Java Programming

Basic Concept of Java Programming Language

Dipendra Shrestha

* Java first program :

Package create गर्दा, यो create गर्दा नै 'package' बनेर आउँछ

```
package practice_1;

public class hello {
    public static void main (String[] args)
    {
        System.out.println ("Hello world");
    }
}
```

Inbuilt class हो

Java

- Java was developed in the early 90s by Sun Microsystems.
- Java is high-level language.
- Java programs are portable across platforms.
  - Each program is translated into Java bytecode
  - Each machine has a Java virtual machine (JVM) which knows how to execute Java bytecode
- Java is object oriented.

Scanned with CamScanner

⊛ Variable in Java :

↱ Variables are names that we declare to store the value.

Eg :

```
package practice_2;

Public class Variables{

        Public static void main(String [] args)
        {
(Integer) → int my_variables = 10;
            System.out.println (my_variables);
        }
}
```

⊙ Data types :

byte (number, 1 byte)
Short (number, 2 bytes)
int (number, 4 bytes)
long (number, 8 bytes)
float (float number, 4 bytes)
duble (float number, 8 bytes)
Char (a character, 2 bytes)
boolean (true or false, 1 byte)

By Dipendra Shrestha

Scanned with CamScanner

(*) Getting user Input :

```
package practice_3;
import java.util.Scanner;
```

(input लिन लाई Scanner भन्ने class लाई import गर्नु पर्छ।)

```
Public class UserInput {

    Public static void main (String[] args)
    {   //for Integer Input.
        Scanner scan = new Scanner (System.in);
        System.out.println ("Enter some number");
        int user_input = scan. nextInt ();
        System.out.println ("The Entered value is:");
        system.out.println (user_input);

        //for character or string.
        Scanner scan1 = new Scanner (system.in);
        System.out.println ("Enter Some character");
        String user_input1 = scan1. next ();
        System.out.println (" The Entered String is:");
        system.out.println (user_input1);
    }
}
```

(To get input)

(Java को Class हो जस्ले input लिनमा help गर्छ।)

(यहाँ धेरै खोलको class हुन्छ like nextInt, nextBoolean, nextByte, nextFloat ⋮)

Scanned with CamScanner

⊛ Math & Arithmetics :-

```
package practic_4;

Public class Math_Arithmetic {

    public static void main (String[] args)
    {
        int x, y, answer;
        x = 5;
        y = 10;
        answer = x + y ;
        System.out.println ("Answer = " + answer);
    }
}
```

यहाँ mathematical operator like *, /, -, % हुन् सक्छ

Concatenate गर्ने तरिका

⊙ Increment Operator :-

~~Eg :-~~

```
int x = 10;
x = x++;      → 10
x = x+5;      → 15
x += 5;       → 15
x *= 5;       → 50
x -= 5;       → 10
x = ++x;      → 11
x /= 5        → 5
```


Scanned with CamScanner

⊛ Conditional Statement :

```
if (boolean expression)
 {
    execute if the boolean expression is true.
 }

else
 {
    execute if the boolean expression is false.
 }
```

⊙ Operator that can be used within 'if' statement.

= =	→	is equal to
! =	→	is not equal to
>	→	is greater than
<	→	is less than
>=	→	is greater than or equal to
<=	→	is less than or equal to.

⊙ Logical operator

&& → AND

|| → OR

By Dipendra Shrestha

Scanned with CamScanner

⊛ Switch Statement :

Syntax : (User ले define गरेको हुन्छ।)

```
Switch (Case_variable)
{
    case 1 :
        - - -
        - - -
        break;
    case 2 :
        - - -
        - - -
        break;
    :
    :
    default :
        - -
        break;
}
```

⊛ While Statement : (While loop)

Loop : A loop is piece of statement that executes some block of code again & again until some condition is met.

Syntax : (User ले दिन्छ।)

```
while (condition)
{
    - - -
    increment;
}
```

By Dipendra Shrestha

Scanned with CamScanner

(*) Arrays in Java :

↳ Same as variable but can store multiple values of same type.

↳ Array can be declared in Java in different ways :

Eg :

```
int myIntArray [] = {1, 2, 3, 4, 5};
```

(Mostly prefered) ✓ `int [] myIntArray = {1, 2, 3, 4, 5};`

```
int [] myIntArray = new int [5];
int[] myIntArray = new int [] {1, 2, 3, 4, 5};
```

(*) For Loops in Java :

Syntax :

```
for ( initialization ; expression ; increment )
{
    ----
}
```

(•) Another way to use for Loop with Arrays

Easy method

```
for ( int variable_name : Array_name)
{
    ---
}
```

Eg :

```
for (int element : myIntArray)
{ system.out.println (element);
}
```

By Dipendra Shrestha

Scanned with CamScanner

(*) String in Java :

- String is the sequence of characters.
- It can also be said as 'character Array'.

Eg :

```
String myString = "Hello I am Dipen";
System.out.println (myString);

//Length of String.
int myStringLength = myString.Length ();
System.out.println (myStringLength);

// Converting String to Lower Case.
String myStringLowerCase = myString.toLowerCase ();
System.out.println (myStringLowerCase);

// Converting String to Upper Case.
String myStringUpperCase = myString.toUpperCase ();
System.out.println (myStringUpperCase);

// replacing the Character of string.
                    (oldChar) (newChar)
System.out.println (myString.replace ('D', 'J'));

// get the position of the character.
System.out.println (myString.indexOf ('D'));
```

By Dipendra Shrestha

Scanned with CamScanner

⊛ Method / function in Java

↱ Method are also known as function.

Eg:

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

}
```

(This is also the method.)

Example :

```
package practice.12
public class Method {

    Public static Void main (String[] args)
    {
        myFirstMethod ();

    }
```

(Access Specifier ET)

(Keyword)

```
public static Void myFirstMethod ()
{
    System.out.println ("Hello Everyone");

}
```


Scanned with CamScanner

⊙ passing parameter to Method :

Eg :

```
public class ParameterToMethod {

    public static void main (String[] args)
    {
        SayHello ("Dipendra");
        Add (5, 10);
    }

    // Method with parameter
    public static void SayHello (String name)
    {
        System.out.println ("Hello" + name);
    }

    // Method with multiple parameter.

    Public static void Add (int x, int y)
    {
        int add = x + y;
        System.out.Println ("Addition value = " + add);
    }
}
```


Scanned with CamScanner

Returning Value :

Example :

```
public class returningValue
{
    public static void main (String[] args)
    {
        int ans = add (5, 10)   //(return गरेको value 'ans' मा आएर बस्छ ।)
        System.out.println ("The Addition value = " + ans);

    }

    public static int add (int x, int y)
    {
        int answer = x + y;
        return (answer);
    }

}
```

By Dipendra Shrestha

Scanned with CamScanner

Class & Object in Java

Class :

- Class is data structure or type inside which we can define 'member variables' and 'methods'.

- The most Imp property of class is we can create Instance of class known as object.

Example : Class with it's Instance (object)

Student.Java

```
package practice_13;
public Class Student
{
    int id;
    int age;
    String name;
}
```

यो class लाई MyClass.Java file ले object बनाएर चलाएको छ यो class लाई।

Scanned with CamScanner

'new'
↳ It initiates a class by allocating memory for a new object and returns a reference to that memory.
↳ It invokes the object constructor

यो keyword हो used to create instance of class.

MyClass.Java

```
package practice_13;

public class Myclass
{
    public static void main (String[] args)
    {
        Student mark = new Student ();

        mark.id = 1;
        mark.name = "Mark";
        mark.age = 18;

        System.out.println (mark.name + "is " + mark.age
                            + "years old.");
    }
}
```

Class name

Object of Student class

खासमा यो class को use यस्ती गर्दा data जस्ले नि change गर्न सक्छ, हामी fn बनाएर class को use गर्छौं।

Scanned with CamScanner

⊙ Proper way to access Class in Java :

Student.Java :

```
package practice_13;
public class Student
{
    int id;
    int age;
    String name;

    // getter & Setter for id
    public int getId()
    {
        return id;
    }
    public void SetId (int id)
    {
        this.id = id;
    }
    // getter & Setter for age
    public void setAge (int age)
    {
        this.age = age;
    }
    public int getAge()
    {
        return age;
    }
```

यो getter र Setter हरू IDE आफैले generate गर्दिने बनाऊन पनि मिल्छ।

यो 'this.id' ले माथी declare गरेको id लाई जनाउँछ।

यो चाहिं parameter बाट आएको id हो।

Scanned with CamScanner

```
//getter & Setter for name
        public String getName ()
        {
            return name;
        }
        public void SetName (String name)
        {
            this. name = name;
        }
}
```

MyClass.Java

```
package Practice_13;
public class Myclass
{
    public Static void main ( String [] args)
    {
        Student mark = new Student ();
        mark. SetId (1);
        mark. SetName ("Mark");
        mark. SetAge (18);

        System. out. println (mark. getName () + " is"
                + mark. getAge() + "yrs old");
```

By Dipendra Shrestha

Scanned with CamScanner

⊛ Constructor in Java :

- In Java, a Constructor is a block of codes similar to the method.
- It is called when an instance of the class is Created.
- At the time of calling constructor, memory for the object is allocated in the memory.
- It is special type of method which is used to initialize the object.
- Every time an object is created using the new() Keyword, at least one Constructor is called.
- It calls default Constructor if there is no Constructor available in the class. In such case, Java compiler provides a default Constructor by default.

Note :- It is called Constructor because it constructs the values at the time of object creation. It is not necessary to write a Constructor for a class. It is because Java Compiler creates a default Constructor if your class doesn't have any.

By Dipendra Shrestha

Scanned with CamScanner

⊛ Rules for Creating Java Constructor :

1) Constructor name must be the Same as it's Class name.
2) A Constructor must have no explict return type.
3) A Java Constructor can't be abstract, static, final & Synchronized.

Example :

Cube.Java

```
package practice_14;
public class Cube
{
    int length;
    int breadth;
    int height;
    public int getCubeVolume ()
    {
        return (Length * breadth * height);
    }

    Cube () // यो default Constructor हो।
    {
        Length = 10;
        breadth = 20;
        height = 30;
    }

    Cube (int L, int b, int h) // Parameterized Constructor.
    {
        Length = L;
        breadth = b;
        height = h;
    }
}
```

Constructor Overloading को Example पनि माथि राखिन्छ।

Scanned with CamScanner

Constructor.java

```
package practice_14;

public class Constructor
{
    public static void main (String[] args)
    {
        Cube cube1 = new Cube();
```

यो सधैं default Constructor हो। यसले default Constructor लाई call गर्छ।

```
        System.out.println (cube1.getCubeVolume());
```

यसलाई Constructor overloading पनि भन्छ।

```
        Cube cube2 = new Cube(20, 30, 40);
```

parameterized Constructor लाई call गर्छ।

```
        System.out.println (cube2.getCubeVolume());
    }
}
```

Output :

6000
24000

Scanned with CamScanner

(*) Overloading Method :

- In overloading of method, we define two or more method within the same class that share the same name, as long as their parameter declarations are different.

- When this is the case, the methods are said to be overloaded and the process is referred to as method overloading.

Example :

```
package Practice_15;
public class MethodOverloading
{
    public static void main (String[] args)
    {
        int SumInt = add ( 5, 6) ;
        double SumDouble = add ( 5.4, 6.6) ;

        System.out.println ("SumInt =" + SumInt);
        System.out.println ("SumDouble =" + SumDouble);
    }

    public static int add (int a, int b)
    {
        return (a+b);
    }
    Public static double add( double a, double b)
    {
        return (a+b) ;
    }
}
```

Overloading Method

Same fn name but different return type & Parameter.

Scanned with CamScanner

→ Static method बनायो भने यो method लाई access गर्दा class name बाटै access गर्न सकिन्छ, class को object बनाएर obj. गरेर यसलाई झन्झट गरिराख्नु पर्दैन।

* Static Method /function :

- If you want to define a class member that will be used independently of any object (Instance) of that class then Static method can be used.

- Normally, a class member must be accessed only in conjunction with an object of its class.

- However, it is possible to create a member that can be used by itself, without reference to a specific instance (object).

- To create such a member, precede its declaration with the 'keyword Static'.

- When a member is declared static, it can be accessed before any objects of its class are created, and without reference to any object.

- We can declare both (methods) & (variable) to be static.

⊙ Method declared as static have several restrictions :-

1) They can only call other static methods.
2) They must only access static data.
3) They can't refer to (this) or Super in any way.

By Dipendra Shrestha

Scanned with CamScanner

Example :

myClass.java

```
package practice_16 ;
public class myClass {

    Public static String doSomething( String message)
    {
        return message ;
    }

    Public String doSomethingElse (String message)
    {
        return message ;
    }
}
```

यो static method हो।

यो non-static लाई चाहिं 'object' बनाएर मात्र access गर्न मिल्छ।

StaticMethod.java

```
package practice_16;
public Class StaticMethod {
    public static Void main (String[] args)
    { // Static Member belongs to the class instead of specific object
        myClass object = new myClass ();
        // object. doSomething ("Hello") ;
        String message ;
        message = myClass. doSomething ("Hello") ;
        System. out. println ("message") ;
    }
}
```

यसरी Instance को through access गर्दा static member लाई error त आउँदैन but not the correct way.

यसरी class को name बाट static method लाई access गर्ने गरिन्छ।

Scanned with CamScanner

Variable लाई static बनाउँदा चाहिं Same variable लाई class को different Instance (object) हरूले Shareablly use गरिरहेको हुन्छ ।

* Static Variable :

- Static Variable declared as static are essentially global variables.
- Static variable can be declared by adding keyword 'Static' at the precede of declaration of variable.

  Eg:

  ```
  Static int number;
  ```

- When objects of its class are declared, no copy of a static variable is made.
- Instead, all instances of the class share the same static variables.

Example : myClass1.java

```
package practice_16;
public class myClass1 {
    Static int noOfStudent = 0;
    myClass1 ()
    {
        noOfStudent ++;
    }
    public int getNoOfStudent ()
    {
        return noOfStudent;
    }
}
```

By Dipendra Shrestha

Scanned with CamScanner

StaticVariable.Java

```
package practice_16;
public class StaticVariable
{
    public static void main (String[] args)
    {
        myClass1 obj = new myClass1();
        System.out.println(obj.getNoOfStudent());

        myClass1 obj1 = new myClass1();
        System.out.println(obj1.getNoOfStudent());
    }
}
```

यो object बनाउँदा एकचोटि constructor call भएर static variable increment हुन्छ।

यो object बन्दा फेरी constructor call हुन्छ र static भएको कारण अघिको value मा फेरी increment हुन्छ।

output:

1
2

This shows that static variable चाहिँ different object हरूले share गरेको हुन्छ।

By Dipendra Shrestha

Scanned with CamScanner

⊛ Access Modifiers in Java :

→ It tells the how member of class can be accessed within a package or outside the package.
(यो धेरैको collection of classes हुन्छ।)

	Modifier	Class	Package	Subclass	World
1)	public	Y	Y	Y	Y
2)	protected	Y	Y	Y	N
3)	private	Y	N	N	N
4)	No modifier	Y	Y	N	N

⊙ Public Modifier :

Eg :

PublicModifiers.Java

```
package practice_17;
public class publicModifiers
{
    public String name;
}
```

Member variable लाई 'public' define गरेर access गर्नु isn't the best practice

Scanned with CamScanner

AccessModifiers.Java

```
package practice_17;
public Class AccessModifiers
{ Public static Void main (String[] args) {
    // creating object of publicModifiers class;

    PublicModifiers public_obj = new publicModifiers();
    public_obj.name = "Dipendra"

    System.out.println ("Name :- " + public_obj.name);
}
}
```

यो member Variable 'Public' declare भएको ले 'object' बनाएर यसलाई direct access गर्न मिलेको हो।

Private Modifier :

Eg :

PrivateModifiers.Java

```
package practice_17;
public class privateModifiers
{
    Private String name;
    // Class को variable लाई private declare गर्नु is good practice.
    // यसलाई class को बाहिर बाट direct access गर्न मिल्दैन।
    // यसलाई access गर्न member function बनाउनु पर्छ which
    // IDE can automatically generate for you known as getter and
    // Setter.

    Public String getName()
    {
        return name;
    }
```


Scanned with CamScanner

```
public void SetName (String name)
{
    this.name = name;
}
```

यो 'this' ले class मा declare भएको member variable लाई जनाउँछ।

यो name ले Local variable i.e आएको Parameter लाई जनाउँछ।

AccessModifiers.Java

```
package practice_17;
public Class AccessModifiers {

    public Static void main (String[] args)
    {
        //Creating object for privateModifiers Class.
        privateModifiers private_obj = new privateModifiers();

        private_obj.SetName("Dipen");

        System.out.println("Name :- " + private_obj.getName());
    }
}
```

Private member लाई access गर्दा member function बाट मात्र गर्न सक्छौं।

Scanned with CamScanner

(*) Final Variables :

- A variable can be declared as final.
- Doing so prevents its contents from being modified.
- This means that we can initialize a final variable when it is declared. and can only be initialize once in a program.
- Final is similar to 'const' in C/C++/C#.
- 'Final' keyword has numerous way to use.
  - A final class can't be subclassed.
  - A final method can't be overridden by subclasses.
  - A final variable can only be initialized once.

Example of 'final variable' :

Eg : FinalVariable.Java

```
package Practice_18;
public class FinalVariable
{
    Public final int number;

    FinalVariable()
    {
        number = 10;
    }
}
```

Constructor बनाउँदा फाइदा के भो भन्दा जब हामी class को Object बनाउँदा तब नै Constructor ला भईदिन्छ & variable initialize भईदिन्छ So,

Constructor बनाएर initialize गरेको 'Final' variable लाई।

Scanned with CamScanner

Final.Java

```
package practice_18;
public class Final
{
    public static void main (String[] args)
    {
        FinalVariable obj = new FinalVariable();
        obj.number = 12;
        System.out.println(obj.number);
    }
}
```

Final variable (i.e number) already initialize भई सक्छ र केही त्यो variable लाई initialize गर्दा चाहि error आउँछ।

यो constructor बनाउँदा नै 'number' भन्ने 'final' variable initialize भई सक्छ।

Scanned with CamScanner

* Inheritance in Java :

- Inheritance is one of the cornerstones of object-oriented programming because it allows the creation of hierarchical classification.

- Using inheritance we can create a general class that defines traits (characterstics) common to a set of related items.

- Class can be inherited by other, more specific classes, each adding those things that are unique to it.

- Class that is inherited is called Super class.

- Class that does the inheriting is called Subclass.

- Subclass is a specialized version of a Superclass.

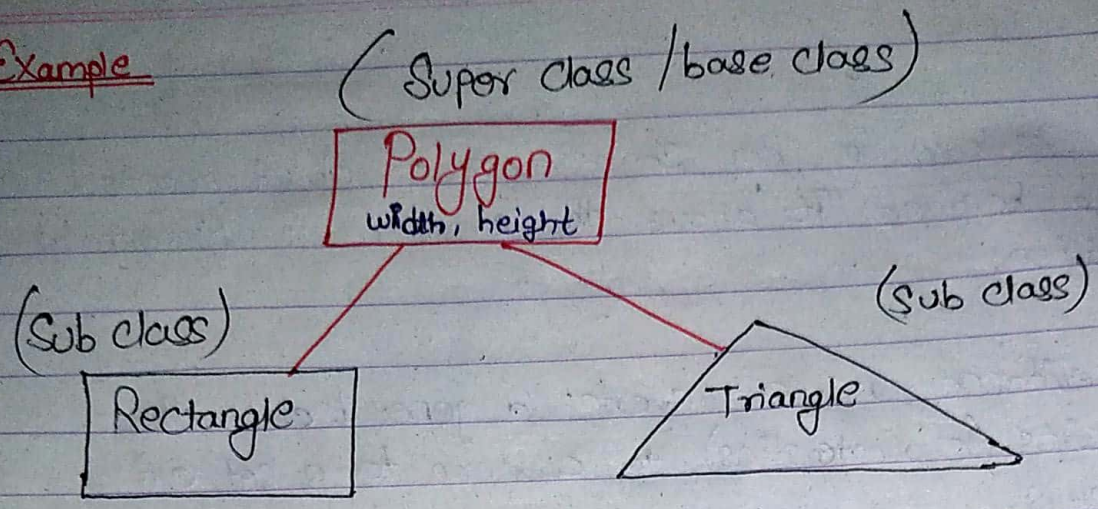
- It inherits all of the instance variables & methods define by the superclass and adds its own, unique element.

- We use 'extends' keyword in Java for inheriting class.

By Dipendra Shrestha

Scanned with CamScanner

Example



↳ 'Polygon' मा define गरेको 'width' र 'height' भन्ने member लाई 'Rectangle' & 'Triangle' class ले access गर्न को लागी inheritance use गर्ने।

⊙ polygon.java (Super Class)

```
package_practice_19;
public class Polygon
{
    protected int width;
    protected int height;

    Protected void SetValues (int a, int b)
    {
        width = a;
        height = b;
    }
}
```

By Dipendra Shrestha

Scanned with CamScanner

Rectangle.Java

```
package practice_19;
public class Rectangle extends polygon
{
    int areaOfRectangle ()
    {
        return (width * height);
    }
}
```

(Super class)

// Polygon class लाई inherit गरेको ले width र height यहाँ accessible भयो।

Triangle.Java

```
package practice_19;
public class Triangle extends polygon
{
    double areaOfTriangle ()
    {
        return (width * height)/2 ;
    }
}
```

(Super class)

By Dipendra Shrestha

Scanned with CamScanner

Inheritance.Java

```
package practice_19;
public class Inheritance
{
    public static Void main (String[] args)
    {
        polygon pol = new Polygon();
        Rectangle rec = new Rectangle();
        Triangle tri = new Triangle();

        rec.SetValues (10, 10);
        tri.SetValues (10, 10);

        System.out.println (rec.areaOfRectangle());
        System.out.println (tri.areaOfTriangle());
    }
}
```

Polygon class मा
भएको SetValues fn
member लाई 'Rectangle'
र 'Triangle' भन्ने class ले
access गर्न पाउँछ becz of
inheritance

Scanned with CamScanner

⊛ Super Keyword :

↱ Super has two general forms.

1) The first calls the superclass constructor

2) The second is used to access a member of the superclass that has been hidden by a member of a subclass.

1) Using super to call superclass constructors.

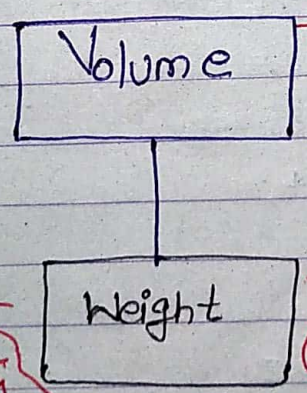
↱ A subclass can call a constructor method defined by its superclass by use of the following form of super:

Super (parameter-list)

Here, parameter-list specifies any parameter needed by the constructor in the superclass.

↱ Super ( ) must always be the first statement executed inside a subclass constructor.

Example



यो एउटा Super class हो र यसमा Constructor हरू define गरेको छ।

यसमा 'Volume' भन्ने class लाई 'inherit' गरिएको छ र superclass 'Volume' को constructor लाई call गर्दा 'Super( )' keyword use गरेर access गरेको छ।

Main f^n बाट जब 'Weight' class को object बनाएर parameter pass गरिन्छ तब 'Weight' class को constructor ले 'Volume' class को constructor लाई call गरेउँछ र 'Weight' class को object बाट सबै काम perform गरेको हुन्छ।

Scanned with CamScanner

Volume.java : (super class)

```
package practice_20;
public class Volume
{
    protected int length;
    protected int breadth;
    protected int height;

    //Default Constructor
    Volume ( )
    {
        Length = -1;      // uninitialized
        breadth = -1;     // हो '-1'
        height = -1;      // अनेको।
    }

    // Volume for Cuboid

    Volume (int L, int b, int h)
    {
        Length = L;
        breadth = b;
        height = h;
    }

    //Volume of Cube
    Volume (int L)
    {
        Length = breadth = height = L;
    }
```

सब Constructor हरू हो।

Scanned with CamScanner

```
int calcVolume()
{
        return length * breadth * height;
}
```

Weight.java

```
package practice_20;
public class Weight extends Volume
{
        double Weigh;
        // Default Constructor;
        Weight()
        {
            Super();
            Weigh = -1;
        }
        // For cuboid along with its weight.
        Weight(int L, int b, int h, double m)
        {
            Super(L, b, h);
            Weigh = m;
        }
        // For cube along with its weight.
        Weight(int L, double m)
        {
            Super(L);
            Weigh = m;
        }
}
```

यस्ले Super Class 'Volume' को Constructor लाई initialize गर्दिन्छ

Scanned with CamScanner

Superkeyword.Java

```
package practice_20;
public class Superkeyword
{
    public static void main (String[] args)
    {
        Weight Weig = new Weight(2, 3, 4, 6.5)
```

Cuboid वाला Constructor of Weight class.

```
        Weight Weig1 = new Weight(2, 6.5);
```

Cube वाला Constructor of Weight class.

```
        Weight Weig2 = new Weight();
```

Default Constructor.

```
        int Vol = Weig.CalcVolume();
        System.out.println("Volume = " + Vol);
        System.out.println("Weight =" + Weig.Weigh);

        int Vol1 = Weig1.CalcVolume();
        System.out.println("Volume =" + Vol1);
        System.out.println("Weight =" + Weig1.Weigh);

        int Vol2 = Weig2.CalcVolume();
        System.out.println("Volume =" + Vol2);
        System.out.println("Weight= " + Weig2.Weigh);
    }
}
```

By Dipendra Shrestha

Scanned with CamScanner

2) A Second Use for Super

- This Second form of Super is most applicable to Situations in which member names of a subclass hide members by the Same name in the superclass.

- This Usage has the following general form:

  Super. member

  Here, member Can either be a method or an instance Variable.

Example :

```
// Using Super to overcome name hiding.
Class A
{
    int i;
    int j;
}

// Create a subClass by extending Class A.
class B extends A
{
    int i;   // This hides the 'i' in Class 'A'.

    B(int a, int b, int c)
    {
        Super. i = a;   // 'i' in 'A'
        i = b;          // 'i' in 'B'.
        j = c;          // 'j' in 'A'
    }
```

Constructor हो।

यसले super Class मा भएको 'i' लाई चिनाउँछ।

Scanned with CamScanner

```
Void Show()
{
    System.out.println ("i in superclass :" + super.i);
    System.out.println ("i in subclass :" + i);
}


Class UseSuper
{
    public static Void main(String args[])
    {
        B obj = new B(1, 2, 3);

        Obj.Show();
    }
}
```

Output :

i in Superclass : 1
i in Subclass : 2.

By Dipendra Shrestha

Scanned with CamScanner

⊛ Method Overriding :

↱ Method overloading मा चाहिं Same method name but different return type, parameter different हुन सक्छ but,

In method Overriding, Same method name, Same return type & Same no. of parameter हुनपर्छ but त्यो method ले काम भने different गर्छ ।

Example :

SuperClass.Java

```
package practice_21;
public class SuperClass
{
        int getReturn()
        {
            return 0;
        }
}
```

यो method लाई तलको Sub class को method ले over ride गर्दिन्छ So, Sub class को method चल्छ ।

SubClass.Java

```
package practice_21;
public class SubClass extends SuperClass
{
        int getReturn()
        {
            return 5;
        }
}
```

Inherit गरेको छ ।

Scanned with CamScanner

Method Overriding.Java

```
package_practice_21;
public class MethodOverriding
{
    public static void main (String [] args)
    {
        SubClass sub_obj = new SubClass();
        System.out.println (sub_obj.getReturn());
    }
}
```

Output :

5

Scanned with CamScanner

⊛ Polymorphism :

- In polymorphism, objects of a class has ability to be in a different form.
- The word polymorphism means having many forms
- When we define object of class then quality of object to be in a different form is known as polymorphism.

Example :

Bank.Java (Base /Super class)

```
package practice_22;
public class Bank
{
    int getInterestRate ()
    {
        return 0;
    }
}
```

Bank_ABC.Java

```
package practice_22;
public Class Bank_ABC extends Bank
{
    int getInterestRate ( )
    {
        return 5;
    }
}
```

By Dipendra Shrestha

Scanned with CamScanner

Bank_DEF.Java

```
package practice_22;
public class Bank_DEF extends Bank
{
    int getInterestRate ()
    {
        return 10;
    }
}
```

Bank_XYZ.Java

```
package practice_22;
public Class Bank_XYZ extends Bank
{
    int getInterestRate()
    {
        return 15;
    }
}
```

By Dipendra Shrestha

Scanned with CamScanner

Polymorphism.Java

```
package practice_22;
public class Polymorphism
{
    public static void main (String[] args)
    {
        Bank bank = new Bank();
        Bank abc = new Bank_ABC (); // 'abc' refers Bank_ABC
        Bank def = new Bank_DEF (); // 'def' refers Bank_DEF
        Bank xyz = new Bank_XYZ (); // 'xyz' refers Bank_XYZ

        System.out.println (bank.getInterestRate());
        System.out.println (abc.getInterestRate());
        System.out.println (def.getInterestRate());
        System.out.println (xyz.getInterestRate());
    }
}
```

Bank भन्ने class को object हो but with different form

Output :-

0
5
10
15

By Dipendra Shrestha

Scanned with CamScanner

⊛ Abstract Class & Abstract Method :

⊙ Abstract Class :

- Abstract Class is a class which can't be instanciated.
- If we define abstract class then we can't make object of that class.
- To define abstract class we use 'abstract' keyword at the begining of declaration of the class.

Example :

Bank.Java (Super class)

```
package practice_23;
abstract class Bank
{
    int getInterestRate ()
    {
        return 0;
    }
}
```

Bank_ABC.Java (Sub class)

```
package practice-23;
public Class Bank_ABC extends Bank
{
    int getInterestRate ()
    {
        return 5;
    }
}
```


Scanned with CamScanner

AbstractClass.Java

```
public class AbstractClass
{
    public static void main (String[] args)
    {
        ✗ Bank bank = new Bank();
        ✓ Bank abc = new Bank_ABC();
        System.out.println (abc.getInterestRate());
    }
}
```

यसरी abstract class को object बनाउन पाउँदैन।

'bank' भन्ने object ले Bank() भन्ने abstract class लाई referencing गर्न पाउँदैन due to 'abstract' type class

// 'abc' भन्ने object ले Bank_ABC भन्ने class लाई reference गर्छ।

यो class को object ले कुनै पनि abstract class लाई reference गरेको हुन So, पाउँछ गर्न।

Scanned with CamScanner

Abstract Method :-

- जब कुनै पनि method लाई abstract define गर्नु पर्ने हुन्छ, तब abstract Class भित्र मात्र abstract method define गर्न मिल्छ।

- abstract keyword लेखेर कुनै पनि method लाई abstract बनाउन सकिन्छ।

- जब abstract method declare गरिन्छ then त्यसको definition part भने Super Class मा नभई त्यसको subclass मा गएर गरिन्छ।

Example :-

Bank.Java (Super class)

```
package practice_24;
abstract public class Bank
{
    abstract int getInterestRate();
}
```

त्यसको definition part subclass मा हुन्छ।

This is abstract method.

Bank_ABC.Java (Sub Class).

```
package practice_24;
public class Bank_ABC extends Bank
{
    int getInterestRate()
    {
        return 5;
    }
}
```

यो definition मा थी super class को abstract method को हो।

Scanned with CamScanner

AbstractMethod.Java :

```
package practice_24 ;
public class AbstractMethod
{
    public static void main(String[] args)
    {
        Bank abc = new Bank.ABC();

        System.out.println(abc.getInterestRate());
    }
}
```


Scanned with CamScanner

⊛ Package :

- Packages are containers for classes that are used to keep the class name space compartmentalized.

- Package allows us to create a class named List, which we can store in our own package without the concern that it will collide with some other class named List stored elsewhere.

- Packages are stored in a hierarchical manner and are explicitly imported into new class definitions.

- Package is both naming & visibility control mechanism.

- We can define classes inside a package that are not accessible by code outside that package.

- We can also define class members that are only exposed to other members of the same package.

- This allows the classes to have intimate knowledge of each other, but not expose that knowledge to the rest of the world.

By Dipendra Shrestha

Scanned with CamScanner

Defining a package :

Package pkg_name;

- Here, 'Package' is a keyword. & 'pkg_name' is the name of the package.

  Eg: Following statement creates a package called MyPackage.

  package MyPackage;

- In Java a hierarchy of packages can be created.

- To do so, simply we can separate each package name from the one above it by use of a period.

- The general form of multileveled package: pkg1[.pkg2[.pkg3]];

- A package hierarchy must be reflected in the file system of Java development system.

  For eg, a package declared as

  package java.awt.image;

  needs to be stored in Java\awt\image on the Windows file system.

By Dipendra Shrestha

Scanned with CamScanner

Importing Packages :-

- Java includes the 'import' statement to bring certain classes, or entire packages, into visibility.

- Once imported, a class can be referred to directly, using only its name.

- The general form of the import statement is :

```
import pkg1[.pkg2].(classname | *) ;
```

- Here, pkg1 is the name of a top-level package and pkg2 is the name of subordinate package inside the outer package separated by a dot (.)

- There is no practical limit on the depth of the package hierarchy, except that imposed by the file system.

- Finally we specify either an explicit class name or a star (*) which indicates that the Java compiler should import the entire package. This code fragment shows both forms in use :

```
import Java.Util.Date;
import Java.io.*;
```

By Dipendra Shrestha

Scanned with CamScanner

Eg :

```
import Java.util.Date
Class MyDate extends Date
{
    - - -
}
```

यसरी एउटा import भन्ने ले गर्दा 'Date' लाई access गर्न पायौं तर long date 'Date' भन्ने class को फेरी लेख्नि शरुमा पर्थ्यो So, झन्झट हटायो।

By Dipendra Shrestha

Scanned with CamScanner

(Can't be Instantiated)

(*) Interfaces in Java :-

⊙ What is Interfaces in Java?

↳ Interfaces are syntactically similar to classes, but they lack instance variables & their methods are declared without any body (without function definition).

Imp Note: Interface भनेको same as abstract class नै जस्तै हो जसको 'object' create गर्न सकिन्दैन & यो भित्र declare गरेको method हरू by default abstract method हुन्छ जसको declaration part त हुन्छ तर definition part भने subclass मा आएर हुन्छ।

⊙ Why we need Interfaces if we have Concept of abstract class?

↳ हामी generally 'class' हरूलाई inherit गर्दा 'extend' keyword use गरेर inherit गर्छौं तर Interfaces हरूलाई 'extend' गर्न मिल्दैन but 'implement' गर्छौं।

↳ Using interface, we can specify what a class must do, but not how it does it.

Imp ↳ Interfaces are designed to support dynamic method resolution at run time.

Generally यदि एउटा class ले अर्को class लाई call गर्दा both class चाहिं present हुनु पर्थ्यो Compile time मा तर Interface मा हामी interface लाई implement गर्छौं नाकी call.

Scanned with CamScanner

यदि Interface लाई नै 'public' भनेर declare गरेको छ भने त्यसमा भएको हरेक method र variable पनि by default public हुन्छ।

⊙ Defining an Interface

↱ An interface is defined much like a class.

↱ The general form of an interface :

```
access interface name
{
    return-type method-name1 (parameter-list);
    return-type method-name2 (parameter-list);
    type final-varname1 = value;
    type final-varname2 = value;
    :
    return-type method-name N (parameter_list);
    type final-varname N = value;
}
```

(keyword: interface; name → (name of Interface))

Public हुन सक्छ or खाली नराख्दा पनि हुन्छ but राख्ने भने त्यही package भित्र भएकोले मात्र access गर्न पाउँछ।

Note:

1) Method defined within Interface doesn't have body. (i.e definition part)
2) By default method declared within Interface are of abstract type.
3) यदि कुनै class ले Interface लाई 'Implement' गरेको छ भने Interface मा declare भएको हरेक method लाई definition (body) provide गरेकै हुनु पर्छ।
4) Variables declared inside interface are implicitly final and static meaning they can't be changed by implementing class.

By Dipendra Shrestha

Scanned with CamScanner

Note :-

When to use 'extends' and when to use 'implements'.

SuperClass	SubClass	Keyword to use.
Class	Class	extends
Interfaces	Class	implements
Interfaces.	Interfaces	extends

Example of Implementing Interface

Bank.Java

```
package practice_25;
public interface Bank
{
    int getInterestRate();
}
```

Interface हो So automatically abstract type हुन्छ त्यसको object बनाउन सकिन्न।

→ यहाँ interface मा त्यो पनि method हुन त्यो सबैको definition त्यहाँ Subclass मा हुन पर्छ।

Bank_ABC.Java

```
package practice_25;
public class Ban_ABC implements Bank
{
    public int getInterestRate()
    {
        return 5;
    }
}
```

by nature public हुनुँ पर्छ यो Interface को method लाई definition (body) दिँदा

Scanned with CamScanner

Implementing Interfaces.Java

```
Package practice_25;
public Class ImplementingInterfaces
{
    public static Void main (String [] args)
    {
        Bank abc = new Bank_ABC ();

        System.out.println (abc.getInterestRate ());
    }
}
```

By Dipendra Shrestha

Scanned with CamScanner

⊛ Arraylist In Java :

→ Arraylist dynamic array हो in Java.
→ 'Util' भन्ने Library मा define गरेको छ Arraylist लाई।

→ To use Arraylist we import 'Util' Library as :

```
import Java.util.Arraylist;
```

Example : Arraylist.Java

```
package practice_26;
import Java.util.ArrayList;

Public class arrayList
{
    public static void main (String[] args)
    {
        ArrayList <Integer> number = new ArrayList
                                     <Integer> (5);
        number.add(1);
        number.add(2);
        number.add(3);
        number.add(4);
        number.add(5);
        number.add(6);

        for (Integer x: number)
            system.out.println (x);
```

(Array को नाम)

Array को initial size हो। यो array मा पाँच वटा मात्र value राख्न मिल्छ भनेको चाहि होइन। It's Just initialCapacity हो।

Scanned with CamScanner

// for calculating Size of Array.

```
System.out.println ("Array Size = " + number.size());
```

// To remove value from array use 'remove' method;

```
number.remove (2);
```

↳ Index No. '2' मा भएको Array को value remove हुन्छ।

```
for (Integer x : number)
    System.out.println (x);
```

// To interchange the value of Array index.

```
number.Set (0, 100);
```

↳ 'zero' index को value लाई 100 ले replace गर्दिने।

// To clear all the value of Array we use 'Clear' method.

```
number.clear ();
```

```
    }
}
```

By Dipendra Shrestha

Scanned with CamScanner

⊛ Exception Handling :

- Exception is an abnormal condition that arises in a code sequence at run time.

- An exception is a run time error.

- Java exception handling is managed via five keywords : try, catch, throw, throws and finally.

  - program statements that we want to monitor for exceptions are contained within a try block.
  - If an exception occurs within the try block, it is thrown.

  - To manually throw an exception we use the keyword throw.

  - Any exception that is thrown out of a method must be specified as such by a throws clause.

  - Any code that absolutely must be executed before a method returns is put in finally block.

By Dipendra Shrestha

Scanned with CamScanner

General form of an exception handling

```
try {
    // block of Code to monitor for errors.
}
Catch ( Exception Type1 exOb)
{
    // exception handler for Exception Type1
}
:

finally{
    //block of Code to be executed before
    try block ends.
}
```


Scanned with CamScanner

Example :

```
package practice_27;

public class exceptionHandling
{
    public static void main (String[] args)
    {
        try
        {
            int a = 10/0;
            int b[] = new int[2];
            System.out.println (b[3]);
        }

        Catch (Arithmetic Exception e)
        {
            System.out.println (e);
        }

        Catch(Index Out Of Bounds Exception e)
        {
            System.out.println (e);
        }

        Catch (Exception e)   // General type of Exception लाई Catch गर्ने।
        {
            System.out.println (e);
        }
```

By Dipendra Shrestha

Scanned with CamScanner

```
finally        // यो चाहिं at last मा exception हरु handle भएर
                पछि चल्ने block हो ।.
  {
    System.out.println("-------");
  }
 }
}
```

output :

```
Java.Lang.ArithmeticException : / by zero
--------------
```

By Dipendra Shrestha

Scanned with CamScanner