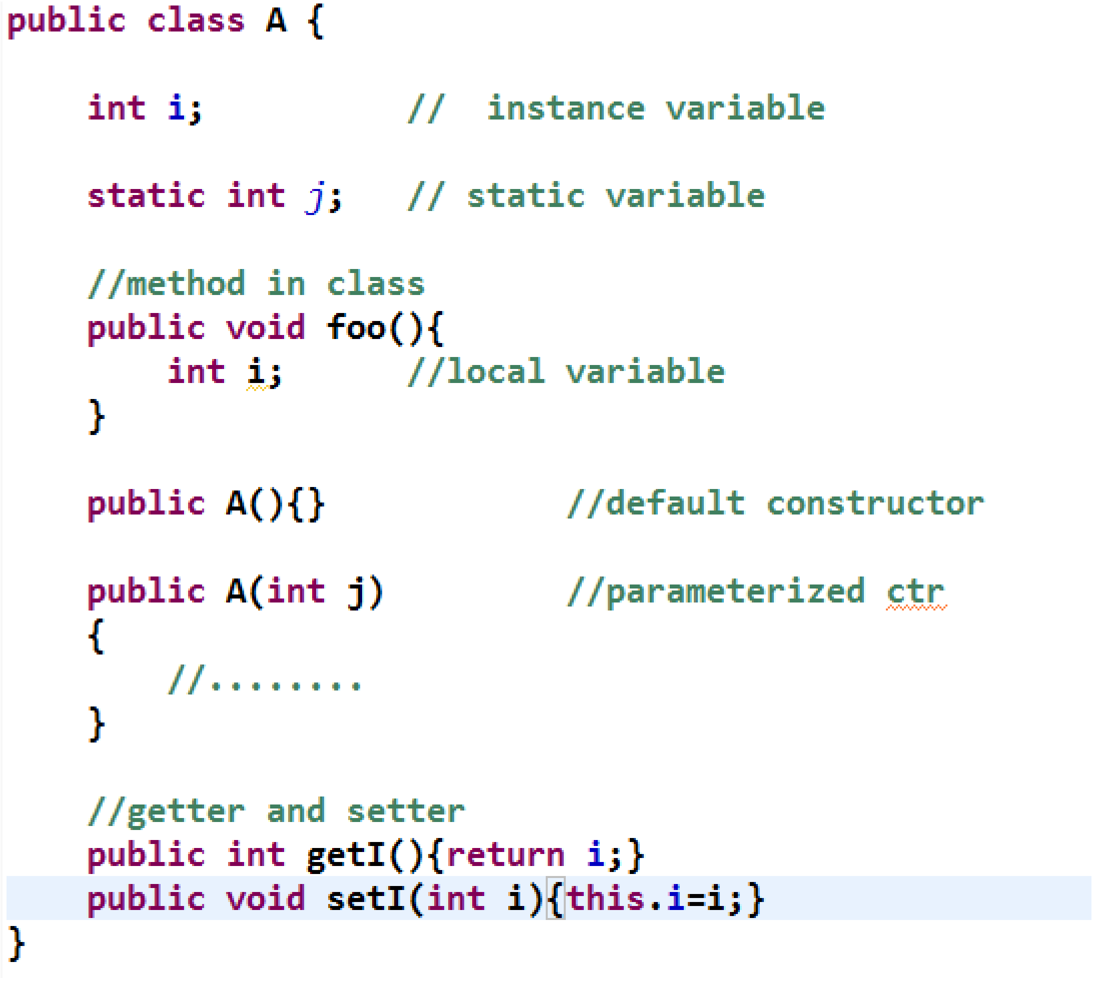
DAY -2

rgupta.mtech@gmail.com

Day 2: Object Orientation & JVM introduction

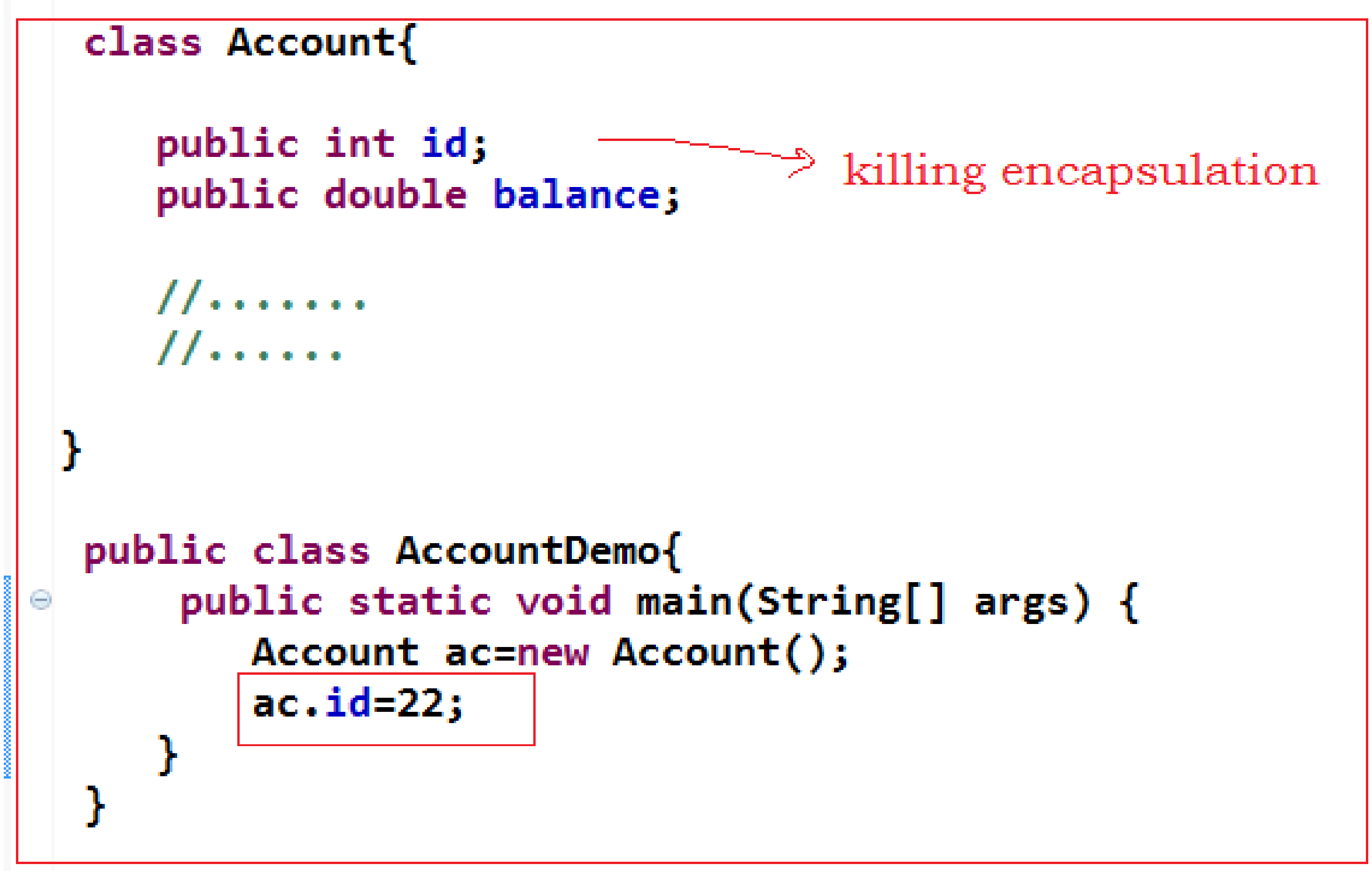
- Creating Class, Object, constructor, init paramaters
- Static variable and method
- Concepts of packages, Access specifier
- Inheritance, Types of inheritance in Java, Inheriting Data Member and Methods
- Role of Constructors in inheritance,Overriding super Class methods, super
- Hands On & Lab

What can goes inside an class?



```
public class A {

    int i;          //  instance variable

    static int j;   // static variable

    //method in class
    public void foo(){
        int i;      //local variable
    }

    public A(){}            //default constructor

    public A(int j)         //parameterized ctr
    {
        //........
    }

    //getter and setter
    public int getI(){return i;}
    public void setI(int i){this.i=i;}
}
```

Creating Classes and object

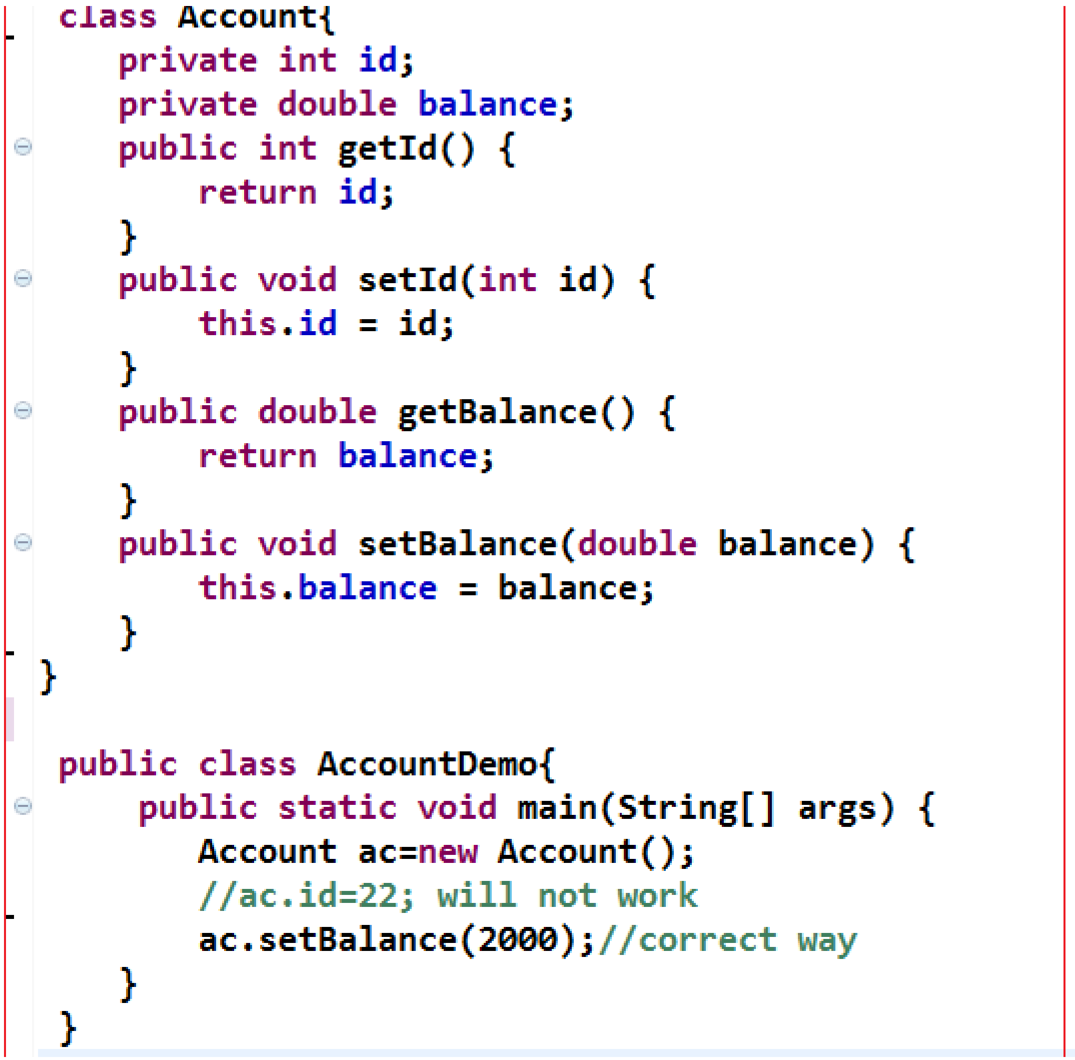


```
 class Account{

    public int id;
    public double balance;

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

}

 public class AccountDemo{
     public static void main(String[] args) {
        Account ac=new Account();
        ac.id=22;
    }
 }
```

killing encapsulation

rgupta.mtech@gmail.com

Correct way?

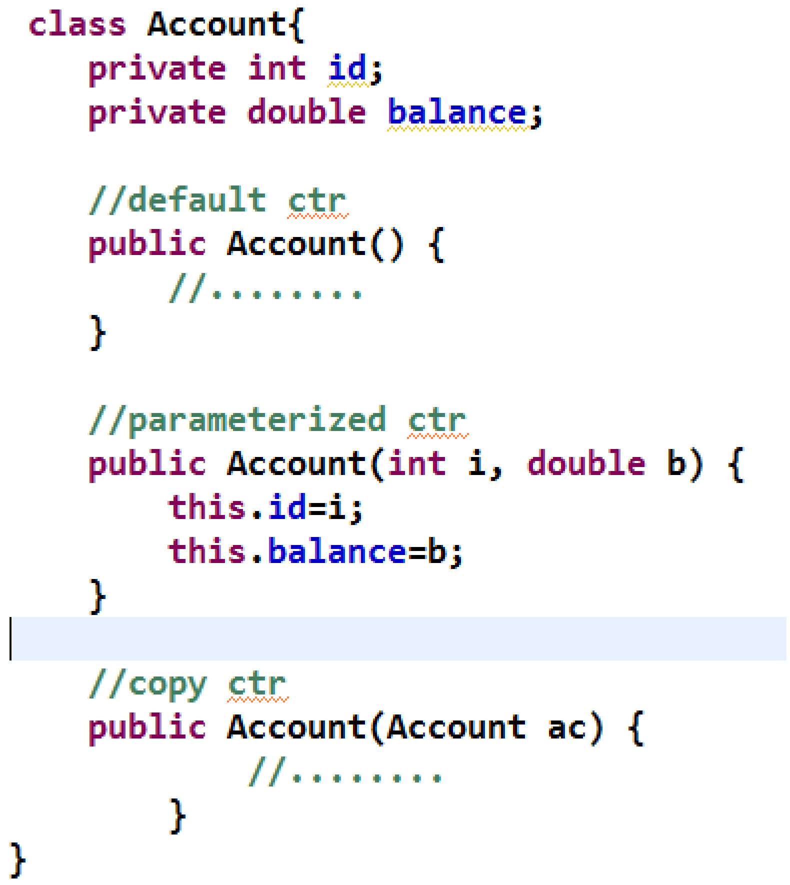


```
 class Account{
    private int id;
    private double balance;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public double getBalance() {
        return balance;
    }
    public void setBalance(double balance) {
        this.balance = balance;
    }
}

 public class AccountDemo{
     public static void main(String[] args) {
        Account ac=new Account();
        //ac.id=22; will not work
        ac.setBalance(2000);//correct way
    }
 }
```

rgupta.mtech@gmail.com

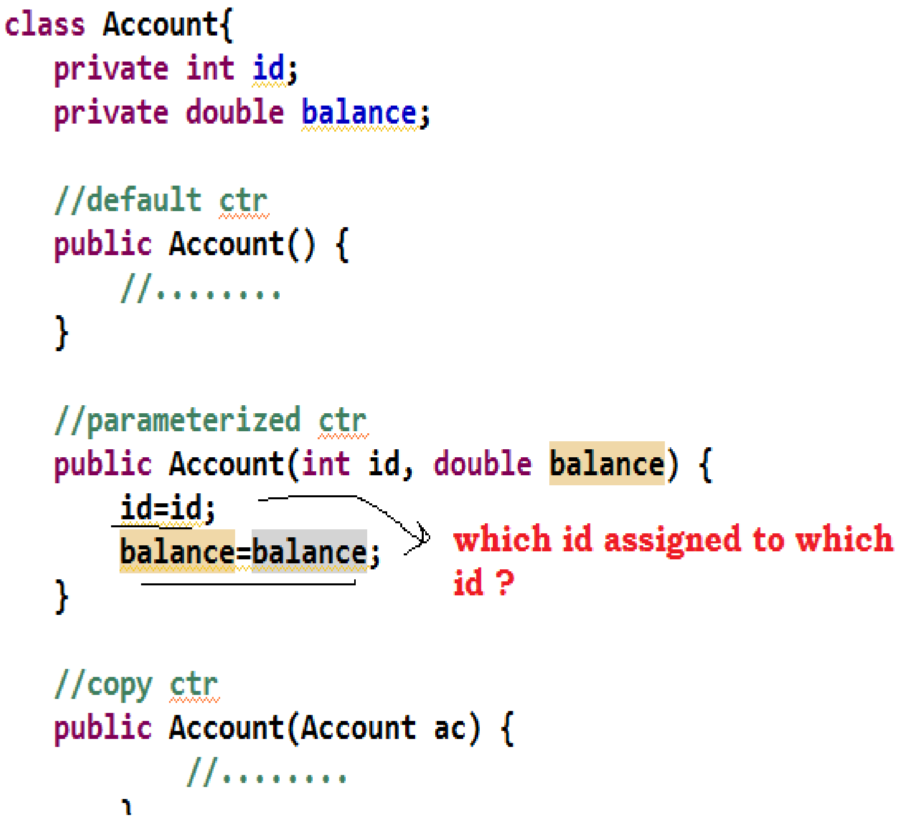
Constructors: default, parameterized and copy

- Initialize state of the object
- Special method have same name as that of class
- Can't return anything
- Can only be called once for a object
- Can be private
- Can't be static*
- Can overloaded but can't overridden*
- Three type of constructors
  - Default, Parameterized and Copy constructor



```
 class Account{
    private int id;
    private double balance;

    //default ctr
    public Account() {
        //.........
    }

    //parameterized ctr
    public Account(int i, double b) {
        this.id=i;
        this.balance=b;
    }

    //copy ctr
    public Account(Account ac) {
            //.........
        }
}
```

Need of “this” ?



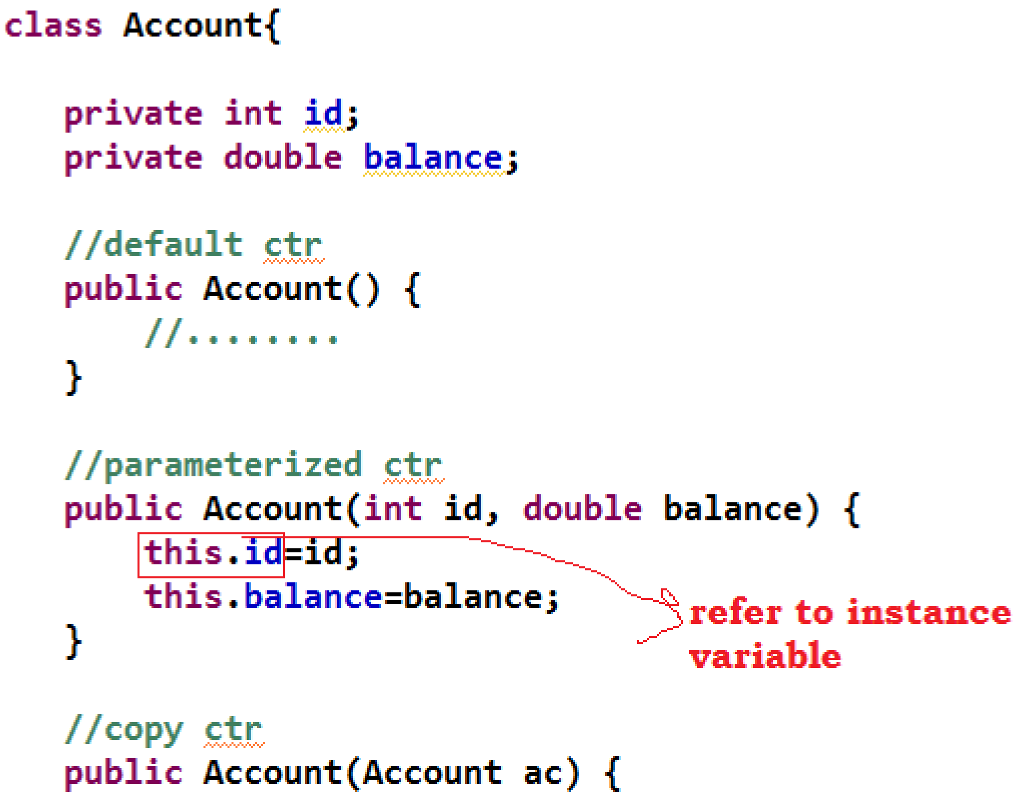
```
class Account{
   private int id;
   private double balance;

   //default ctr
   public Account() {
       //........
   }

   //parameterized ctr
   public Account(int id, double balance) {
       id=id;
       balance=balance;
   }

   //copy ctr
   public Account(Account ac) {
           //........
       }
```

which id assigned to which id ?

- Which id assigned to which id?
- “this” is an reference to the current object required to differentiate local variables with instance variables

Refer next slide…

“this” used to resolve confusion…



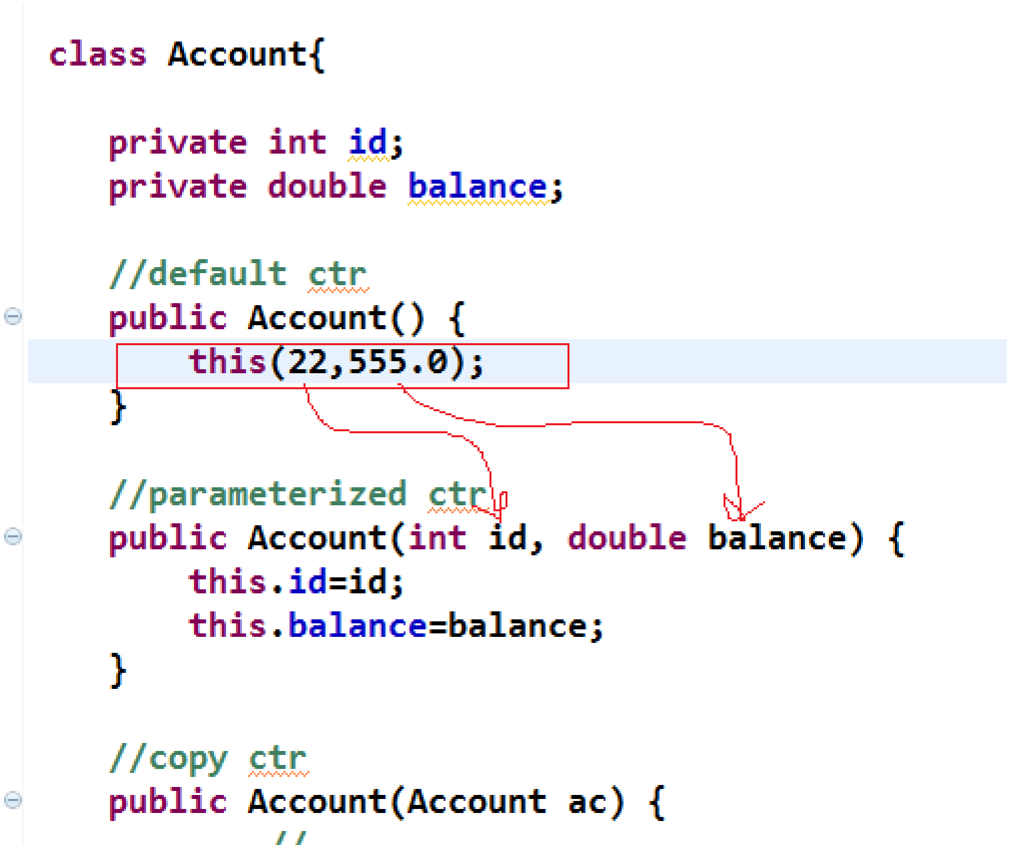
```
class Account{

    private int id;
    private double balance;

    //default ctr
    public Account() {
        //........
    }

    //parameterized ctr
    public Account(int id, double balance) {
        this.id=id;
        this.balance=balance;
    }

    //copy ctr
    public Account(Account ac) {
```

refer to instance variable

rgupta.mtech@gmail.com

this : Constructor chaining?

- Calling one constructor from another ?



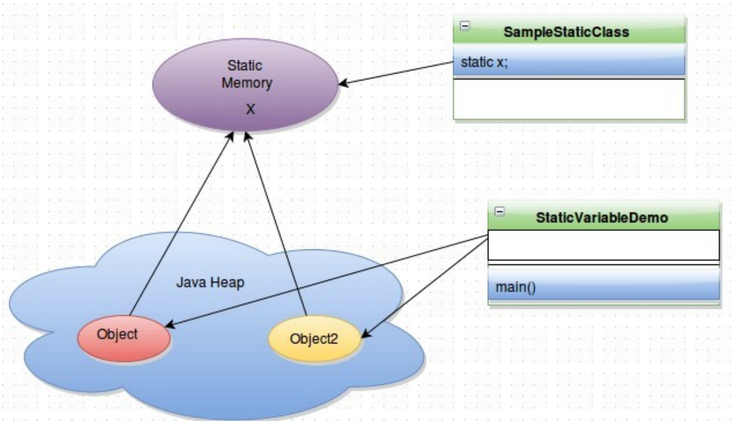
```java
class Account{

    private int id;
    private double balance;

    //default ctr
    public Account() {
        this(22,555.0);
    }

    //parameterized ctr
    public Account(int id, double balance) {
        this.id=id;
        this.balance=balance;
    }

    //copy ctr
    public Account(Account ac) {
        //
```

rgupta.mtech@gmail.com

Static method/variable

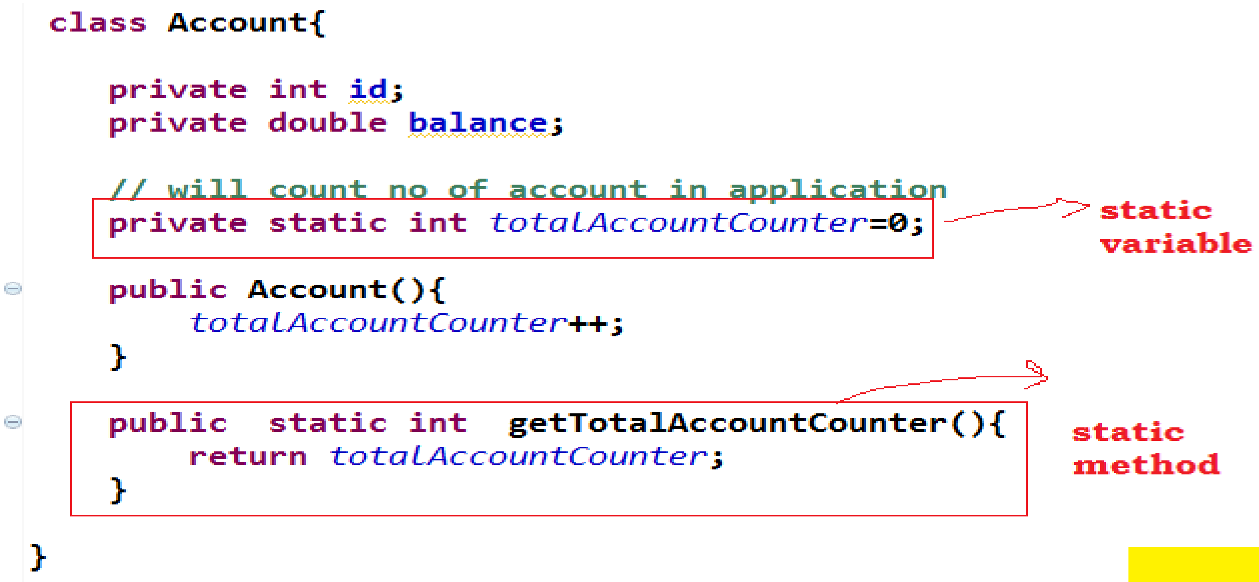
- Instance variable –per object while static variable are per class
- Initialize and define before any objects
- Most suitable for counter for object
- Static method can only access static data of the class
- For calling static method we don’t need an object of that class

How to count number of account object in the memory?



Now guess why main was static?

Using static data..

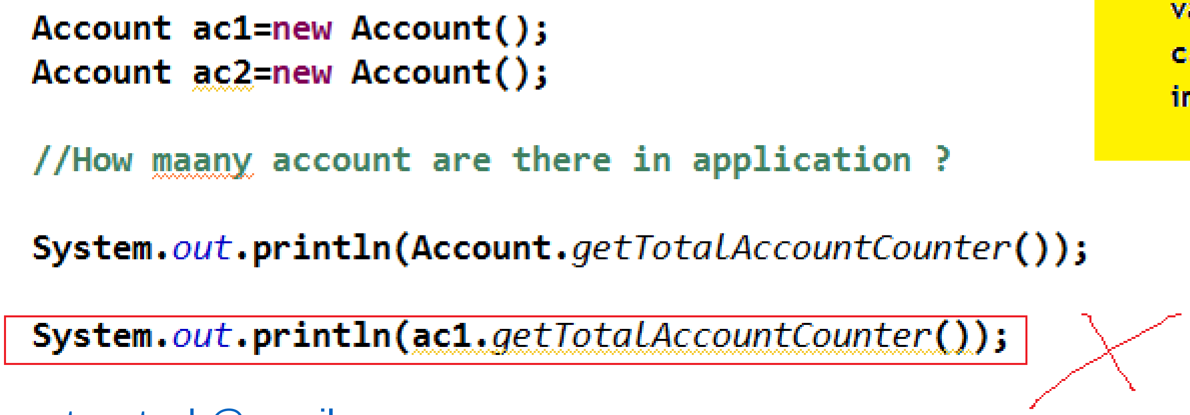


```
class Account{

    private int id;
    private double balance;

    // will count no of account in application
    private static int totalAccountCounter=0;

    public Account(){
        totalAccountCounter++;
    }

    public  static int  getTotalAccountCounter(){
        return totalAccountCounter;
    }

}
```

static variable

static method

We can not access instance variable in static method but can access static variable in instance method



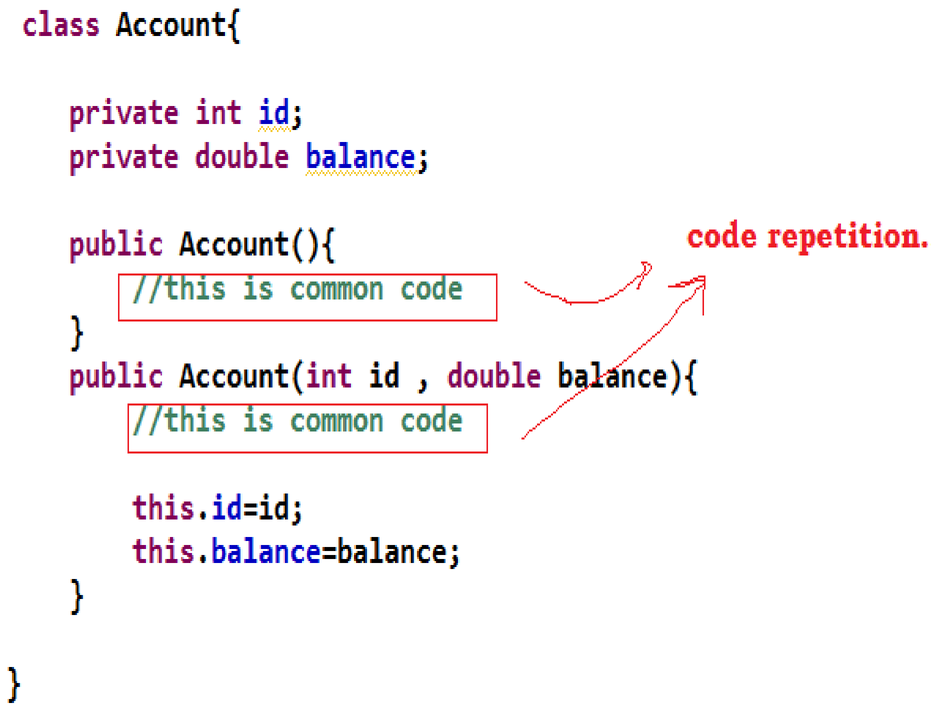
```
Account ac1=new Account();
Account ac2=new Account();

//How maany account are there in application ?

System.out.println(Account.getTotalAccountCounter());

System.out.println(ac1.getTotalAccountCounter());
```

rgupta.mtech@gmail.com

Initialization block

- We can put repeated constructor code in an Initialization block…

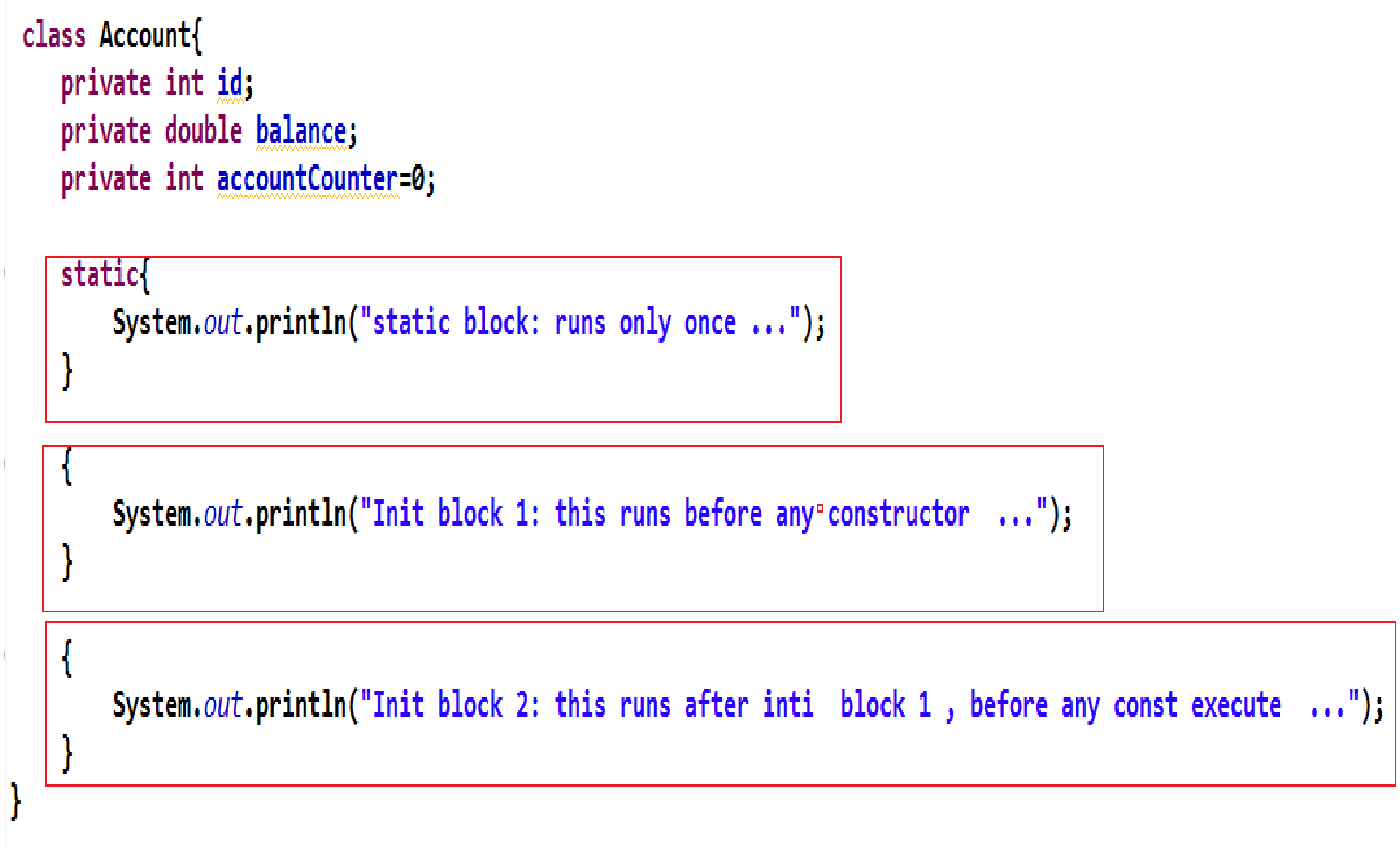
- Static Initialization block runs before any constructor and runs only once…



```
class Account{

    private int id;
    private double balance;

    public Account(){
        //this is common code
    }
    public Account(int id , double balance){
        //this is common code

        this.id=id;
        this.balance=balance;
    }

}
```

code repetition.

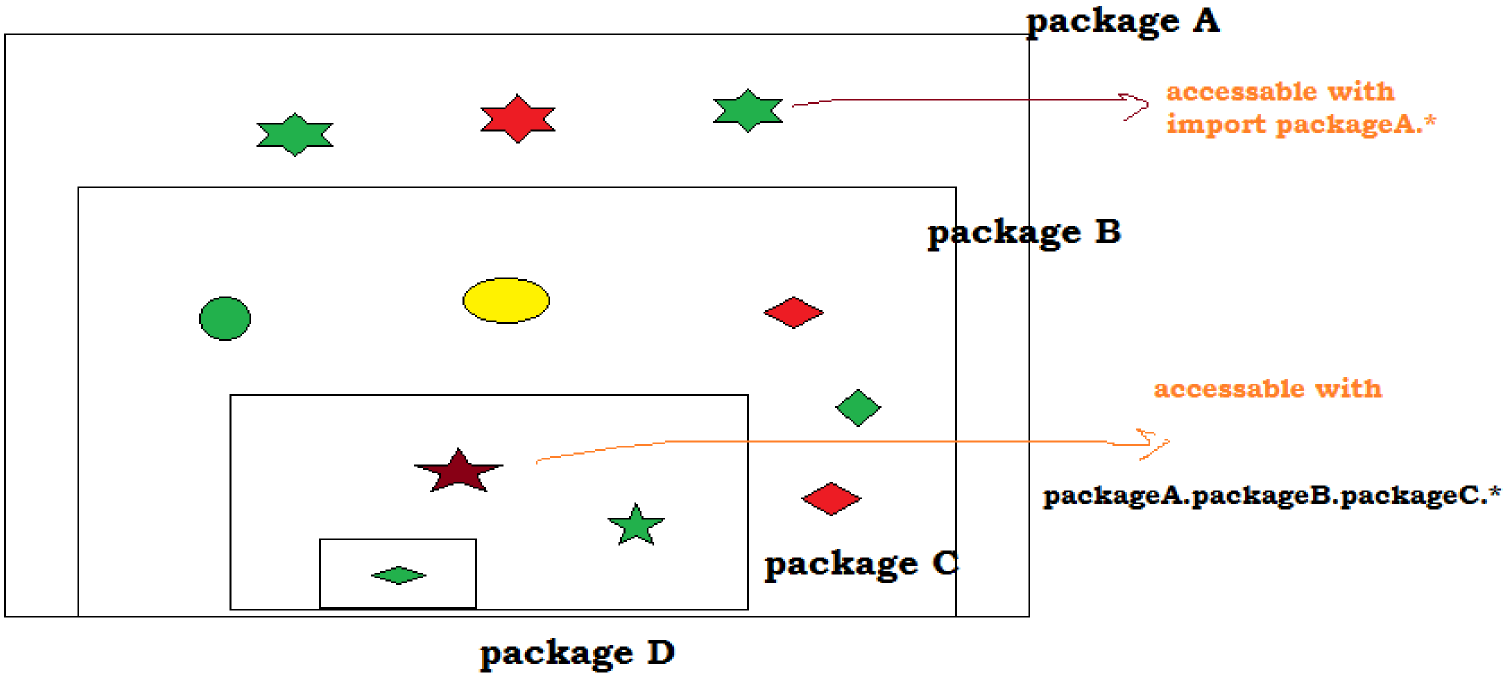
Initialization block



```
class Account{
   private int id;
   private double balance;
   private int accountCounter=0;

   static{
       System.out.println("static block: runs only once ...");
   }

   {
       System.out.println("Init block 1: this runs before any constructor  ...");
   }

   {
       System.out.println("Init block 2: this runs after inti  block 1 , before any const execute  ...");
   }
}
```

Packages

- Packages are Java’s way of grouping a number of related classes and/or interfaces together into a single unit.
- Packages act as “containers” for classes.



rgupta.mtech@gmail.com

Java Foundation Packages

- Java provides a large number of classes groped into different packages based on their functionality.
- The six foundation Java packages are:

java.lang

- Contains classes for primitive types, strings, math functions, threads, and exception

java.util

- Contains classes such as vectors, hash tables, date etc.

java.io

- Stream classes for I/O

java.awt

- Classes for implementing GUI – windows, buttons, menus etc.

java.net

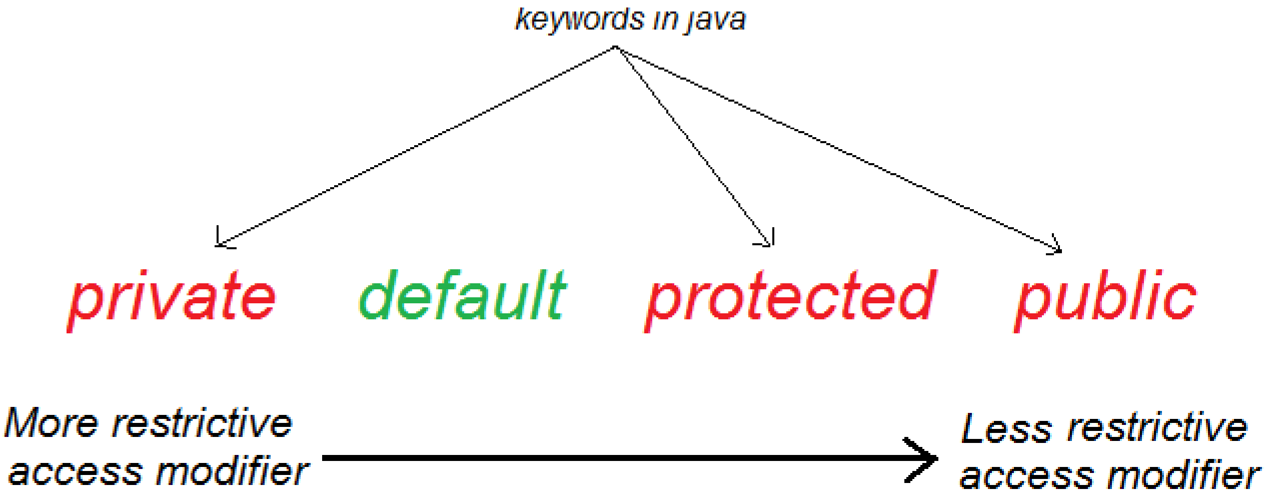
- Classes for networking

java.applet

- Classes for creating and implementing applets

Visibility Modifiers

- **For instance variable and methods**
  - Public Protected
  - Default (package level) Private
- **For classes**
  - Public and default



keywords in java

private default protected public

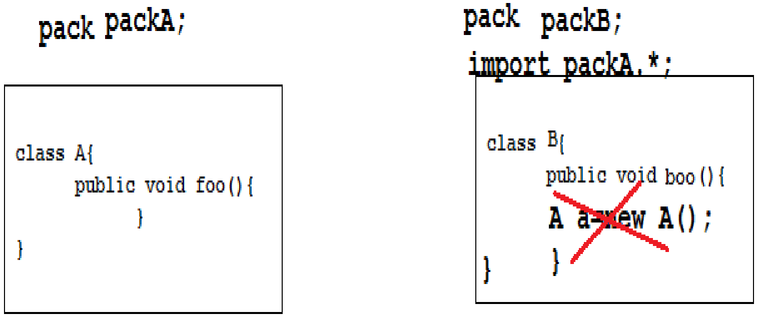
More restrictive access modifier

Less restrictive access modifier

Visibility Modifiers

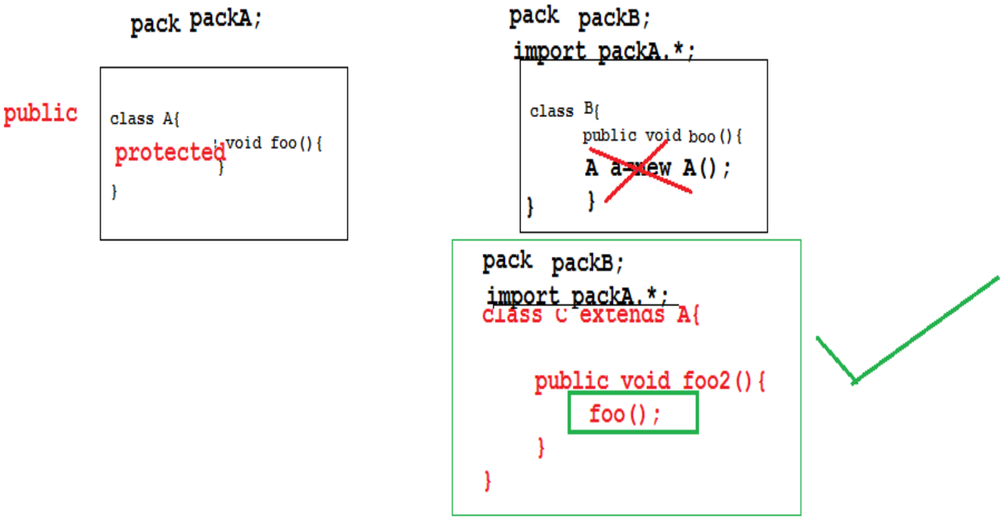
- class A has default visibility
  - hence can access in the same package only.
- Make class A public, then access it.

- Protected data can access in the same package and all the subclasses in other packages provide class itsef is public



```
pack packA;

class A{
        public void foo(){
            }
}
```

```
pack packB;
import packA.*;

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



```
pack packA;

public
class A{
protected void foo(){
          }
}
```

```
pack packB;
import packA.*;

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

```
pack packB;
import packA.*;
class C extends A{


        public void foo2(){
            foo();
        }
}
```

rgupta.mtech@gmail.com

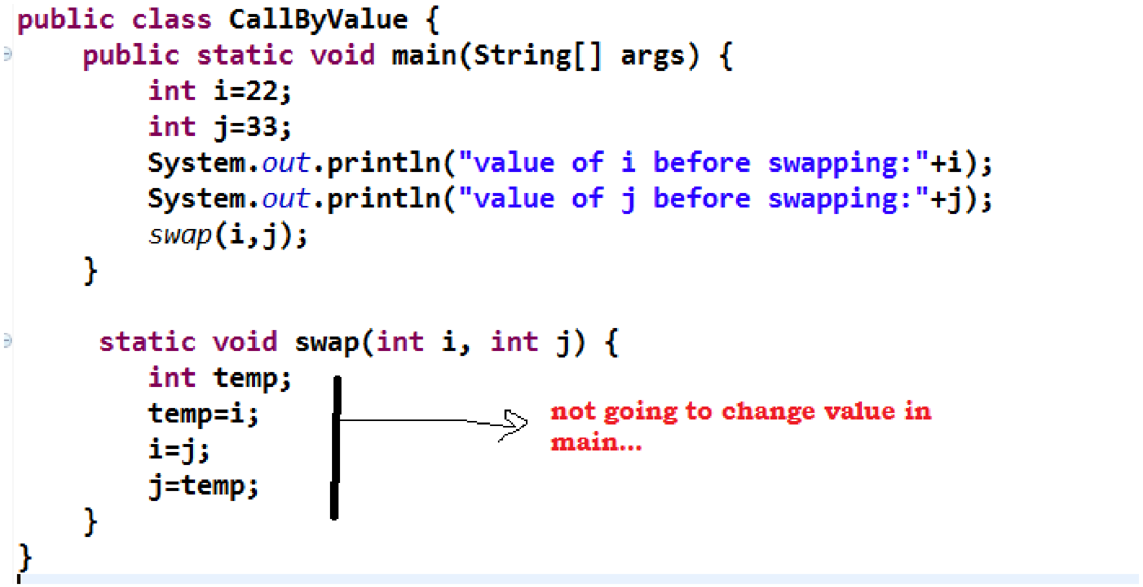
Want to accept parameter from user?

java.util.Scanner (Java 1.5)

```
Scanner stdin = Scanner.create(System.in);

int n = stdin.nextInt();
String s = stdin.next();


boolean b = stdin.hasNextInt()
```

Call by value

- Java support call by value
- The value changes in function is not going to reflected in the main.

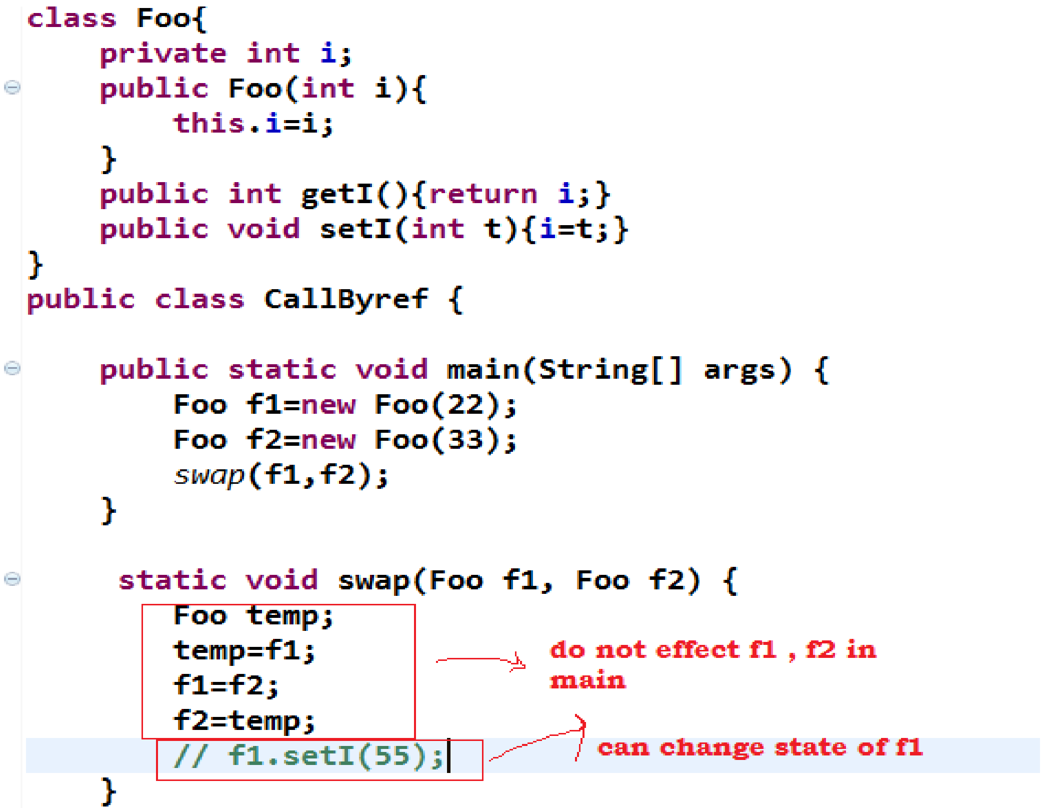


```
public class CallByValue {
    public static void main(String[] args) {
        int i=22;
        int j=33;
        System.out.println("value of i before swapping:"+i);
        System.out.println("value of j before swapping:"+j);
        swap(i,j);
    }

     static void swap(int i, int j) {
        int temp;
        temp=i;
        i=j;
        j=temp;
    }
}
```

not going to change value in main...

Call by reference

- Java don’t support call by reference.
- When you pass an object in an method copy of reference is passed so that we can mutate the state of the object but can’t delete original object itself



```
class Foo{
    private int i;
    public Foo(int i){
        this.i=i;
    }
    public int getI(){return i;}
    public void setI(int t){i=t;}
}
public class CallByref {

    public static void main(String[] args) {
        Foo f1=new Foo(22);
        Foo f2=new Foo(33);
        swap(f1,f2);
    }

     static void swap(Foo f1, Foo f2) {
        Foo temp;
        temp=f1;
        f1=f2;
        f2=temp;
        // f1.setI(55);
    }
```

do not effect f1 , f2 in main

can change state of f1