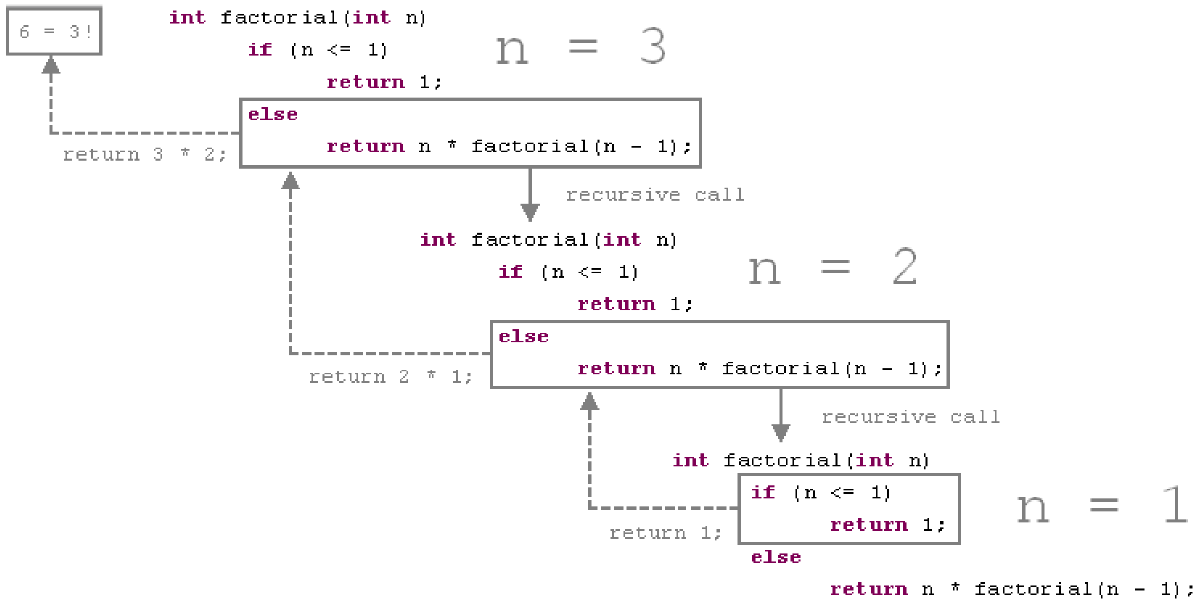
Constructor Overloading

```java
class Test
{
  int x,y;
  int i,k;

  Test(int x, int y)
  {
    System.out.println("entered the 2-param
    constructor");
    this.x = x;
    this.y = y;
  }

  Test(int x, int y, int i, int k)
  {
    this(x,y);// Must be in first line
    System.out.println("called the this
    constructor to avoid redundant code");
    this.i = i;
    this.k = k;
    System.out.println("added 2 assignments");
    System.out.println(x + y + i + k);
  }
}

class ThisConstructorDemo
{
   public static void main(String[] args)
   {
      Test myTest = new Test(1,2,3,4);
   }
}
```

Method Overloading

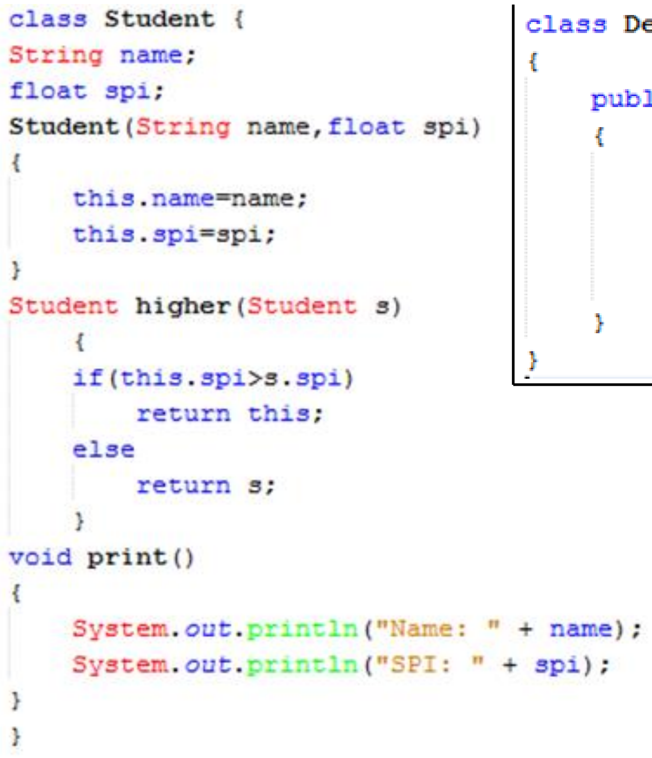
```
class Program
{
    public static int square(int num)
    {
        return num * num;
    }
    public static long square(long num)
    {
        return num * num;
    }
    public static double square(double num)
    {
        return num * num;
    }

}
```

Recursion

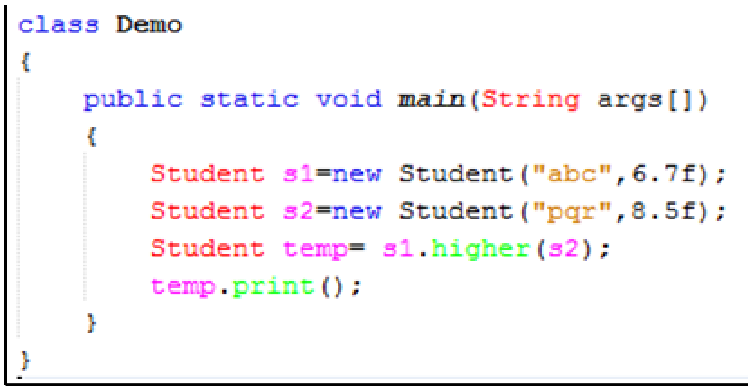


```
int factorial(int n)
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
```

Passing and Returning object from Method

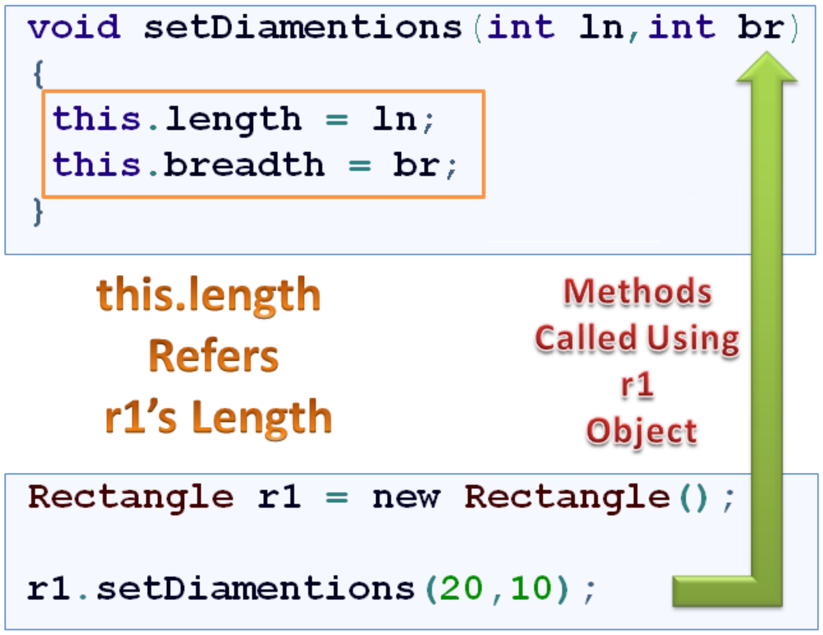


```
class Student {
String name;
float spi;
Student(String name,float spi)
{
    this.name=name;
    this.spi=spi;
}
Student higher(Student s)
    {
    if(this.spi>s.spi)
        return this;
    else
        return s;
    }
void print()
{
    System.out.println("Name: " + name);
    System.out.println("SPI: " + spi);
}
}
```



```
class Demo
{
    public static void main(String args[])
    {
        Student s1=new Student("abc",6.7f);
        Student s2=new Student("pqr",8.5f);
        Student temp= s1.higher(s2);
        temp.print();
    }
}
```

this Keyword



```
void setDiamentions(int ln,int br)
{
 this.length = ln;
 this.breadth = br;
}
```

```
Rectangle r1 = new Rectangle();

r1.setDiamentions(20,10);
```

static data member

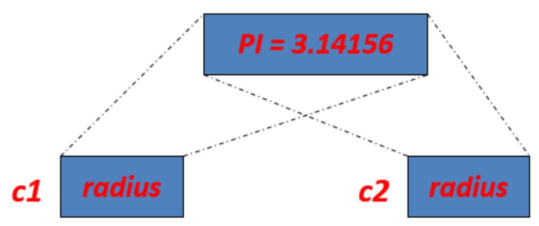
```
class circle
{
static double PI=3.14156;
double radius;
double area()
{
return PI * radius * radius;
}
double perimeter()
{
return 2*PI*radius;
}
} // End of circle class
```

Static Member

Non-static instance field

```
circle c1 = new circle();
circle c2 = new circle();
```

Memory Map

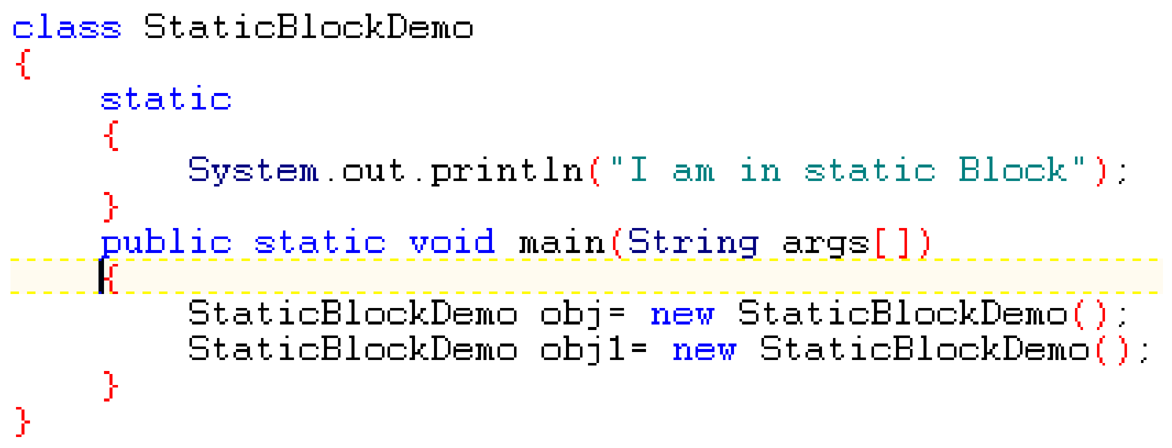


static method

```
class BOX
{
private double l,b,h;    // Instance Fields
BOX(double a,double b,double c)
{
    l=a;this.b=b;h=c;
}       // Constructor
boolean isEqual(BOX other)
{
if (this.l == other.l &&
        this.b == other.b && this.h == other.h)
return true;
else
return false;
}
static boolean isEqual(BOX b1, BOX b2)
{
if (b1.l == b2.l && b1.b == b2.b && b1.h == b2.h)
return true;
else
return false;
}
} // End of BOX class
```

```
class statictest
{
public static void main(String args[])
{
BOX b1 = new BOX(10,6,8);
BOX b2 = new BOX(10,6,8);
BOX b3 = new BOX(1,16,18);
BOX b4 = new BOX(2,6,8);

System.out.println(b1.isEqual(b2));
System.out.println(BOX.isEqual(b1,b2));
System.out.println(b3.isEqual(b1,b2));
System.out.println(b4.isEqual(b2));
System.out.println(b4.isEqual(b4,b2));
}
}
```

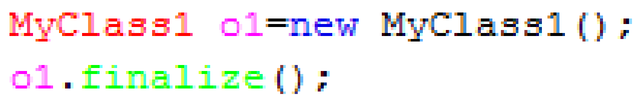
static Block



```
class StaticBlockDemo
{
    static
    {
        System.out.println("I am in static Block");
    }
    public static void main(String args[])
    {
        StaticBlockDemo obj= new StaticBlockDemo();
        StaticBlockDemo obj1= new StaticBlockDemo();
    }
}
```

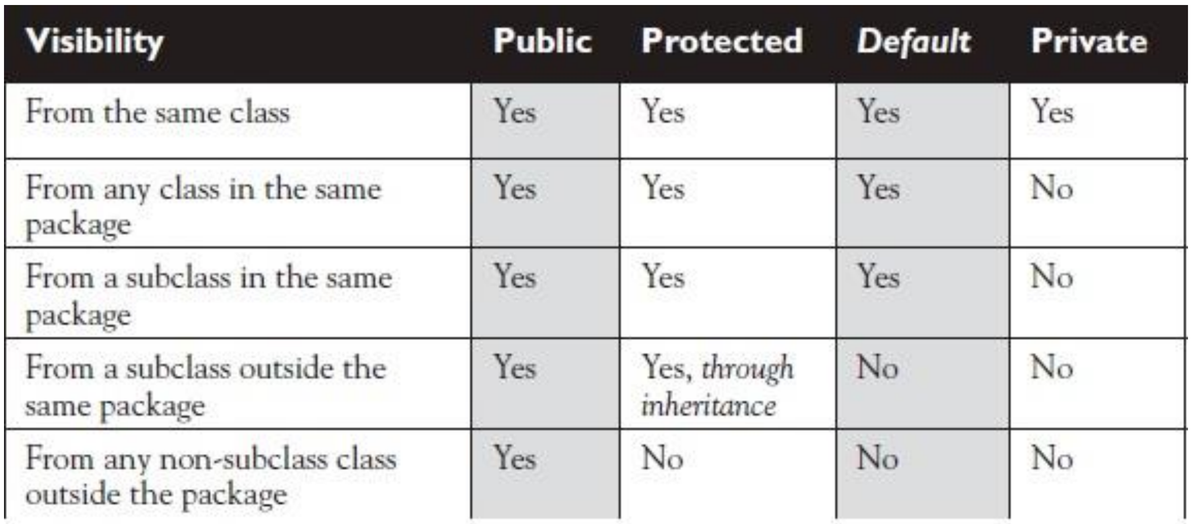
finalize() method

```
public class MyClass1 {
   @Override
   protected void finalize()
      {
           //....
      }

}
```



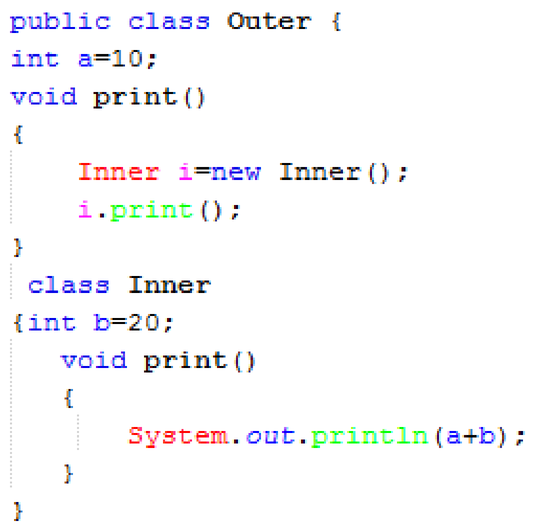
```
MyClass1 o1=new MyClass1();
o1.finalize();
```

Access Control Modifier

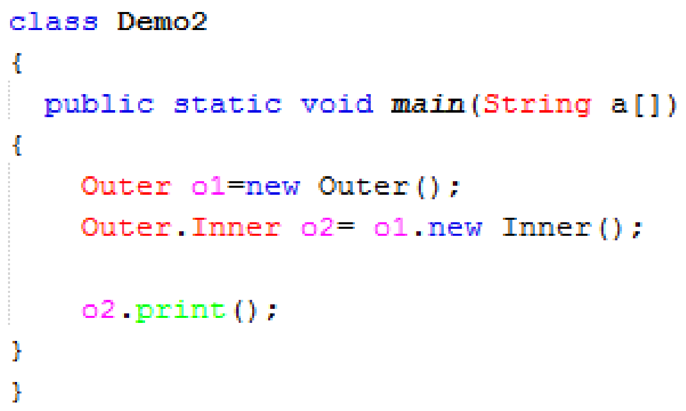


Visibility	Public	Protected	*Default*	Private
From the same class	Yes	Yes	Yes	Yes
From any class in the same package	Yes	Yes	Yes	No
From a subclass in the same package	Yes	Yes	Yes	No
From a subclass outside the same package	Yes	Yes, *through inheritance*	No	No
From any non-subclass class outside the package	Yes	No	No	No

Inner class (Nested Class)

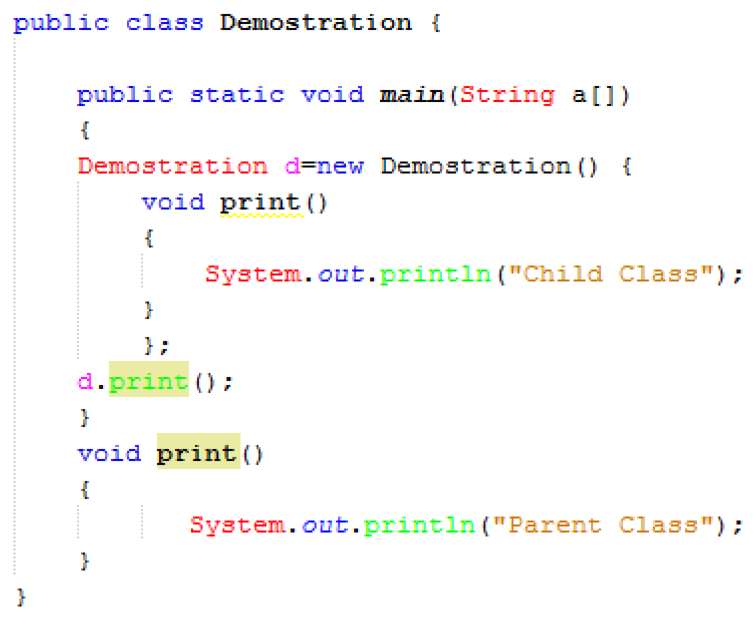


```
public class Outer {
int a=10;
void print()
{
    Inner i=new Inner();
    i.print();
}
 class Inner
{int b=20;
   void print()
   {
       System.out.println(a+b);
   }
}
```



```
class Demo2
{
  public static void main(String a[])
{
    Outer o1=new Outer();
    Outer.Inner o2= o1.new Inner();

    o2.print();
}
}
```

Anonymous inner class



```
public class Demostration {

    public static void main(String a[])
    {
    Demostration d=new Demostration() {
        void print()
        {
            System.out.println("Child Class");
        }
        };
    d.print();
    }
    void print()
    {
            System.out.println("Parent Class");
    }
}
```

References:

- http://programcall.com/8/csnet/oops-features-in-brief.aspx
- http://www.programmersnight.com/class-in-java/
- http://www.cpp-home.com/archives/206.html
- http://www.c4learn.com/java/java-assigning-object-reference/
- http://www.javatpoint.com/constructor
- http://www.sree9it.com/Java/constructors

Questions/Comments