Java

Java is Everywhere

- JAVA resides in mobiles, client machines, server machines, embedded devices, smart phones, cloud, etc.
- It shares the same basic features of the language and libraries.
- Principle of Java: Write Once, Run Anywhere(WORA)

What is Library?

- Java Library is a collection of predefined classes.
- You can use these classes either by inheriting them or by instantiating them.

Version History of JAVA

Version	Release Date	No of Classes	No of Packages
JAVA1.0	May-1996	212	8
JAVA1.1	Feb-1997	503	23
JAVA2.0	Dec-1998	1520	59
JAVA5.0	Sep-2004	3562	166
JAVA6.0	Dec-2006	3792	203
JAVA7.0	Jul-2011	4024	209

Features of JAVA

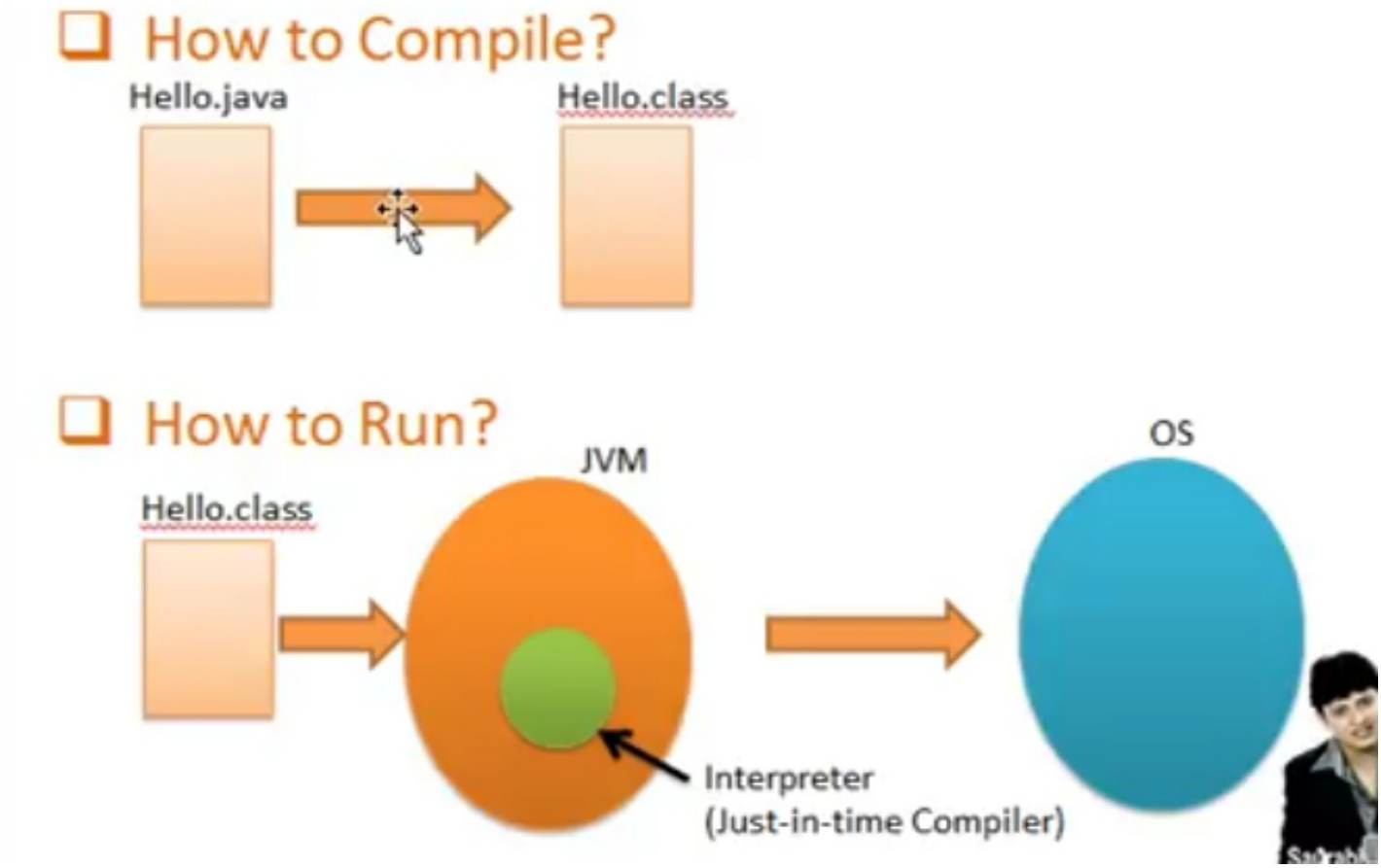
- ❑ Simple
- ❑ Object Oriented Language
- ❑ Distributed
- ❑ Interpreted
- ❑ Robust
- ❑ Secure
- ❑ Portable
- ❑ Multi-threaded
- ❑ Garbage Collector

- Multi threaded means java run two program simultaneously. ex:- typing and speling check

simultaneously.

- Garbage collector deleate a memory which is not in use currently

Compile and Run



- In java there are only one compiler which compile any operating system.
- But .class which also known as byte code which is not understand by operating system.

- If in one java code there is 5 class file then it make 5 .class file.

- Interpreter read the compiler code and tranfer the operating system.

JDK

- **Java Developer Kit** contains tools needed to develop the Java programs
- These tools could be Compiler (javac.exe), Application Launcher (java.exe), etc

- Javac.exe is java compiler
- Java.exe is java launcher

JRE

- **Java Runtime Environment**
- It contains **JVM** (Java **V**irtual **M**achine) and Java Package Classes(Java Library)

JVM

- JVM is platform dependent
- The ***Java Virtual Machine*** provides a platform-independent way of executing code
- Java Virtual Machine interprets the byte code into the machine code depending upon the underlying operating system and hardware combination.

It is truly software

Remember

- Java is a case sensitive language like C and C++
- Java is nearly 100% object oriented language
- In java, it is not possible to make a function which is not a member of any class (as we can do in C++)

First Program

```
public class HelloWorld
{
  public static void main(String [] args)
  {
     System.out.println("Hello World");
  }
} // do not put semicolon here
```

- There are four access pacifier
- Public
- Private
- Protected
- Default
- Inner class may be public ,private,protected.

- System and String is predefined class.
- Static is use for avoid object.
- Void means function not written anything.

- After in build class if (.) is written i.e static member name which may be variable or function.

- Out is static refrence variable which represent object and after out (.) It make print stream class object.

Keywords

abstract	default	goto	package	this
assert	do	if	private	throw
boolean	double	implements	protected	throws
break	enum	import	public	transient
byte	else	instanceof	return	true
case	extends	int	short	try
catch	false	interface	static	void
char	final	long	strictfp	volatile
class	finally	native	super	while
const	float	new	switch	
continue	for	null	synchronized	

Types (Data Types)

- A type identifies a set of values (and their representation in memory) and a set of operations that transform these values into other values of that set.
- Java is strongly typed language

- Java is strongly typed language means there is important is data type(student is also data type)

There are two data types:-

Types

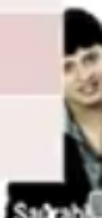
- Primitive Types
- User-defined Types

- Premetive type is inbuild data type ex (int,char,float)

- User define type is created by user

Primitive Type

Primitive Type	Size
boolean	implementation dependent
char	16 bits (stores unicode)
byte	8 bits
short	16 bits
int	32 bits
long	64 bits
float	32 bits
double	64 bits



1 bytes = 8 bits

- Char is 16 bit because store of unicode character size is more
- In byte,short,int,long we store integer value
- Float and double is use for real number i.e which have decimal number (ex 2.0)

Variables

- Examples:
  - int counter;
  - double temp;
  - String name;
  - int[ ] ages;
  - char letters[ ];

Constants

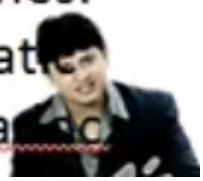
- **integer** constant consists of a sequence of digits
- If the constant is to represent a long integer value, it must be suffixed with an uppercase L or lowercase l.
- If there is no suffix the constant represents a 32 bit integer (an int).

Constants

- **Integer** constant can be specified in the decimal, hexadecimal, octal or binary format
  - 127
  - 0x7f
  - 0177
  - 0b01101100

Comments

- *Block style* comments begin with /* and terminate with */ that spans multiple lines.
- *Line style* comments begin with // and terminate at the end of the line.
- *Documentation style* comments begin with /** and terminate with */ that spans multiple lines. They are generally created using the automat documentation generation tool, such as java



Conversion

- Every expression written in the Java programming language has a type that can be deduced from the structure of the expression and the types of the literals, variables, and methods mentioned in the expression.

Ex 3+4.5
Here 3 is int type value
4.5 is double type value

Widening Conversion

```
int y=3;
float x=y;  //Widening Conversion, no error
```

We write this Because there no data loss

Narrowing Conversion

```
float x=3.4f;
int y=x; //narrowing conversion, error

int y=(int)x; //no error

float k=3.5; //narrowing conversion, error

float k=3.5f; //no error
```

- If we not written f in last of 3.4 then it will declear as double type.

- Error beause degrades i.e data will loss (from higer data to lower data not possible)

Permitted Conversions

- ❑ byte to short, int, long, float, or double
- ❑ short to int, long, float, or double
- ❑ char to int, long, float, or double
- ❑ int to long, float, or double
- ❑ long to float or double
- ❑ float to double

Why Class?

- ❑ Primitive data type
- ❑ Non-primitive data type

- In premetive data type we store single value but
- We need class because we store multiple in that object.
- So we need non premetive data type.

Class

- Class is a description of an object's property and behaviour
- Creating class is as good as creating data type
- Class is defining a category of data

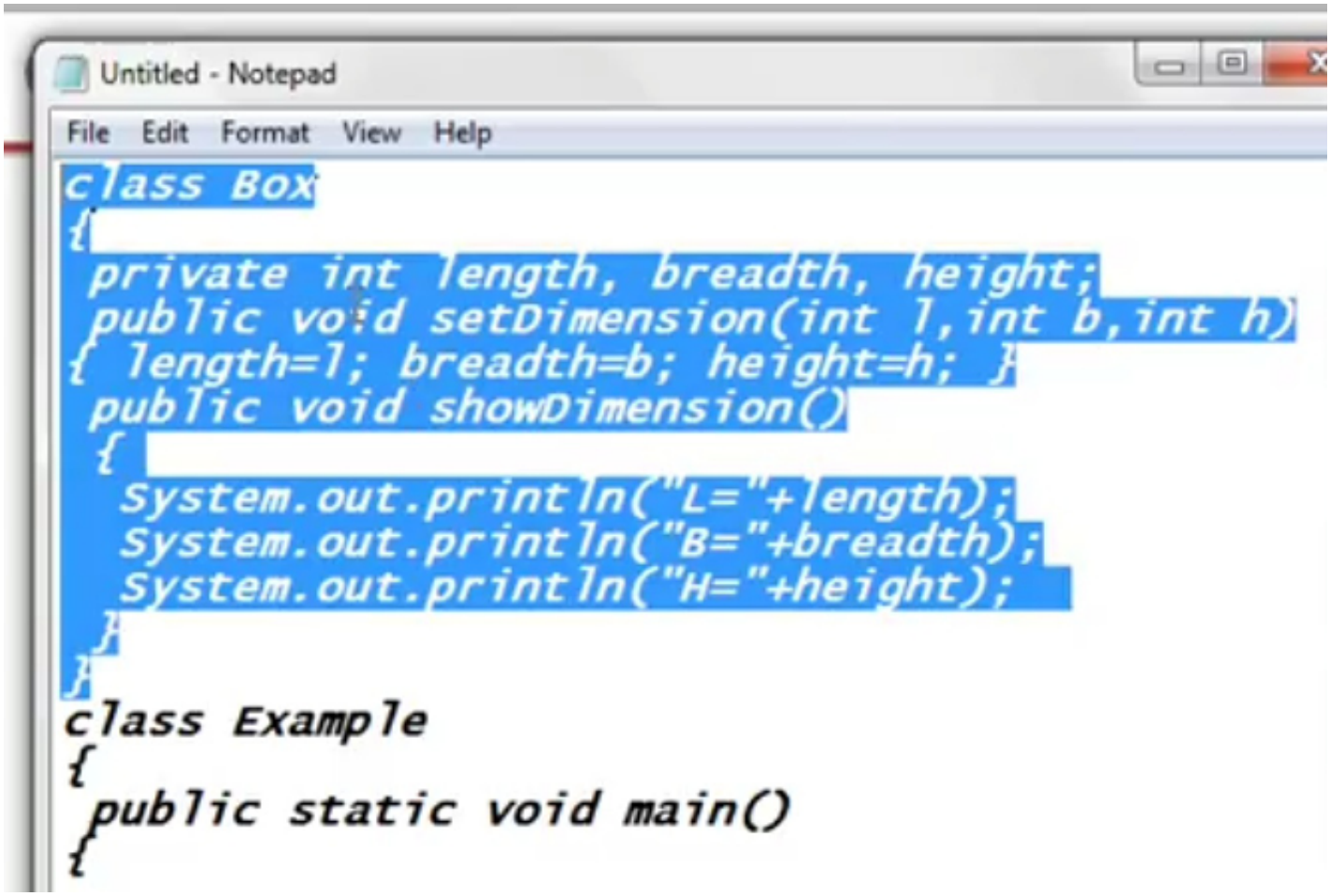
Object

- Object is a real world entity
- Object is an instance of a class
- Object consumes memory to hold property values

- Object is a memory block which we have store data with there related entity.
- //This is a variable but big variable//

Class

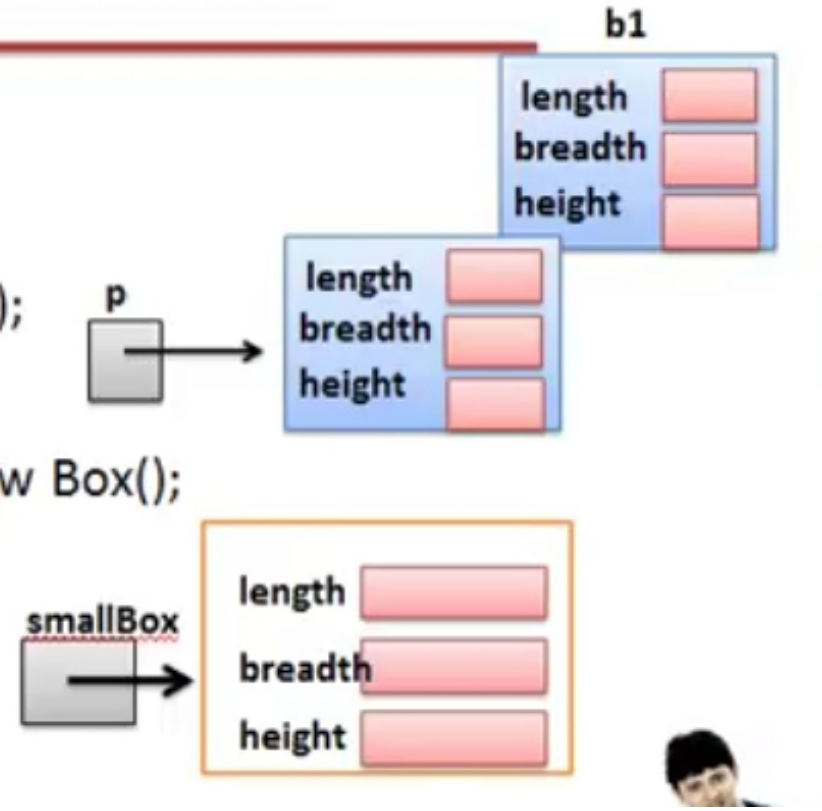
- Define a class Box with length, breadth and height as member variables . Also define setDimension() and showDimension() as member functions.



- Here length , breadth and height is properties.

Creating Objects in Java

- In C++
  - Box b1;
  - Box *p=new Box();
- In Java
  - Box smallBox =new Box();



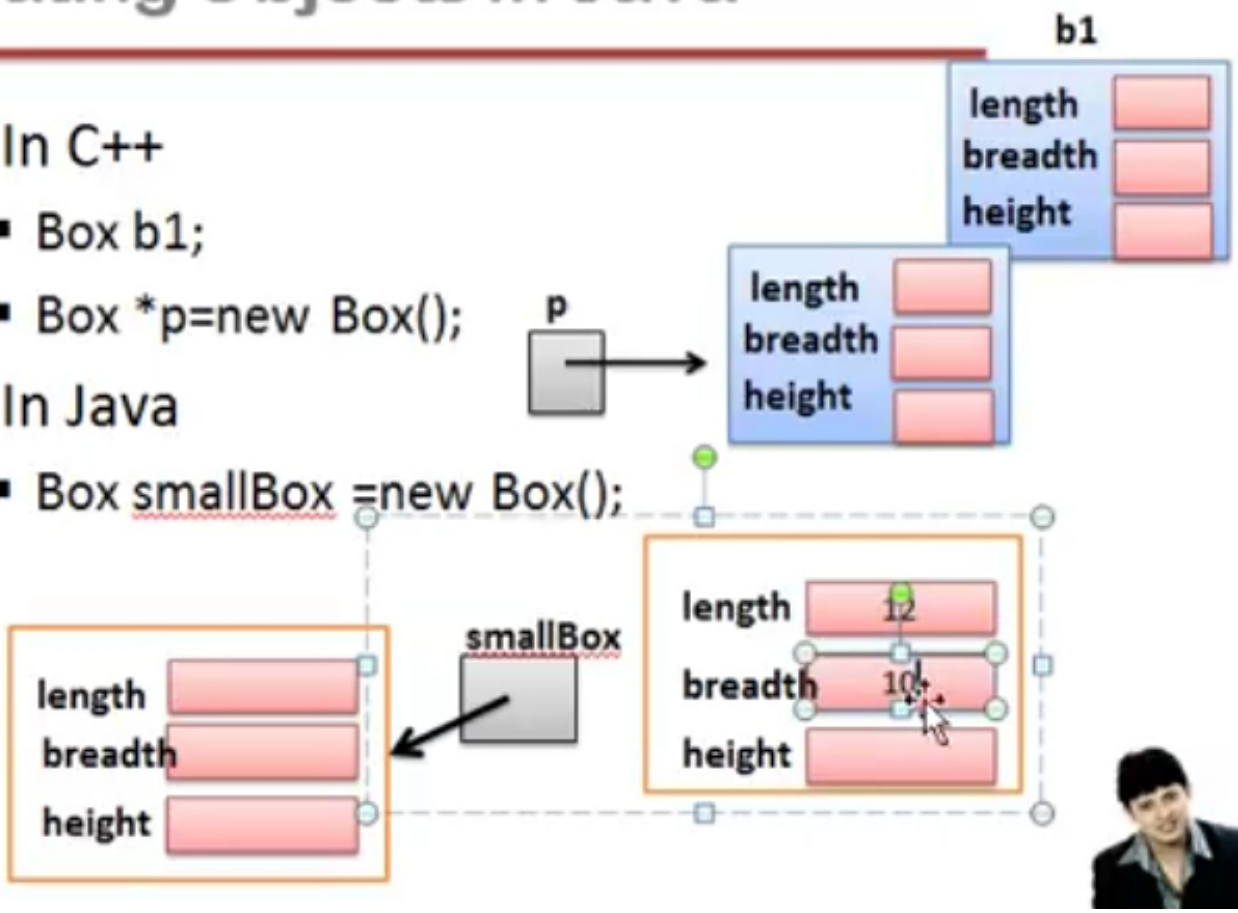
- In java object make dynamic only
- Then we create refrence variable i.e smallBox is a refrence variable.

```
class Box
{
 private int length, breadth, height;
 public void setDimension(int l,int b,int h)
{ length=l; breadth=b; height=h; }
 public void showDimension()
 {
  System.out.println("L="+length);
  System.out.println("B="+breadth);
  System.out.println("H="+height);
 }
}
class Example
{
 public static void main()
{
 Box smallBox=new Box();
 smallBox.setDimension(12,10,5);
 smallBox.showDimension();
 smallBox=new Box();
 smallBox.showDimension();
}
```

- When we write again smallBox=new box(); Then small box point new smallbox And previous object will be garbage value.

Creating Objects in Java

- In C++
  - Box b1;
  - Box *p=new Box();
- In Java
  - Box smallBox =new Box();



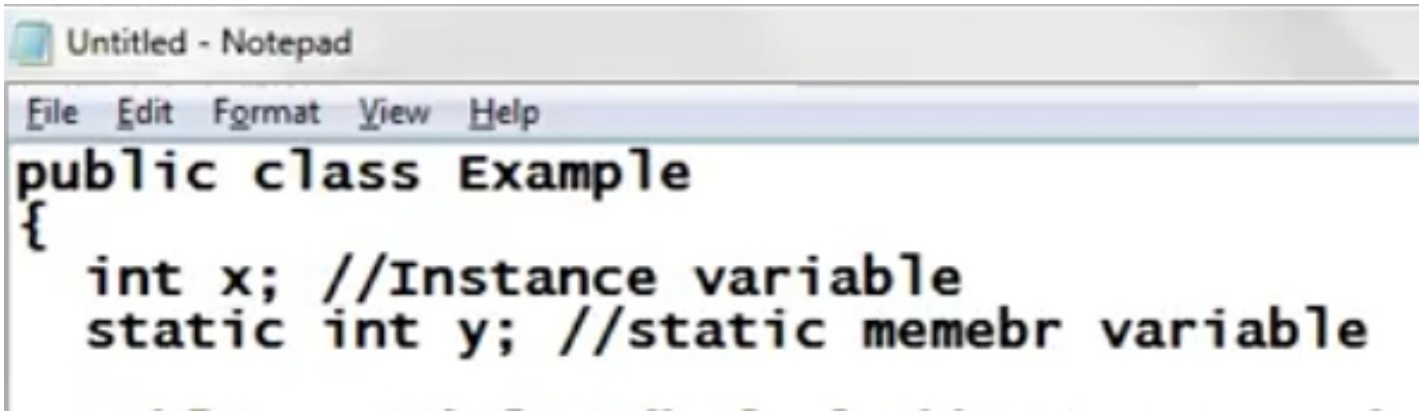
Lecture 7

In Java

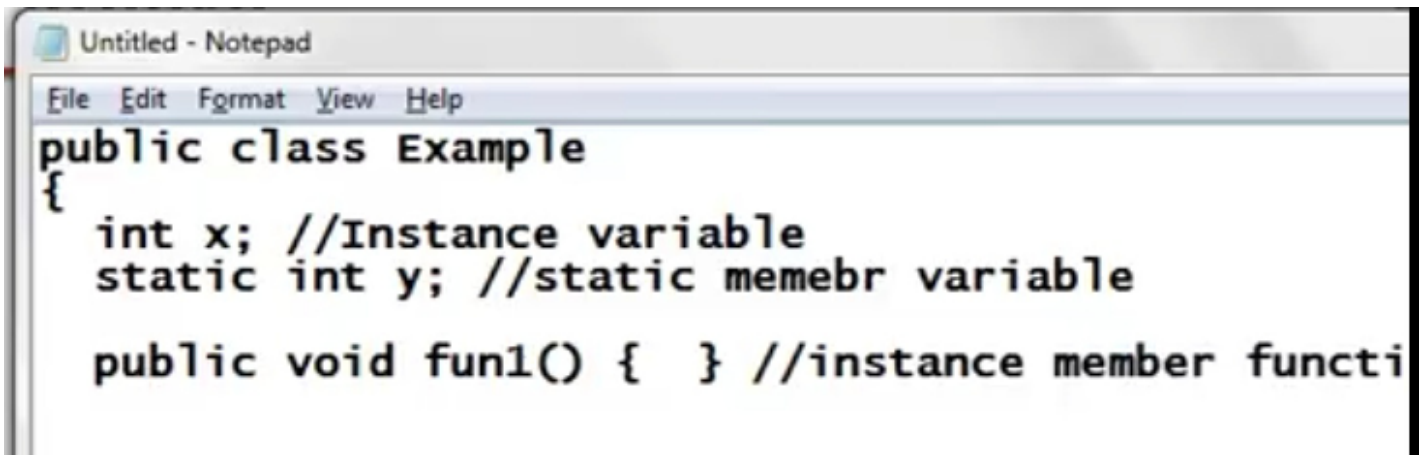
- Static member variable
- Static member function
- ...and not static variable in methods
- ...but we can have static inner class

In Java

- Static member variable



- Static member function



- ...and not static variable in methods

```
public class Example
{
  int x; //Instance variable
  static int y; //static memebr variable

  public void fun1() { static int a; } //instance
  public static void fun2() {  } //static member f
```

- ...but we can have static inner class

Static variables

Untitled - Notepad

File Edit Format View Help

```
public class Example
{
   int x; //Instance variable
   static int y; //static memebr variable

   public void fun1() {  } //instance member functi
   public static void fun2() {  } //static member f

   static class Test
   {
   }
   public static void main(String[] args)
   {
   }
}
```

- In class we write static class test in the class this is static inner class.

Static variables

- Static variables are declared in the class using **static** keyword.
- Static variables are by default initialized to its default value
- Static variable has a single copy for the whole class and does not depend on the objects

**3rd point notable

Static variables

Untitled - Notepad

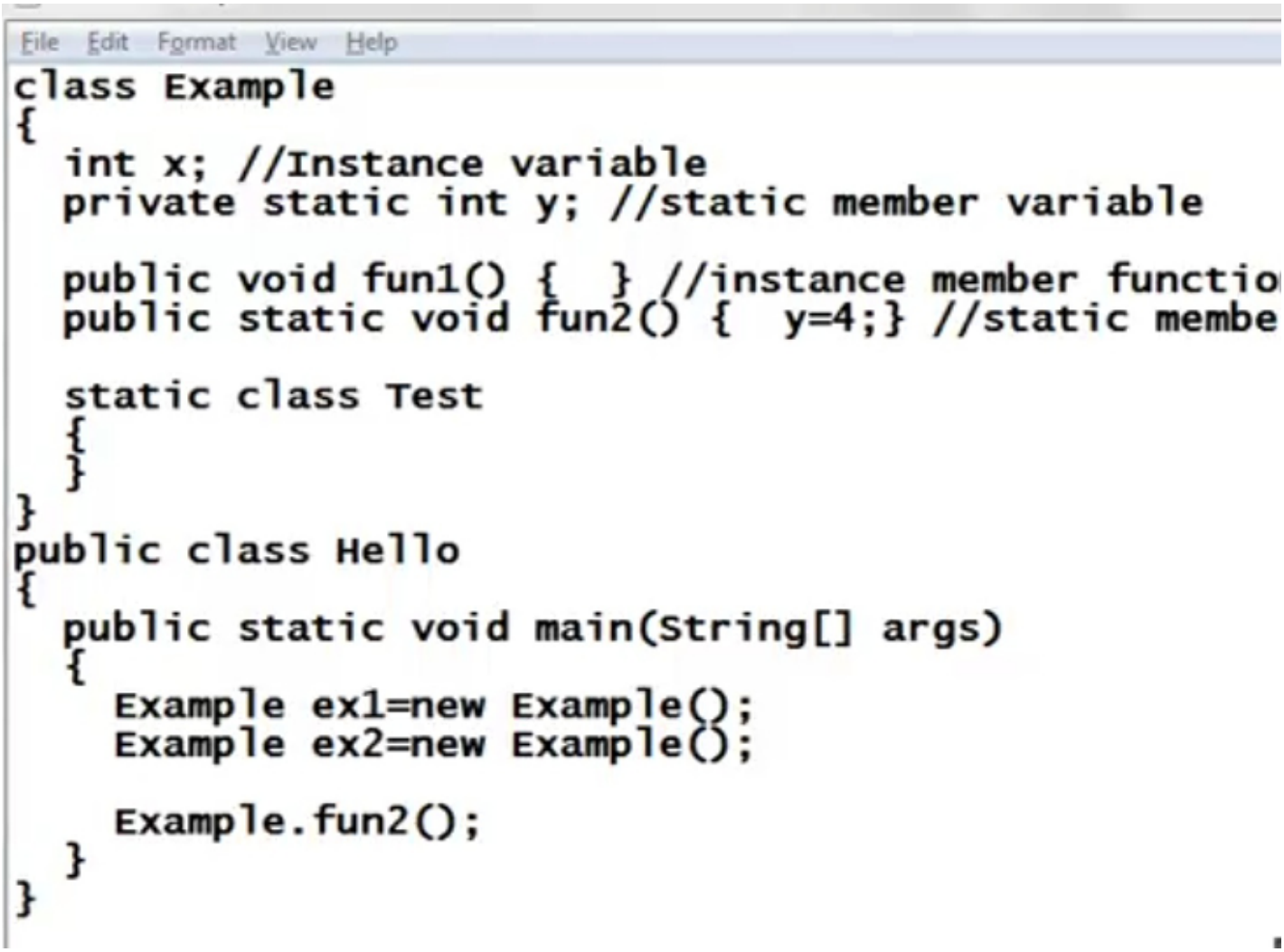
File Edit Format View Help

```
public class Example
{
  int x; //Instance variable
  static int y; //static member variable

  public void fun1() {  } //instance member functi
  public static void fun2() {  } //static member f

  static class Test
  {
  }
  public static void main(String[] args)
  {
    Example ex1=new Example();
    Example ex2=new Example();
    Example.y=5;
  }
}
```

Static function

- Static functions defined inside the class are qualified with the keyword static
- Static function can only access static members of the same class
- Static function can be invoked using class name and dot operator

- [ ] Static member function call without object.



```
File  Edit  Format  View  Help
class Example
{
  int x; //Instance variable
  private static int y; //static member variable

  public void fun1() {  } //instance member functio
  public static void fun2() {  y=4;} //static membe

  static class Test
  {
  }
}
public class Hello
{
  public static void main(String[] args)
  {
    Example ex1=new Example();
    Example ex2=new Example();

    Example.fun2();
  }
}
```

- Here y is private variable i.e we cannot access.
- But fun2 is public so we can access this.

```
class Example
{
  int x; //Instance variable
  private static int y; //static member variable

  public void fun1() {  } //instance member functior
  public static void fun2() {  y=4;} //static member

  static class Test
  {
    public static String country="INDIA";
  }
}
public class Hello
{
  public static void main(String[] args)
  {
    Example ex1=new Example();
    Example ex2=new Example();

    Example.fun2();
    System.out.println(Example.Test.country);
  }
}
```

Static class

- We can have a class inside a class which is known as inner class

- Inner class can be qualified with the keyword static

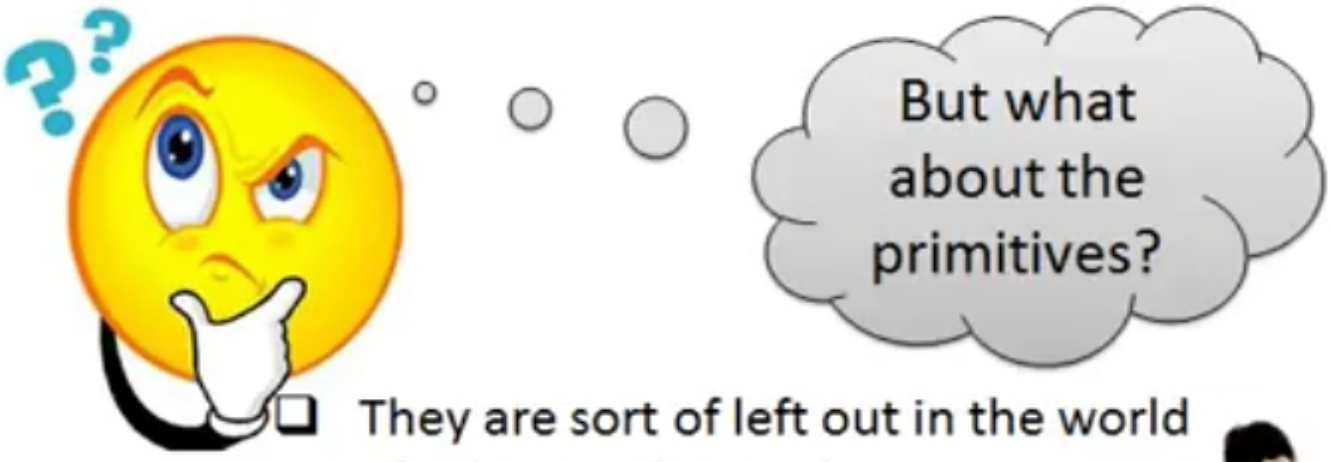
- Ex in bank rate of interest is same for all class i.e we cannot make object variable which is static and

Available in static

Lecture 8

JAVA is 99% OOP

- Java is an object-oriented language and as said everything in java is an object.



- They are sort of left out in the world of Objects, that is, they cannot participate in the object activities

Wrapper Classes

- As a solution to this problem, Java allows you to include the primitives in the family of objects by using what are called **wrapper classes.**

- There is a wrapper class for every primitive data type in Java.

Wrapper Classes...

- This class encapsulates a single value for the primitive data type

- For instance the wrapper class for int is Integer, for float is Float, and so on

Primitive type → Wrapper Class

boolean	→	Boolean
byte	→	Byte
char	→	Character
short	→	Short
int	→	Integer
long	→	Long
float	→	Float
double	→	Double

Useful methods of wrapper class

- **valueOf()**
  - Static method.
  - Return Object reference of relative wrapper class
- **parseXxx()**
  - Static method
  - Xxx can be replaced by any primitive type
  - It returns xxx type value
- **xxxValue()**
  - Instance method of wrapper class
  - Xxx can be replaced by any primitive type
  - Returns corresponding primitive type

- Static method is that function which doesn't require to make object.

Useful methods of wrapper class

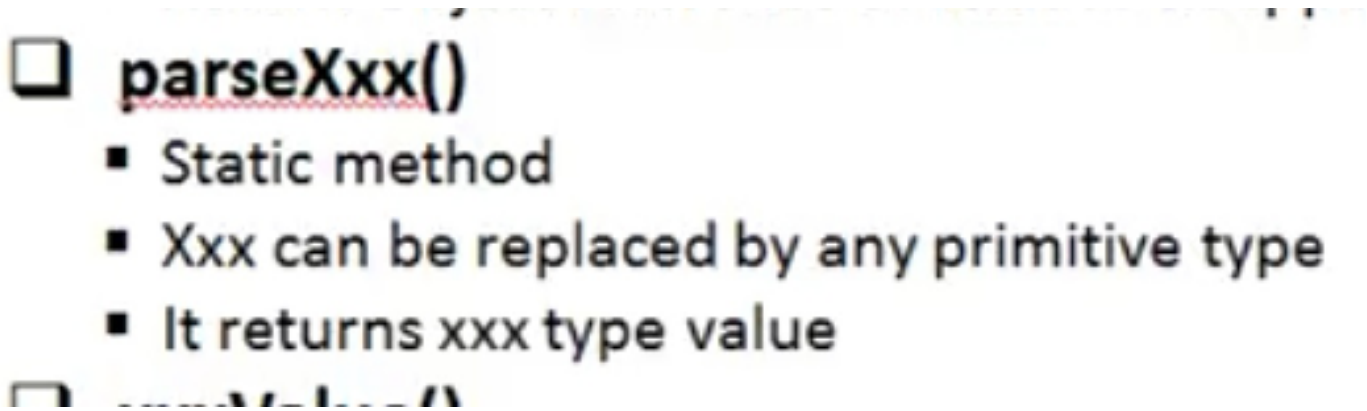
- valueOf()
  - Static method.
  - Return Object reference of relative wrapper class

Untitled - Notepad

File Edit Format View Help

```
public class Example
{
 public static void main(String[] args)
 {
  Integer i1= Integer.valueOf("101011",2);
  Double d1=Double.valueOf("3.14");

 }
}
```

- Under " " we write inside this is string.



- Parsexxx() here xxx is premetive type.
- This is static function i.e no need to make object.

```
public class Example
{
 public static void main(String[] args)
 {
  Integer i1= Integer.valueOf("101011",2);
  Double d1=Double.valueOf("3.14");
  int a=Integer.parseInt("123");
  double b=Double.parseDouble("13.45");

 }
}
```

- parseInt return int type value not integer
- In second line strig convert into double line.

xxxValue()

- Instance method of wrapper class
- Xxx can be replaced by any primitive type
- Returns corresponding primitive type

```
public class Example
{
 public static void main(String[] args)
 {
  Integer i1= Integer.valueOf("101011",2);
  Double d1=Double.valueOf("3.14");
  int a=Integer.parseInt("123");
  double b=Double.parseDouble("13.45");

  int c=i1.intValue();

 }
}
```

- Int return int type
- Float return float type.
- If data is in integer class then we take to use in ordinary int value then we need this.

Lecture 9

Command Line Arguments

- A Java application can accept any number of arguments from the command line.

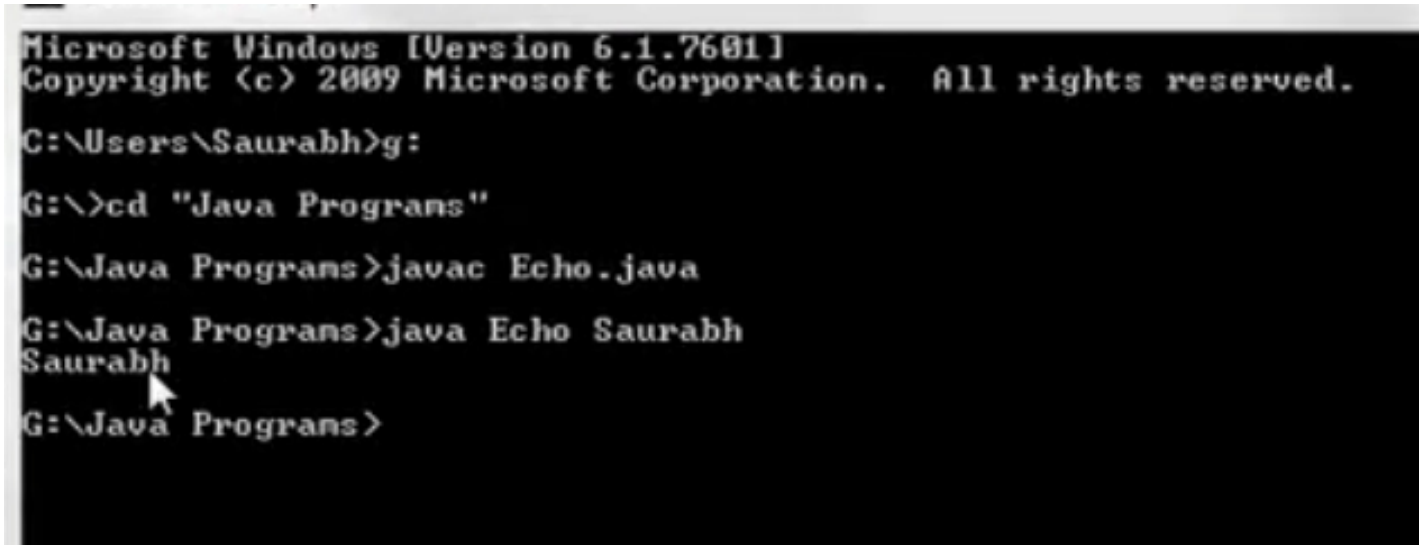
Command Line Arguments

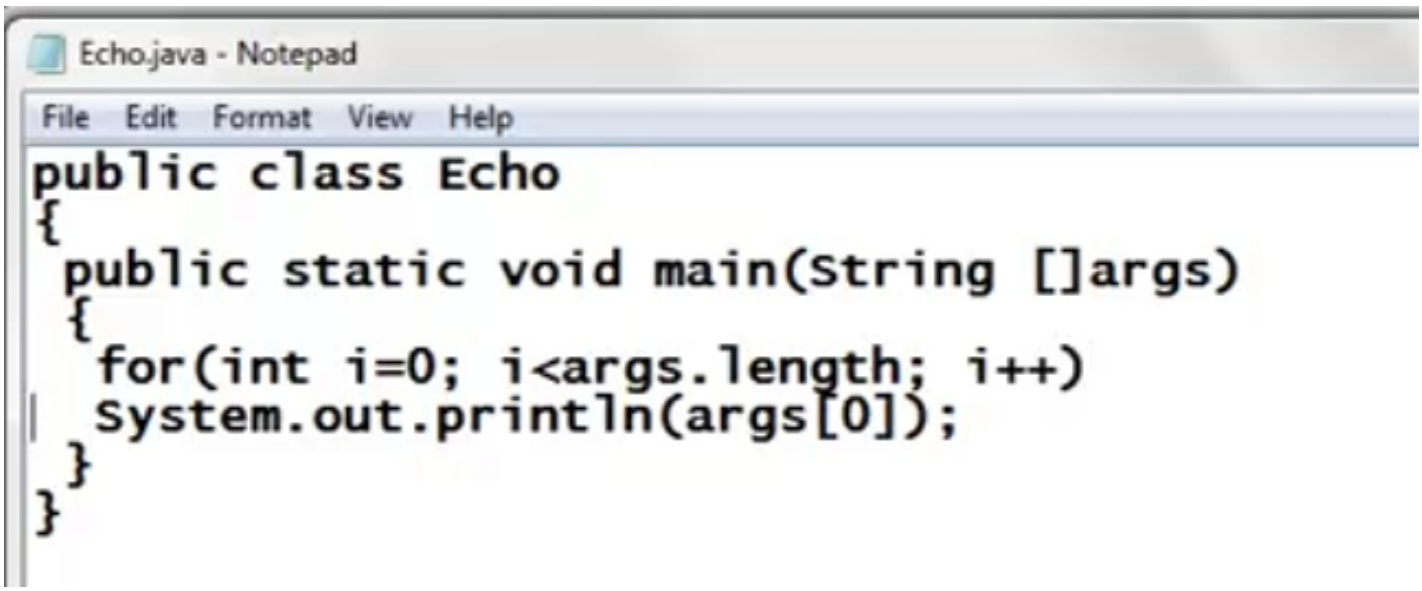
```
public class Echo
{
        public static void main (String[] args)
        {
          for (int i=0;i<args.length;i++)
                System.out.println(args[i]);
        }
}
```

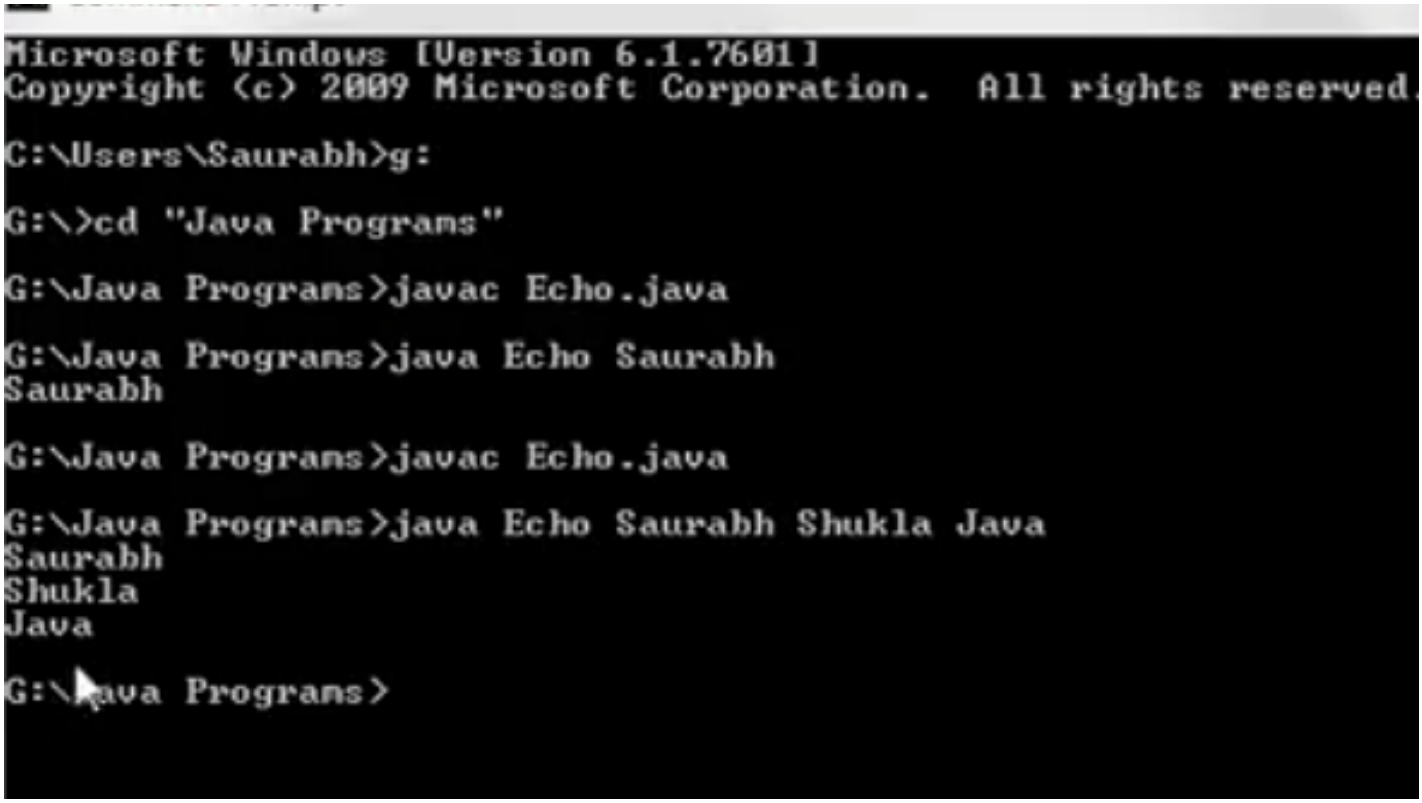
Echo.java - Notepad

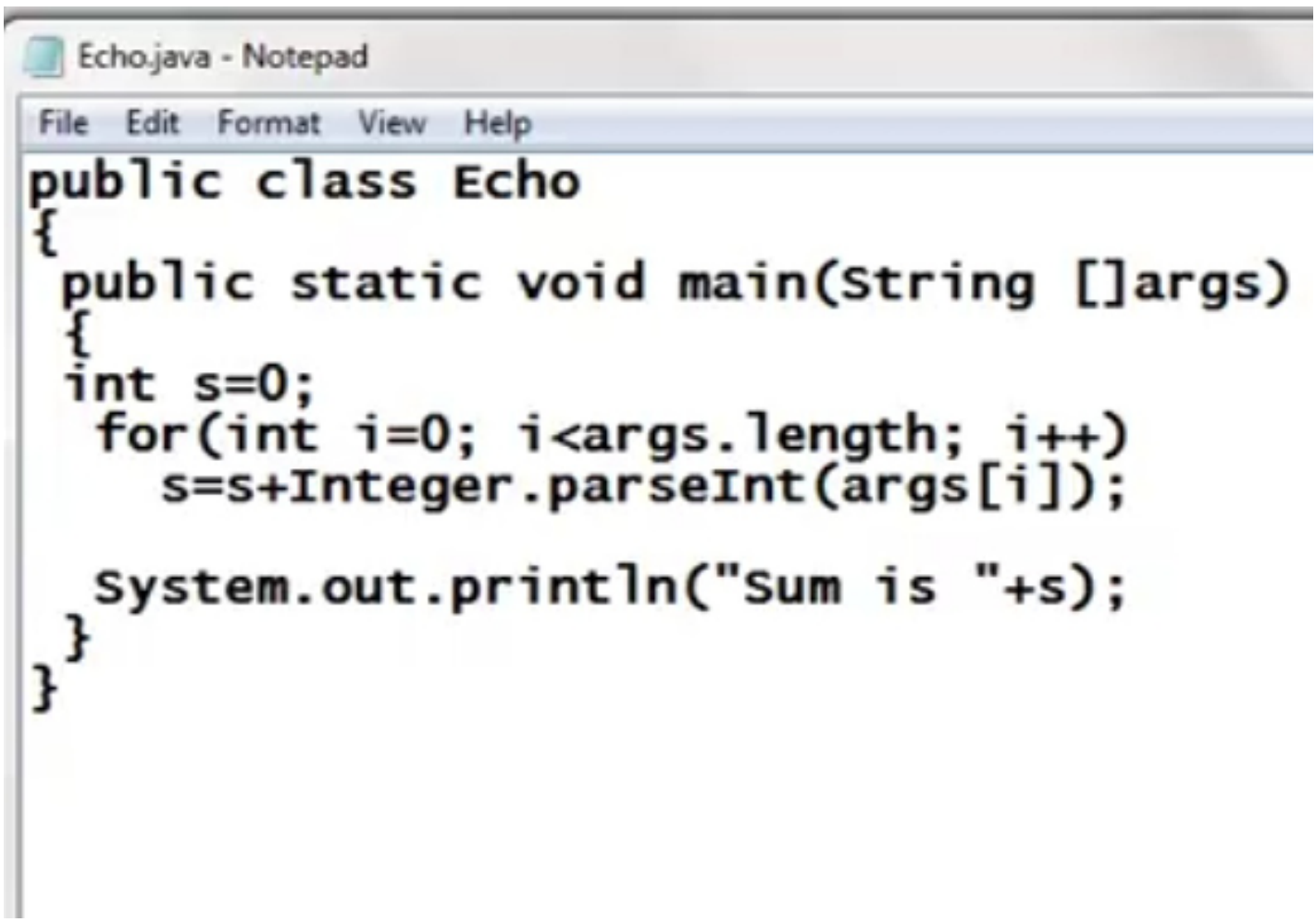
File Edit Format View Help

```
public class Echo
{
 public static void main(String []args)
 {
  System.out.println(args[0]);
 }
}
```

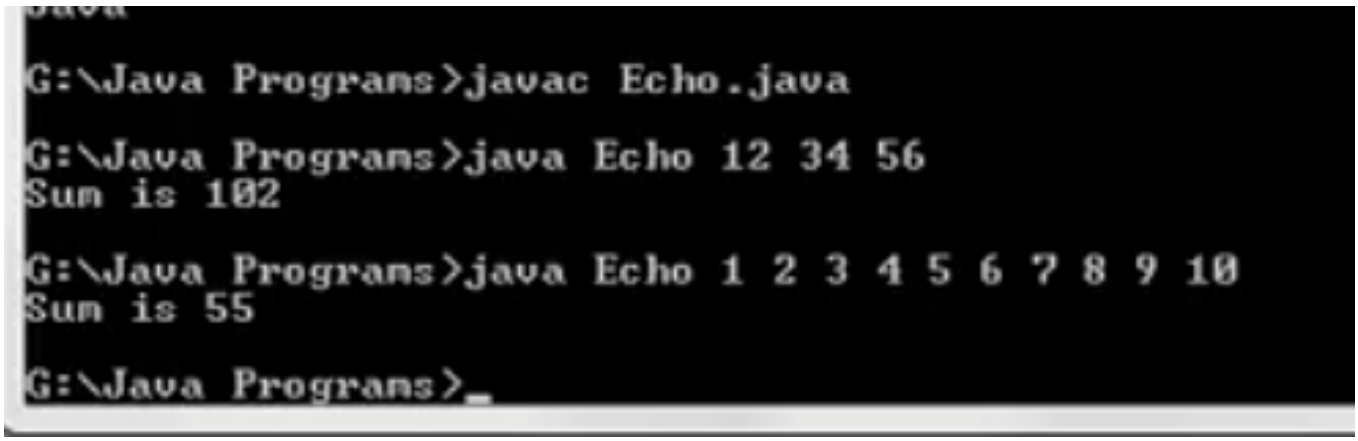

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Saurabh>g:
G:\>cd "Java Programs"
G:\Java Programs>javac Echo.java
G:\Java Programs>java Echo Saurabh
Saurabh
G:\Java Programs>


Echo.java - Notepad
File Edit Format View Help
public class Echo
{
public static void main(String []args)
{
for(int i=0; i<args.length; i++)
System.out.println(args[0]);
}
}





☐ This programme for sum number.



Lecture 10

Java Packages

- Packages are nothing more than the way we organize files into different directories according to their functionality, usability as well as category they should belong to
- Files in one directory (or package) would have different functionality from those of another directory.

Example

- For example: files in java.io package do something related to I/O, but files in java.net package give us the way to deal with the Network

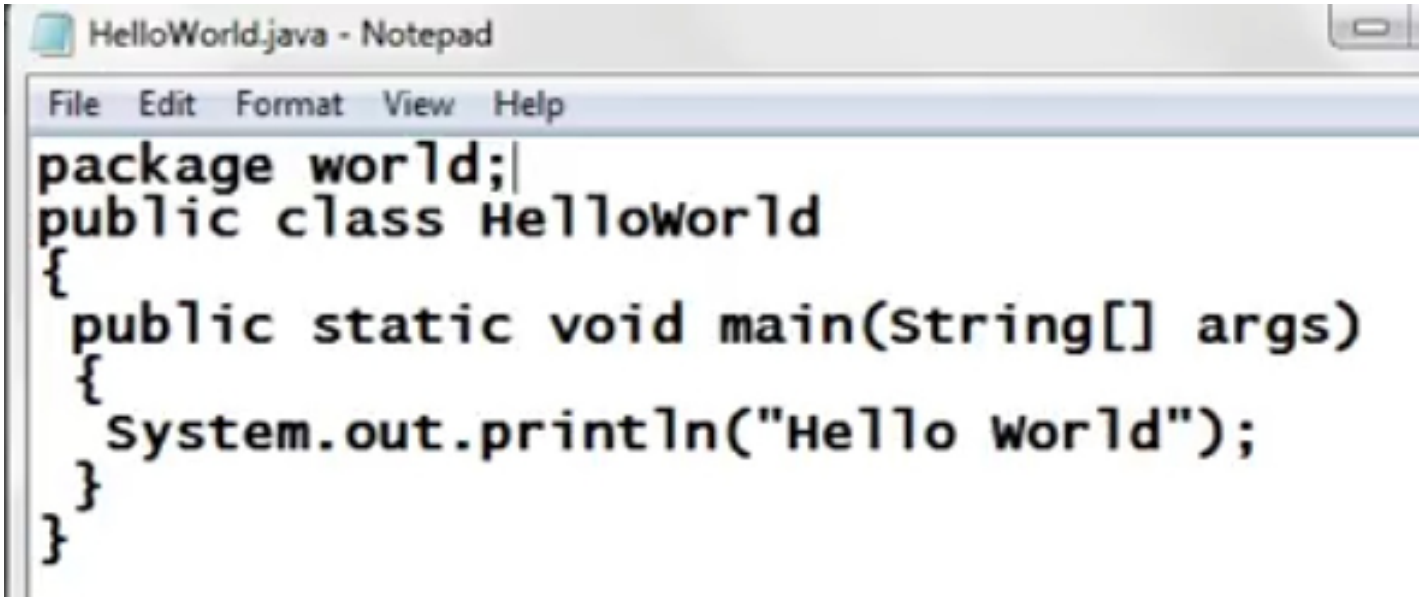
- Java.io is that package which have all class which work input and output related work.
- Java.net work network related class.

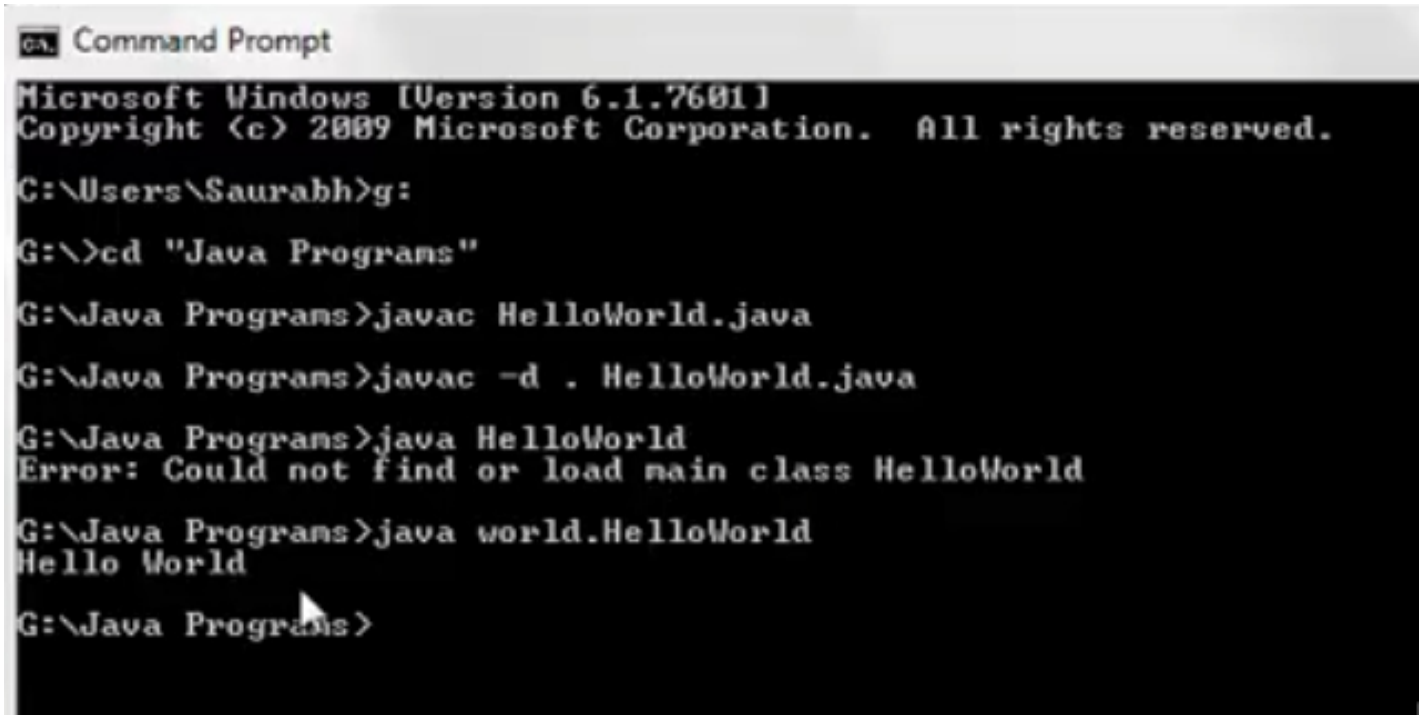
Name Collision

- Packaging also help us to avoid class name collision when we use the same class name as that of others
- The benefits of using package reflect the ease of maintenance, organization, and increase collaboration among developers

How to create package?

- Suppose we have a file called HelloWorld.java, and we want to put this file in a package **world**


HelloWorld.java - Notepad
File Edit Format View Help
package world;
public class HelloWorld
{
public static void main(String[] args)
{
System.out.println("Hello World");
}
}


Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Saurabh>g:
G:\>cd "Java Programs"
G:\Java Programs>javac HelloWorld.java
G:\Java Programs>javac -d . HelloWorld.java
G:\Java Programs>java HelloWorld
Error: Could not find or load main class HelloWorld
G:\Java Programs>java world.HelloWorld
Hello World
G:\Java Programs>

Remember

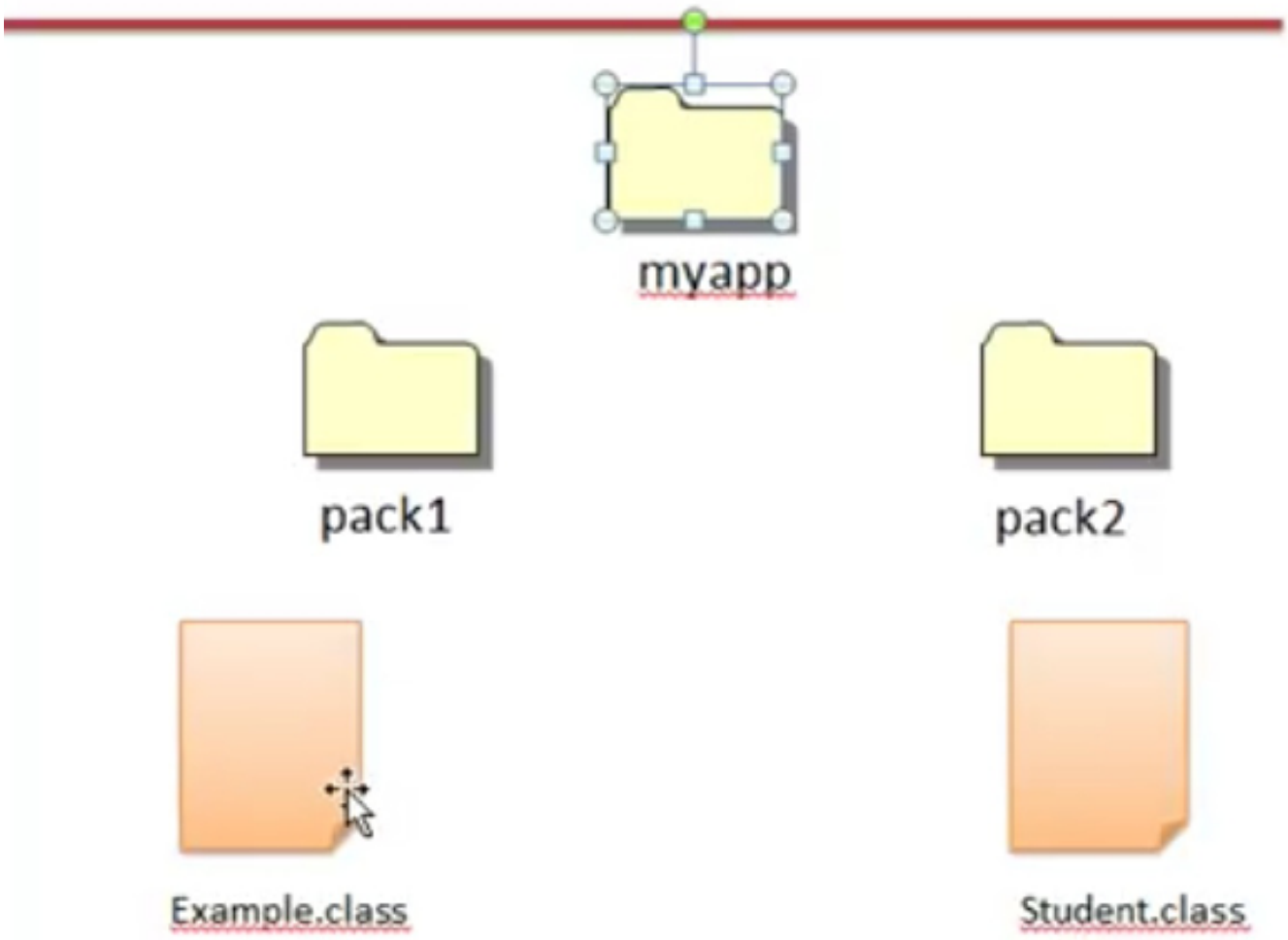
- We can have only one public class in a single java file.
- Name of the file should be same as the name of public class
- In absence of public class, any class name can be given to the file name.

Lecture 11

import

- import is a keyword in Java
- It is used to import classes of other packages

Example



Student.java - Notepad

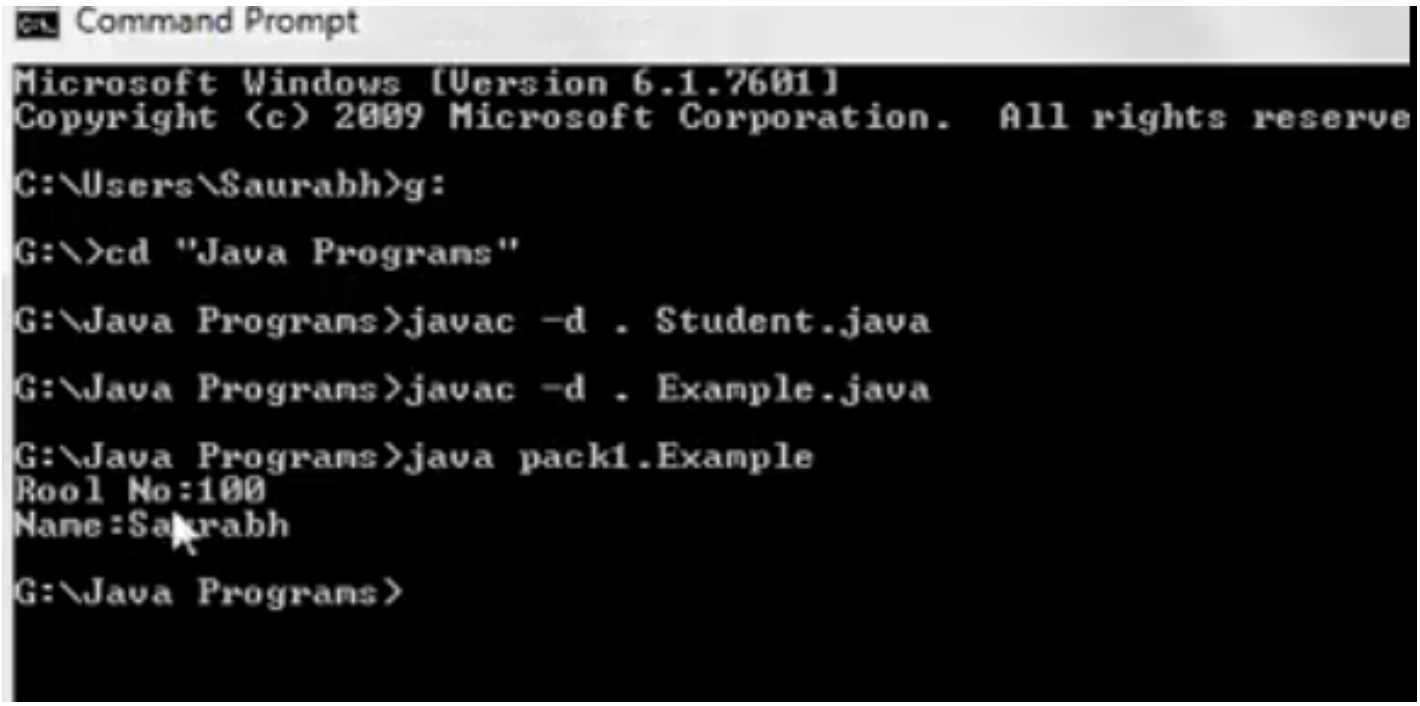
File Edit Format View Help

```
package pack2;
public class Student
{
 private int rollno;
 private String name;
 public void setRollno(int r)
 { rollno=r; }
 public void setName(String n)
 { name=n; }
 public int getRollno()
 { return(rollno); }
 public String getName()
 { return(name); }
}
```

Example.java - Notepad

File Edit Format View Help

```
package pack1;
import pack2.Student;
public class Example
{
 public static void main(String []args)
 {
  Student s1=new Student();
  s1.setRollno(100);
  s1.setName("Saurabh");
  System.out.println("Rool No:"+s1.getRollno());
  System.out.println("Name:"+s1.getName());

 }
}
```



Lecture 12

Java Access Modifiers

- Java supports four categories of accessibility rules
  - private
  - protected
  - public
  - default
- Modifiers can be used for class, member variables and member functions

With class

- Outer class and inner class
- For outer class, there can be only two possibilities, either class is a public class or just a class which means it is of default type.
- For inner class any among four access modifiers can be used

File Edit Format View Help

```
public class Example //outer class
{

 class Dummy //inner class
 {

 }

}
```

Remember

- There can be only one public class in a single java file.
- The name of the java file must be the same as the name of the public class.
- Only public class can be accessed directly from outside the package

Member variables and functions

- When members of the class are private, they can not accessed from outside the class body. They are meant to be accessed from the same class in which they are declared.
- When members are protected, they can be accessed from any class of the same package and child class from other package
- When members are public, they are accessible from any class of any package.
- When members are default, they are accessib only from the class of same package.

Lecture 13

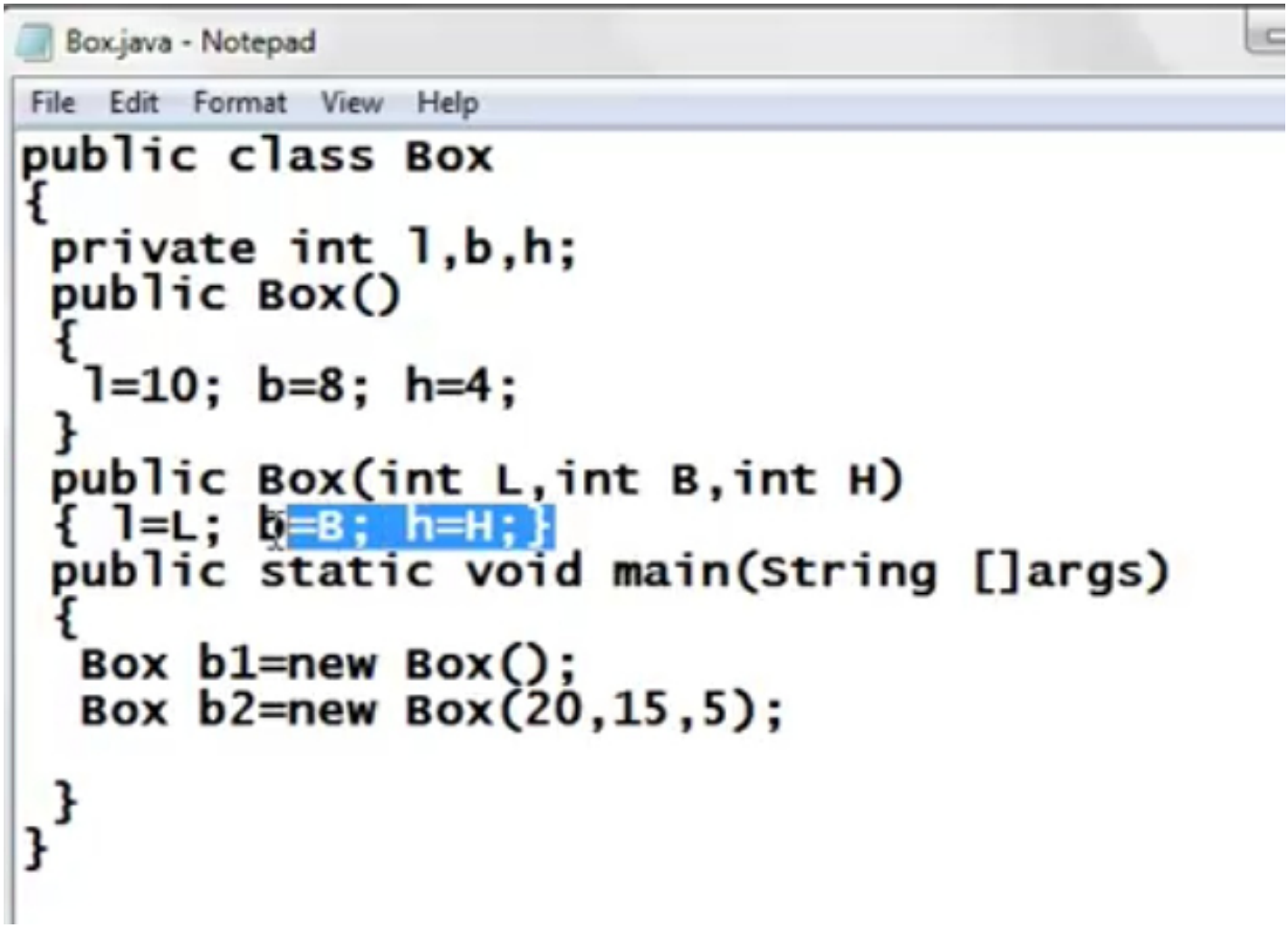
Constructors

- Constructor is a member function of a class
- The name of constructor is same as the name of the class.
- Constructor has no return type.

Box.java - Notepad

File Edit Format View Help

```
public class Box
{
 private int l,b,h;
 public Box()
 {
  }
}
```



- Box b1=new Box(); is a object,
- Constructor is automatically call by making of object.

- **Object represent real world entity.**

- **We can make more then one constructor.**

- Box b2=new Box(20,15,5);//Represent constructor.

Constructor is special

- A **constructor** is a special method that is used to **initialize a newly created object** and is called implicitly, just after the memory is allocated for the object
- It is **not mandatory** for the coder to write a constructor for the class
- When there is no constructor defined in the class by programmer, compiler implicitly provide a default constructor for the class.

Constructor Overloading

- Constructors can be parameterized
- Constructors can be overloaded

Last modified: 14 Sep 2019