Static:

When u access static variable with the reference will get waraning maessage ,wanraing message not hold program we can run the program

```
public class A {
	 static int c=12;
	public static void main(String[] args) {

		A a=new A();
		System.out.println(c.a);

	}

}
```

Non static : non static variable are deleare outside all the method but then inside the class with out static keyword

Non static variable can be access only after object creation

It is non mandatory to inilization non satic varible if we don't automatic get default value by compiler

Class A

{ int i=10;

Int j;// gets 0

Psvm()

{

A a=new A()

Sop(a.i) //

Sop(A.i)// error

}

}

Reference variable : Local and static

Rv are used to store memory address ,it can never store aodinary variable

Logal rv:

They are create with in a method and should be used only with created method

 ex

```
public class A12 {
	int i=67;

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	A12 a=new A12();
```

```
		System.out.println(a.i);
		a.test();



		}
		public void test() {
			System.out.println(a.i);// error
		}

}
```

Static referen v:

This variable are created outside all the method but then inside the class using static keyword

This variable can be used any where in the class as they have gobal access

Expel

```
public class A12 {
		static A12 a;

		public static void main(String[] args) {
			// TODO Auto-generated method stub
		a=new A12();
		System.out.println(a.i);
		a.test();



		}
		public void test() {
			System.out.println(a.i);
		}
```

Data types in java

Data type	Memory	Defulat	
Byte	1	0	
Short	2	0	
Int	4	0	
Long	8	0	
float	4	0.0	I=12.0f
Double	8	0.0	
boolean	na	flase	
char	2	blank	
string	na	Null	

Type casting : converting a particular datatype into require data type its call type casting

Auto upcasting :

Converting smaller datatype into bigger data type with out loss any of data its call auto upcasting

Int i=10;

Long j=I;

If long i=12;

Int j=I;// error

Float i=12.0f int j=

Long j=I;// error because only used integer

Explicit down casting:

Converting digger data type into smaller datatype its call down casting

During convertion if any loss of data then reguerless of memory side we need to perform explicit

Int j=10;

Byte k=(byte)j;

Long k=0;

Byte j=(byte)(int)k// 0

Unary operator

Method will execute only when call it

Method will return the control back to place from where it is call

```
public class A6 {

	public static void main(String[] args) {
		A6 a=new A6();
		a.test();
		System.out.println("from main ");
```

```
		// TODO Auto-generated method stub

	}
	public void test() {
		System.out.println("fromtest");
	}

}
```

Return keyword

Returns control of the method back to place where it call when we are using only return keyword make sure that method is of the type void

Return keyword should be the last statement inside the method

If there are some statement immmedately after return keyword then those statement will never executed and hands we get error

```
public class A9 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		A9 a = new A9();
		a.test();
		// TODO Auto-generated method stub

	}
public void test() {
	System.out.println("text");
	return ;// by default
	System.out.println("text2");// error

	}
}
```

Constructor : it should have same name at that class

It will executed when a object created

```
public class A {
	A(){
		System.out.println("A");
	}

	public static void main(String[] args) {
		new A();

	}

}
```

Next program

```
public class A {
	A(){
		System.out.println("A");
	}

	public static void main(String[] args) {
		new A();//A
```

```
		new A();//A
		A a =new A();//A
	}

}
```

Constructor value insert

```
public class A {
	A(int x){
		System.out.println(x);// first print 122
	}

	public static void main(String[] args) {

		A a =new A(122); // it call first constructor
		System.out.println("main "); //second main
	}

}
```

multiple Constructor we can create multiple ctructor in the same class provided they defference base on name of arg and type of arg

```
public class A {
	A(int x){
		System.out.println(x); first this one print 122
	}

	A()
	{
		System.out.println("a"); second a print
	}
	public static void main(String[] args) {

		A a =new A(122);
		A a1 =new A();
		System.out.println("main "); third  main
	}

}
```

Constructor inside object create

```
public class A1 {
	A1(){
		A1 a=new A1(122); // it call a1(int x) constructor
		System.out.println("A1");//second A1 print

	}
	A1(int x){
		System.out.println(x);// first print 122
	}
	public static void main(String[] args) {
		A1 a1=new A1(); // it first go A1 constructor
		System.out.println("main");

	}

}
```

Constructor vs method

Constructor always void but method can be void or return type

The retrun type of constructor it always void and hands we can used only return keyword but we cant retrun with value
Method can have same name as that of class and constructor can never Have return type including void

Together used method and constructor

```
public class A2 {
	A2(){
		System.out.println("aa");

	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
A2 a =new A2();
a.text();

	}
public void text()
{
	System.out.println("from text ");
}
}
```

Method can have same name as that of class and constructor can never Have return type including void

Example public class A5 {

```
	public int A5()	// method
	{System.out.println("method");
		return 30;
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
A5 a =new A5();
System.out.println("main");

}

}
```

Constructor first call then method

```
public class A5 {
	void  A5()	// method
	{System.out.println("method"); //second print
		//return 30;
	}
	A5(){
		System.out.println("contructor"); // first cons print
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
A5 a =new A5();
a.A5();
System.out.println("main");
```

}}

IIB(instand initilazation block)

it are exectude when object are create
No of time we create obj same no of time iib call
 It used initizatin all the instance value one place
That are gives us reabilitity code

```
public class A2 {

	{
	System.out.println(" form iib");
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
A2 a =new A2();// call iib
	}

}
```

Constructor and iib both are together then first call iib then constructor

```
public class A2 {

	{
	System.out.println(" form iib");// first print
	}
	A2()
	{
		System.out.println("aa");// second print
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
A2 a =new A2();
Sys.pln("main")// third print
//A2 a1 =new A2();
	}

}
```

 We can initialize both static and no satic variable inside IIb

```
public class Iib {
 int i;
 static int j;
 {
	 i=90;
	 j=23;
	 System.out.println(i);
	 System.out.println(j);
 }
	public static void main(String[] args) {
		Iib a=new Iib();


	}

}
```

SIB(static initialize block)

Static run only one time
Static runs before main method it doesnot required any invoking statement

Static only accept static variable
We can initialize nonstatic variable inside static

```
public class A3 {
	static {
	System.out.println("from sib");// before main means it first the main class
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub

	}

}
```

When more static ib then ,they are run sequence

```
public class A3 {
	static int i;
	static {
		i=10;
		System.out.println(i);// first
	}
	static {
		i=23;
		System.out.println(i); // second
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("from main "); third

	}
```

Static >iib>constructor>main

```
public class A5 {

	{
System.out.println("iib "); // second

}
static {
	System.out.println("sib ");//  first

}
A5()
{
	System.out.println("constructor ");//third
}

public static void main(String[] args) {
	// TODO Auto-generated method stub
A5 a =new A5();
System.out.println("main"); // four
}
}
```

We can create an obj inside sib example

```
public class A7 {
	{
		System.out.println("iib"); // first run
	}
	static
```

```
	{
		new A7();
		System.out.println("sib");// third
	}
	A7()
	{
		System.out.println("constructor"); // second
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	  System.out.println("main ");// last
	}

}
```

This keyword :

This keyword point to the current obj executing
 This keyword cant be used inside static context
using this keyword can access static and non static members
this keyword cant access logal variable
using this keyword we can call constructor of same class but then to do this keyword should be very first statement inside another constructor

```
public class A {
	int i=10;
	public static void main(String[] args) {
		A a =new A();
		System.out.println(a.i);// 10
		a.test();

	}
	 public void test()
	 {
		 System.out.println(this.i);//10
	 }

}
```

This word

```
public class B {
	int i=10;
	public static void main(String[] args) {
		B a =new B();
		B a2=new B();
		System.out.println(a);// address
		System.out.println(a2);
		a2.test();

	}
	 public void test()
	 {
		 System.out.println(this);// current object new a2 adrees
	 }

}
```

This keyword cant be used inside static context

```
public class C {
	int i=10;
	public static void main(String[] args) {
```

```
		B a =new B();

		System.out.println(a);// address
		System.out.println(a2);
		a2.test();

	}
	 public static void test()
	 {
		 System.out.println(this.i);// this  not used static method
	 }


}

}
 using this keyword  can access static and non static numbers
public class A1 {
	 int i=10;
	 static int j=20;
	 public static void main(String[] args) {
		A1 a =new A1();

		a.test();

	}
	 public void test()
	 {
		 System.out.println(this.i);
		 System.out.println(this.j);
	 }



}
Not access local variable
public class A4 {
	public static void main(String[] args) {
		A4 a =new A4();

		a.test();

	}
	 public void test()
	 {
		int k =30;
		System.out.println(this.k);// cant access local variable

	 }
}
using this keyword we can call constructor of same class
public class A8 {

	public A8() {
		// TODO Auto-generated constructor stub
		System.out.println("Aaa");// first

	}
	A8(int i)
```

```
	{
		this();
		System.out.println(i);//122 second
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		A8 a =new A8(122);

	}

}
```

Super keyword:

1)Super keyword we can access the numbers of parent class
2)Using super keyword we can access satic and and non static members
3)super keyword cant used inside static method
4) using super keyword we can call conntructor of parent class but then we should used super keyword child class constructor it should be very first statement
5)if we don’t keep super keyword inside child class contructor then compliler will automatic place super keyword such that it can call only no arg constructor of present class
6)IF we don’t create child class constructor with out arg then compiler will automatily place no arg constructor with super keyword
7)If in a parent class this is only constructor with arg then as a programmer we should explicitly write super keyword in child class constructor

```
class A {
int i=10;

}
class B extends A{


	public static void main(String[] args) {
		// TODO Auto-generated method stub
B a =new B();
a.test();
	}
	public void test()
	{
		System.out.println(super.i);//10
	}

}
```

```
#2 program
class A
{
	public void Xyz() {
		System.out.println("xyz");//3
	}
}

public class A3 extends A {

	public static void main(String[] args) {
		A3 a =new A3();//1
		a.test();


	} public void test() {
		super.Xyz();//2
	}

}
#33 program
class CC
 {
	static int j=12;
}

public class A2 extends CC{
	public static void main(String[] args)
	{


	A2 a =new A2();
	a.test();
	}
	public void test() {
		System.out.println(super.j);
	}


}
Inside static  method  cant used super keyword
class CC
 {
	static int j=12;
}

public class A2 extends CC{
	public static void main(String[] args)
	{


	A2 a =new A2();
	a.test();
	}
	public static void test() {
		System.out.println(super.j);// error/
	}
```

```
}
444 program
class A4
{
	A4(){
		System.out.println("aaa");
	}
}

public class A5  extends A4{
	 A5()
	 {
		 super();
	 }
public static void main(String[] args) {
	new A5();

}
}
5)program
class A4
{
	A4(){
		System.out.println("aaa");
	}
}

public class A5  extends A4{
	 A5()
	 {

		 System.out.println("df");
	 }
public static void main(String[] args) {
	new A5();
}
}
6)program
class A4
{
	A4(){
		System.out.println("aaa");
	}
}

public class A5  extends A4{
	*/ A5()
	{
		super();
	}/*

public static void main(String[] args) {
	new A5();
}
}
7)program
class Aa4{
	Aa4(int i)
	{System.out.println(i);
```

```
		//System.out.println("b2");
	}
}

public class B1 extends Aa4{
	B1(){
		super(100);

	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
new B1();
	}

}
```

Inheritance

1)Here non static member parent class are inherited to the child class object so that we can reused the member of parent class

2)Static member do not get inheritance but then give of inheritance by converting statement

1)program

```
class B
{
	int i=20;
	static int j=16;
}
public class A extends B{

	public static void main(String[] args) {
		// TODO Auto-generated method stub
A a=new A();
System.out.println(a.i);
	}

}
```

2)program

```
public class A3
{
static int i=20;

}

public class A44 extends A3 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println(A44.i);
```

```
} }
2program
class A11

{
	public void test1()
	{
		System.out.println("tesr 1");

	}
	public static void test2()
	{
		System.out.println("test 2");
	}
}
public class AA extends A11 {
	public static void main(String[] args) {
		AA a=new AA();
		a.test1();// inheritance
		a.test2();// converting  statement means A11.i create
	}

}
Multilevel inheritance
class A122
{
	int i=10;

}
class B22 extends A122
{

}

public class Mul extends B22{

	public static void main(String[] args) {
		// TODO Auto-generated method stub
Mul a =new Mul();
System.out.println(a.i);
	}

}
```

Packages

Packages are folder structure create java to store the program in hand origenization manner

```
package p1;

public class A1 {
	int i=100;

}
package p1;
import p11.A11;// all are defferen package so import used
public class B {
	public static void main(String[] args) {
```

```
		A1 a=new A1();// same package we can call directly
		System.out.println(a.i);
		A11  a1=new A11();
		System.out.println(a1.i);
	}

}
package p11;

public class A11 {

public int j=102;
}
2 nd program
package app1.app2.app3;

public class D {
	int k;

}
package p1;

import app1.app2.app3.D;

public class B {
	public static void main(String[] args) {
	D d1=  new D();
	}


}
3 program
package p11;

public class A11 {

public int j=102;
}
package p1;


public class B {
	public static void main(String[] args) {
	p11.A11 c=new p11.A11();
	System.out.println(c.j);
	}
	}
4 program
package p11;

public class A11 {

public int j=102;
public void test()
{
	System.out.println("form test A11 class ");
}
}
package p1;
```

```
public class B {
	public static void main(String[] args) {
	//p11.A11 c=new p11.A11();
		new p11.A11().test();
	System.out.println(new p11.A11().j);
	}


}
```

Import static variable and static method

if want import static nmber in your class then we need to used static input

```
package p11;

public class A11 {

public static int j=102;
public void test()
{
	System.out.println("form test A11 class ");
}
}
```

```
package p1;

import static p11.A11.j;
public class B {
	public static void main(String[] args) {

	System.out.println(j);
	}


}
```

We can never import non static member in java

Access specifier

	default	protected	Public	Private
Same class	Yes	Yes	Yes	yes
Same package	Yes	yes	Yes	no
Same package non sub class	yes	Yes	yes	no
def pack sub class	no	Yes	yes	no

Private

```
package accesss;

public class A {
	private int i=10; // private value access only this class
	private void test() {
		System.out.println("test");
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		A a=new A();
		System.out.println(a.i);
		a.test();

	}
}
```

polymorphism

developing the feature such that which can take more then one form it call polymorphism

overriding :

after inheriting a method from parent class if want the modify that inherited method then we used concept overriding

inheritance is mandatory for overriding

overriding happened only in method not for variable

static member not overriding

while overriding if we are increasing scope of accessspicifier then it will not give any error

private>default>protected >public

example 1

```
package polymorphim1;

class B
{
	public void test()
	{
		System.out.println("A");
	}


}

public class A {
```

```
public void test ()

	{
		System.out.println("b");// op
	}
	public static void main(String[] args) {
A a=new A();
a.test();
	}
}
Ex
public class Silverac {
public void chqbook() {
	System.out.println("2 book ");

}
public void phbank()
{
	System.out.println("all time get many");

}
public void atm()
{
	System.out.println("any where get many ");
}
}

public class Goldact extends Silverac {
	public void chqbook() {
		System.out.println("unlimited  ");

	}

	public static void main(String[] args) {

			Goldact a=new Goldact();
			a.chqbook();
			a.atm();
			a.phbank();
		}

	}
Overloading:


Here we create multiple method with same name then they are deff base on member
and type  of arg

public class D {
	public  void test(int a)
	 {
		System.out.println(a);

	 }
	public  void test(int a,int b)
	 {
		System.out.println(a);
		System.out.println(b);
```

```
	}
public static void main(String[] args) {
	D a=new D();
	a.test(70);

}

}
```

Exception and Exception handling

When a bed user input given the program hold atraply .holding a program atraply it call as expecption

In java to handle the exception we used try catch block

Whenever ex happened inside try block ,try will create and execption obj and reference of the obj it will give u catch .catch is now will handle the exception and hands program not hold atraply

Try create the exception and catch handling exception

```
package polymorphim1;

public class AA1 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
int i=10/0;
System.out.println(i);
System.out.println("complete");
	}

}
```

Try and catch

```
public class Ram {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		try {


		int i=10/0;
		}
		catch (Exception e)
		{
			System.out.println(e);
		}
		System.out.println("completed ");

	}
}
```

Throws able

It two type
Error and exception type of exception
Runtime
Arithmetic ex:

```
Try {
10/0;// arth ex
```

```
}
Catch(Exception e){
Sop(e);
}
```

null pointer:

it we accessing member of the class with null reference then we get null pointer exception

```
public class B {
	static B b1;
	int i=10;
	public static void main(String[] args) {
		System.out.println(b1.i);// null pointer ex
	}

}
```

Exception handling

```
package exception12;

public class B {
	static B b1;
	int i=10;
	public static void main(String[] args) {
		b1=new B();
		System.out.println(b1.i);// null pointer exception

	}

}
```

2 program

```
public class C {
	static C c1=null;
	int i=10;

	public C(C c2) {
		try {
		// TODO Auto-generated constructor stub
		System.out.println(c1.i);
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
c1=new C(c1);
	}

}
```

Number format : an invalid conversion of a string into a number leets to number format exception

Array heps us data using similar kind array users memory which is continuous and hands from memory point of view that efficient

```
public class A {

	public static void main(String[] args) {
String s="test";
```

```
		int i=Integer.parseInt(s);// number format exception
		System.out.println(i);
}
}
```

Try catch

```
package numberformat12;

public class A {

	public static void main(String[] args) {
		String s="test";
		try {


		int i=Integer.parseInt(s);// number format exception
		System.out.println(i);
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
		System.out.println("completed");
	}



}
```

3 rd program

```
package numberformat12;

public class A {

	public static void main(String[] args) {
		String s="233";
		try {


		int i=Integer.parseInt(s);// number format exception
		System.out.println(i);
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
		System.out.println("completed");
	}



} 4 program
package numberformat12;

public class B {

	public static void main(String[] args) {
		int a[]=new int[3];
		a[0]=10;
		a[1]=12;
		a[2]=14;
```

```
		a[3]=23;
		System.out.println(a[1]);
		System.out.println(a[0]);
		System.out.println(a[2]);
		System.out.println(a[3]);// out of bound array that is numberformat
ex
	}

}
```

If we want to store hetrogenis data in an array then create array of type object

If a create a variable object it means that we can store any kind of data init

If you spcifiy size of array then idialy it is satic array .

If want to memory dynamicly increase and decrease then you create dynamic array

```
package numberformat12;

public class C {
	public static void main(String[] args) {
		Object[] d =new Object[3];
		d[0]="test";
		d[1]=22;
		d[2]=12.9;
		for (int i=0;i<3;i++)
		{
			System.out.println(d[i]);
		}

	}

}
```

Array

Array method

```
package array;

public class A1 {
	public int[] test()
	{
		int [] a= new int[3];
		a[0]=12;
		a[1]=23;
		a[2]=33;
		return a;
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
A1 a=new A1();
int[] b=a.test();
for(int x:b)
	System.out.println(x);
	}

}
```

Double index array

```
package array;

public class A2 {
```

```
	public static void main(String[] args) {
		// TODO Auto-generated method stub
int[][] a=new int[2][3];
a[0][0]=10;
a[0][1]=10;
a[0][2]=10;
a[1][1]=10;
a[1][2]=10;

	System.out.println(a.length);//row
	System.out.println(a[0].length);// column

	}

}
 Compile time /straight away
File exp:fileInputStream read the file

import java.io.FileInputStream;

///import javax.annotation.processing.SupportedSourceVersion;

public class A {
	public static void main(String[] args) {
		try {
		FileInputStream f=new FileInputStream("D://testing.txt");
for(int i=0;i<3;i++)
				System.out.println((char)f.read());
	}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
}
```

Input and output file

File file class we help us to count number of character in a given file where as filereader will help us to read contain thee file

File class in the below program to help us to build for loop dynamically

Create defferend files using file class

```
ackage file;

import java.io.File;

public class B {
	public static void main(String[] args) {
		try {
		File f=new File("D://test2.html");
		f.createNewFile();
		File f1=new File("D://test2.png");
		f1.createNewFile();

		}
		catch(Exception e){
```

```
                System.out.println(e);
            }

}

}
```

```
package file;
import java.io.File;
import java.io.FileReader;

public class A7 {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		try {
			File f1=new File("D://test2.txt");


			FileReader f=new FileReader(f1);
			 for(int i=0;i<f1.length();i++) {
			System.out.print((char)f.read());
		}
			}
			catch(Exception e){
				System.out.println(e);
			}

}
}
```

File reader

File reader class used to read the contain

File writer :

File writer it used to write the contain

```
package file;

import java.io.File;
import java.io.FileWriter;

public class A8 {
	public static void main(String[] args) {


	// TODO Auto-generated method stub
	try {
		File f1=new File("D://test2.txt");


		FileWriter w=new FileWriter(f1);
		w.write("abc");
		w.close();
		System.out.println(f1.length());

		}
```

```
		catch(Exception e){
			System.out.println(e);
		}

}
}
```

Bufferreader:we can not used file class with bufferedreader/writter .its only work with filereader and filewriter. Bufferedreader is used to increase the performance can also read the data line by line

Buffer writer

```
package file;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;

public class A9 {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		try {
			FileWriter f1 =new FileWriter("D://test2.txt");
		BufferedWriter w=new BufferedWriter(f1);
		w.write("hello");
		w.newLine();
		w.write("hello2");
		w.newLine();
		w.close();

		}

			catch(Exception e){
				System.out.println(e);
			}

	}

}
```

FileInputStream

FileOutputStream: file output stream request the con tain tobe convert into number using getbyte method where as file Weiter can writer directly file contain

Finally/final finalize

Finally is extraction of try and catch block .anything captian will executed regureless off exception

Used finally very open to perform close operation

```
package finally1;

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

```
	try
	{
		int i=10/0;
		System.out.println(i);
	}
	catch(Exception e)
	{
		System.out.println(e);
	}
	finally
	{
		System.out.println(" i am bad boy");
	}
}
}
```

With out catch block using try

```
package finally1;

public class A2 {
	public static void main(String[] args) {
		try
		{
		System.out.println("hello");
		}
		finally
		{System.out.println("bolo");
		}
	}
}
```

Catch only handle exception

Here no available catch block

```
ackage finally1;

public class A2 {
	public static void main(String[] args) {
		try
		{ int i=10/0;
		System.out.println(i);
			System.out.println("hello");
		}
		finally
		{
			System.out.println("bolo");
		}
		System.out.println("dsfgahjk");// not print because catch no there
	}

}
```

Final

Variable final: the value store in final first time .final variable value can not alter

Wheater we change value or same value not possible and not increment and decrement

If we make a variable as final then we can never re initializition not possible

```
package final11;
```

```java
public class A {

	public static void main(String[] args) {
	final int i=90;

	i=23;// not alter // error given here not run program
	System.out.println(i);
	}

}
```

Non static variable and static variable if made final then initialization is mandatory or else we well get blank filed error

```java
package final11;

public class A {
final int i;// blank  filed error
final static  int j;// error
	public static void main(String[] args) {
	System.out.println(i);
	}

}
```

Local variable with out initialization not error when we print get error

```java
package final11;

public class A {

	public static void main(String[] args) {
		final int j;
	System.out.println(i);// error
	}

}
```

If method final arg given then it also given error

```java
package final11;

public class A1 {
	public static void main(String[] args) {
		A a=new A();
		a.test(29);
	}
public void test(final int i) // hare int i is local so not charge value given error
{
	i=30;// error
	System.out.println(i);
}
}
```

Final value can alter

```java
package final11;

public class A2 {
public static void main(String[] args) {
	final int i=89;
	int j=i;
	System.out.println(j);//10
}
```

```
}
```
If we make array is final then its size can not be alter but array value can be alter
```
package final11;

public class A3 {

	public static void main(String[] args) {
		final int[] a=new int[3];// final it make array size can not alter but value alter
		a[0]=12;
		a[0]=13;
		System.out.println(a[0]);// 13

	}
}
```
Main class string[] args
The perpose of string args is supply command line argument by the programmer
String[] args help us to supply comman line argement
Supply camonad line argument click on drop dwon run then select run configuration go to argument and give arg suparated by space as soon below
And click and apply close
And main method write the follow code
```
package final11;

public class A4 {
	public static void main(String[] args) {
		System.out.println(args[0]);
	}
}
```
If array index bound error if given value 2 and arge want 3 value

---

If make string args make final then it sise can not be change but value can be change
```
package final11;

public class A5 {
	public static void main( final String[] args) {
		args = new String[3];// error given

	}
}
```
If an create
```
package final11;

public class A5 {
	public static void main( String[] args) {
		args = new String[3];
		args[0]="12";
	}

}
```
Dynamic array is not possible in agrs array initialization
```
package final11;

public class A5 {
	public static void main( String[] args) {
		args[] = {"ota","tyu"}

	}
```

```
}
```
Make a class as final then it value can never been inheritance
```
package final11;
final class A
{
	int i =10;
}

public class B extends A{

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		B a=new B();
		System.out.println(a.i);// error
		}
}
```
Final class can inherited member of non final class
```
package final11;
 class B22
{
	int i =10;
}
final class B extends B22{

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		B a=new B();
		System.out.println(a.i);// error
		}
}
```
Non static variable in parent class given final value with out initialization
We get error
```
package final11;
class A11
{
final int i; // error
}

public class A6 extends A11 {
	public static void main(String[] args) {
		A6 a= new A6();
		System.out.println();
	}

}
```
non static varible in parent class final initialization then it call and it alter vale it not change
```
package final11;
class A11
{
final int i=10;
}

public class A6 extends A11 {
	public static void main(String[] args) {
		A6 a= new A6();
		a.i=40;// not alter
		System.out.println(a.i);
	}
```

```
}
 One we make a methos as final we can never override that method

package final11;
class Aa
{
	final public void test() {
		System.out.println("test final ");
	}
}

public class B1 extends Aa {
	public void test()// error  we can not override
	{
		System.out.println("test parent ");
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub


	}
}
```

Finalize

Finalize is method in java

when many object are created they are not used then jvm automatic call java garbage collector to clean up the unused objcct. But is very difficult to predict when garbage collector will be call

a programmer can call finalize method using the statement system.gc()

when we call finalize method it is a request but when jvm call its comment

```
package finalize1;

public class A {
	protected void finalize() {
		System.out.println("from finalize ");
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
A a=new A();
a=null;
System.gc();// user call finalize method
	}

}
```

Final vs finalize vs finally

Throw and Throws

Throws keyword is always return down in front of method we cant used this in front class or variable
Throws keyword throw exception to the calling statement of the method
Weather exception are happen in method or not sounding calling statement in try catch become mandatory if throw keyword used
Example

```
package throwandthrows;
 class A1 {
public static void test() throws Exception
{
	int i=10/0;
	System.out.println("qwe");

}
}
public class A
{
	public static void main(String[] args) {
		A1 a =new A1();

	try
		{
			a.test();

		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
}
```

Multiple exception
Jar
Jar is cloccetion of dot class file and interface
Step to generate jar file
Right click on project select export then jar file click on next and finish
Linking the jar into your curent program right click on project go to properties java build path library tap click on external jar
Reflection : it is helps us to analysis the member of class develop by some one else
The major draw back reflection is it redused ed us sequrity of the program

```
package reflection1;
```

```java
import java.lang.reflect.Constructor;
//import java.lang.reflect.Method;

public class Expleref {

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


	Class cls =  Class.forName("reflection12.A");
	Constructor[] m=cls.getConstructors();
	System.out.println(m[0]);
		}
		catch(Exception e) {
			System.out.println(e);
		}


	}

}
```

2program

```java
package reflection12;

public class B {

	public B() {
		// TODO Auto-generated constructor stub
		System.out.println("conss");
	}



	public void test()
	{
		System.out.println("test");
	}

}
```

```java
package reflection1;

import java.lang.reflect.Constructor;
import java.lang.reflect.Method;

public class Expleref {

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


	Class cls =  Class.forName("reflection12.B");
	Constructor[] m=cls.getConstructors();
	Method[] j=cls.getDeclaredMethods();
	System.out.println(m[0]);
	System.out.println(j[0]);
		}
		catch(Exception e) {
			System.out.println(e);
		}
```

```
	}

}
```

Class casting

Upcasting :
Here child obj memory address is store in parent class reference variable
In upcasting only parent class members are access avalible and not child class member number

```
package upcasting;
class B
{
	int i=10;
}

public class A extends B{

int j=10;
	public static void main(String[] args) {
	 B a=new A();
	 System.out.println(a.i);
	 System.out.println(a.j);// erro not access child class member

	}

}
```

2program

```
package upcasting;
class C1
{

	public void test() {
		System.out.println("test");
	}
}

public class A2 extends  C1{

public void xyz()
{
	System.out.println("xyz");
}
	public static void main(String[] args) {
	 C1 a=new A2();
	 a.test();
	 a.xyz();// error
}
}
```

3rd program

```
package upcasting;
class C1
{

	public void test() {
		System.out.println("test");
```

```
	}
}

public class A2 extends  C1{

public void xyz()
{
	System.out.println("xyz");
}


public class A3  extends A2{
	public void test2()
	{
		System.out.println("tesrt22");
	}

public static void main(String[] args) {
	A2 a1=new A3();
	a1.test();
	a1.xyz();
	a1.test2();// error
}

}
```

Static method in parent class access and give waring bt it run and give out put

```
class D
{

	public static void test() {
		System.out.println("test");
	}
}


public class A4 extends  D{

public static  void xyz()
{
	System.out.println("xyz");
}
public static void main(String[] args) {
	D a =new A4();
	a.test();// waring bt it run for satic method
	//a.xyz();//error
}
}
```

In upcasting overring method will be call and not overridden

```
package upcasting;

class F
{

	public void test() {
		System.out.println("test");
	}
}
```

```java
public class A5 extends  F{

public  void test()
{
	System.out.println("xyz A5");
}
public static void main(String[] args) {
	F a =new A5();
	a.test();// xyz A5 op
}
}
```

Downcosting
Storing memory address of parent class into child class reference variable it call as down casting
No compile time error will give run time Exception
**To perform downcasting we will firdtly create object of parent class
Then create object child class
Then do upcasting
And then down casting as soon in the program bellow

```java
package downcasting;
class B
{
int i=40;
}

public class A  extends B{

int j=23;
public static void main(String[] args) {
	A a =(A) new B();

	System.out.println(a.j);// not given any compile time error run time exception will give
	System.out.println(a.i);// both are
}
}
```

**2prgram

```java
package downcasting;
class B
{
int i=40;
}

public class A  extends B{

int j=23;
public static void main(String[] args) {

B a =new B();// parent class opbjeet create
A a1=new A();// child class object create

a=a1; // upcasting
a1=(A)a;// down casting


	System.out.println(a1.j);
```

```
	System.out.println(a1.i);
}
}
```

Interface

Interface are 100% abstract or 100% incompleted

Interfaces just a like contract what class gets into interfaces and the class should follow the contract

That is implements method inherited from an interface should be completed in a class .if don't completed method in the class then we will get error

We inherited member of interface which is incomplete and then we overrided with the completed method subclass

 abstract is keyword in java which is specific incomplete method

In an interface every method default abstract an hands do not need to define

```
Interface A
{

Public void test();
Public void Xyz()
}
}
```

1 program

```
package interface11;
public interface B {
public  void test();
}
class A implements B{
	public void test()
	{
		System.out.println("from testt ");// op
	}
	public static void main(String[] args) {
		A a=new A();
		a.test();
}
}
```

2program

```
package interface11;
public interface B {
public abstract void test();
public void xyz();
}

public class C implements B{
public  void test()
{
	System.out.println("test ");
}

public void xyz()
{
	System.out.println("form xyz");
}
public static void main(String[] args) {
	C a=new C();
	a.test();
	a.xyz();
```

```
}
}
```
Folder are defferent there two package p1 and p2
When went interface of A u must import p1.A;
In to p2 package

```
package p1;

public interface  A {
public void test();

}
package p2;
import p1.A;
public class C implements A  {
public void test()
{
	System.out.println("test ");
}
public static void main(String[] args) {
	C a=new C();
	a.test();
}
}
```
If it default method

```
package p1;

public interface  A {
	void test();

}

package p2;
import p1.A;
public class C implements A  {
@Overrire
public void test()
{
	System.out.println("test ");
}
public static void main(String[] args) {
	C a=new C();
	a.test();
}
```

Every method by default in an interface not only abstract but also public and hands in the above program when we remove access specifier by default it become public an hands we don’t get any error.
Example:

```
package p1
public interface  A {
void test();// by default public
}


package p1;

public interface  A {
private/protected  void test();// error
```

```
}
```
Interface variable static and final

```
package interface11;
public interface A11
{
	final int i=10;
}
public class A1 implements A11{
public static void main(String[] args) {
System.out.println(A11.i);
}
}
```
We can never create object an interface
We can never keep main method in hands interface

```
package interface11;
public interface A11
{
	final int i=10;
}
public class A1 implements A11{
public static void main(String[] args) {
A11 a =new A11();// error
}
}
```
Reference variable of an interface can be created but then object can not be created

```
package interface11;
public interface D
{
public void test();

}
public   class A1 implements D{
public static void main(String[] args) {
D a1=new A1();// upcasting
a1.test();
}
public void test()
{
	System.out.println("test1");// test
}
}
```
We can not keep a constructor inside of interface

```
package interface11;
public interface AA
{
	AA()// error
	{

	}
}
```
Every variable by default public in interface

```
public interface AA
{
public int i=12;
protected /private i=12;// error ;
```

```
}
```
Class to class -> extends

Interface to interface ->extends

Interface to class –> implements;

Interface and class both using

Example

```
package interface11;
interface A
{
	public void test1();
}
interface B extends A
{
	public void test2();
}
public class E implements B{
	public void test1() {
		System.out.println("test1");
	}
	public void test2()
	{
		System.out.println("test2 ");
	}
	public static void main(String[] args) {
		E a=new E();
		a.test1();
		a.test2();
	}

}
```
In java classes does not support multiple inheritance but interface support multiple inheritance

```
package p4;

public interface A {
public void test();
}
package p4;

public interface B {
public void test2();
}
package p4;

public interface C {
public void test3();
}

package p4;

public class D implements C{

public void test()
{
	System.out.println("test");
```

```
}
public void test2()
{
	System.out.println("test2");
}
public void test3()
{
	System.out.println("test3");
}
public static void main(String[] args) {
	D a=new D();
	a.test();
	a.test2();
	a.test3();

}
}
```

Extends and implements both can be used in a class but first extends then implements

```
package both_extends_and_interface;

public interface A {
public void test1();

}
```

```
package both_extends_and_interface;

public class B {
	public void test2()
	{
		System.out.println("test2");
	}

}
```

```
package both_extends_and_interface;

public interface D {
public void test4();
}
```

```
package both_extends_and_interface;

public class C extends B implements A,D{
public void test1()
{
	System.out.println("test1");
}
public void test3()
{
	System.out.println("test3");
}
public void test4()
{
	System.out.println("test4");
}
public static void main(String[] args) {
	C a=new C();
	a.test1();
	a.test2();
```

```
	a.test3();
	a.test4();
}
}
```

ABstract

Abstract class is 0 to 100% incomplete
Every method should have abstract keyword specify it is incomplete
We can never create object of abstract class but then abstract class can consist of main method

```
 package abstrackexple;

abstract class A {
public abstract void test();
public void test2()
{
	System.out.println();
}
}
```

@program

```
package abstrackexple;

public abstract class B {
public static void main(String[] args) {
	B a=newB();// error not create abs obj

}
public void test()
{
System.out.println("gfhj");
}
}
```

3program

Static value can access by abs class name

```
package abstrackexple;

public abstract class B1 {
static int i=12;
public static void main(String[] args) {
	System.out.println(B1.i);
}
}
```

4program

```
package abstrackexple;
public abstract class B3 {

	public abstract void test();
	public void test2()
	{
		System.out.println("test2");
```

```
	}
}


class B4 extends B3{
public void test() {
	System.out.println("test1");
}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
B4 a =new B4();
a.test();
a.test2();
	}
}
package abstrackexple;

public abstract class C {
public abstract  void test();
}
```

Reference variable off abs class

```
package abstrackexple;

public abstract class C {
public abstract  void test();
}


package abstrackexple;


public class A1 extends C {

	public void test()
	{
		System.out.println("test");
	}
	public static void main(String[] args) {
		C a=new A1();// upcasting
		a.test();
	}
}
```

A constructor can be create in abs class only no arg constructor it possible

```
package abstrackexple;

public abstract class C {
	C()
	{
		System.out.println("cons");
	}
public abstract  void test();
}
package abstrackexple;


public class A1 extends C {

	public void test()
```

```
	{
		System.out.println("test");
	}
	public static void main(String[] args) {
		C a=new A1();// upcasting
		a.test();
	}
}
```

Multiple Inheritance abs class is not support only interface support multiple inheritance

** interface abs and class togher

```
package multi;

public interface A {
public void test();
}
```

```
package multi;

public abstract class B  implements A{
	public abstract void test2();


}
```

```
package multi;

public class C extends B{
public void test()
{
	System.out.println("test");
}
public void test2()
{
	System.out.println("test2");
}
	public static void main(String[] args) {
		C a=new C();
		a.test();
		a.test2();
}
}
```

We acan not create static method in an interface as can not be inherited and they are not inherited then we can not override

```
package multi;

interface A{
public static  void test();// error because static can not inherited
}
```

Abstract class method can not static static

We can not made static method abstract

```
package multi;

public abstract class B  implements A{
	public abstract static  void test2();

}
```

Static and non static final variable if not initializa will give bank flied error

```
interface A{
public final static  int i;//
public final static int i=99;//correct
}
```

A variable in abs class by default not final

Immidade changing value of an not static variable with out creating object it not possible

```
package abstrackexple;

public abstract class D {
int i=20;// non static variable
}
package abstrackexple;

public class D1 extends D{

	public static void main(String[] args) {

D1 a=new D1();
a.i=23;
	}

}


**
package abstrackexple;

public abstract class D {
 static int i=20;// static variable
}
package abstrackexple;

public class D1 extends D{

	public static void main(String[] args) {

D1 a=new D1();
a.i=23;
sop(i);//23
	}

}
```

Local variable we can modify again with out create object

```
package abstrackexple;

public class D1 extends D{
int k=60;// not static variable
k=20;// not possible when create class object then it possible
	public static void main(String[] args) {
		int j=10;// local variable
		j=89;// direct modify
D1 a=new D1();
a.i=25;
System.out.println(i);
	}
}
```

Summary :

Interface can store only incomplete method
By default every method is abs
Every variable by default in interface public static and final
Multiple inheritance support in interface

Abs class can consist of both complete and inclompled method
Every method by default not abstract we should used abstract key to spcifipe incomplete method
Variable by default not final
Multiple inheritance abs class not possible
Inheritance btw abs class and interface possible

Collection

Array vs collection

Array is fixed size ,size of the collection is dynamic
It can store homogenous data, can store homogeneous as well as heterogeneous
Array of memory usage it is less efficient ,it term of memory usage is more efficient
No underlying data structure ,has got underlying data structure to simply our work

Iterable

Collection

List queue set

ArrayList linklist priority tree set hashset

List:

List is an interface

Maintains insertion order

Allows duplicate data

Arraylist

It can contain duplicate items

It maintains insertion order

Allows random access of data

Internally it is implement as dynamic arrays

Program

```
import java.util.ArrayList;

public class A {

	public static void main(String[] args) {
		ArrayList a=new ArrayList();
		a.add(10);
		a.add("pradip");
		a.add(23.9);
		System.out.println(a);

	}

}
```

2program

```
package collection;

import java.util.ArrayList;

public class A {

	public static void main(String[] args) {
		ArrayList a=new ArrayList();
		a.add(10);
		a.add("pradip");
		a.add(23.9);
		System.out.println(a.get(1));

	}

}
```

Finding out size of collection

```
package collection;

import java.util.ArrayList;

public class B {
	public static void main(String[] args) {
		ArrayList a=new ArrayList();
		a.add(10);
		a.add("pradip");
		a.add(23.9);
		System.out.println(a.size());

	}
}
```

Collections in java is the class that help us perform sorting searching etc on collections

Collection can be sorted only when there is homogenous data store init or else we will gar exception

```java
package collection;

import java.util.ArrayList;
import java.util.Collections;

public class C {
	public static void main(String[] args) {
		ArrayList a=new ArrayList();
		a.add(10);
		a.add(56);
		a.add(23);
		Collections.sort(a);
		System.out.println(a);

	}
}
```

Searching

To perform a search in collection make sure that data sorted

```java
package collection;

import java.util.ArrayList;
import java.util.Collections;

public class A1 {
	public static void main(String[] args) {
		ArrayList a=new ArrayList();
		a.add(12);
		a.add(20);
		a.add(25);
	int index=   Collections.binarySearch(a, 20);
	System.out.println(a.get(index));
}
}
```

Contains();

Value present in array show true and not present false

```java
package collection;

import java.util.ArrayList;
import java.util.Collections;

public class C {
	public static void main(String[] args) {
		ArrayList a=new ArrayList();
		a.add(10);
		a.add(56);
		a.add(23);

		System.out.println(a.contains(20));

	}
}
```

Remove
a.remove(indexno);// only one element remove collection
clear :
a.clear();//all element remove from collection and memory also deleted
a.removeAll();// same clear // memory not deleted only value remove
contionsAll : compare the data of two collection

```
package collection;

import java.util.ArrayList;

public class A2 {
	public static void main(String[] args) {
		ArrayList a=new ArrayList();
		a.add(12);
		a.add(4);
		a.add(25);
		ArrayList b=new ArrayList();
		//b.add(12);
		b.add(4);
		//b.add(25);
	System.out.println(a.containsAll(b));  // compare two collection //true
	}
}
```

Io avoid storing heterogeneous

```
package collection;

import java.util.ArrayList;

public class A5 {
	public static void main(String[] args) {
		ArrayList<Integer> a=new ArrayList();
		a.add(12);
		a.add(20);
		a.add(90);
	}
}
```

Link list
“
It contain duplicate data
Maintains insertion order
Its internally implemented as diubly link list
Single link :
Here traversal to read the data happed only in one direction and hands it call as singly link list
Double link list :reading of data can be done both direction

```
package linklist;

import java.util.LinkedList;

public class A {
	public static void main(String[] args) {
		LinkedList a=new LinkedList();
		a.add(12);
		a.add("tset");
		a.add(22.5);
		System.out.println(a);
	}
```

```
}
```
Linklist program

```
package linklist;

import java.util.Iterator;
import java.util.LinkedList;

public class B {
	public static void main(String[] args) {
		LinkedList a=new LinkedList();
		a.add(12);
		a.add(20);
		a.add(22);
		Iterator itr=a.iterator();
		while(itr.hasNext())
		{
			System.out.println(itr.next());
		}
}
}
```
All function same like array

Set:

Its an interface

Does not maintain any insertion order

Cannot contain duplicate values

Hashset :

Uses hash table internally

Will contain only unique element

Does not maintain insertion order

16 row by default

75% load raso

!program

```
package set;

import java.util.HashSet;

public class A {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
HashSet h=new HashSet();
h.add(10);
h.add(12);
h.add(123);
h.add(12.6);
System.out.println(h);
	}
}
```
In the below expel total row 32 but load raso by default o.75f

```java
package set;

import java.util.HashSet;

public class A {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
HashSet h=new HashSet(32);
h.add(10);
h.add(12);
h.add(123);
h.add(12.6);
System.out.println(h);
	}

}
```

**load raso here o.90f

```java
package set;

import java.util.HashSet;

public class A {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
HashSet h=new HashSet(32,0.90f);
h.add(10);
h.add(12);
h.add(123);
h.add(12.6);
System.out.println(h);
	}

}
```

Collections should not apply on set.

Tree set

Sort and store data
Maintains ascending order
Contains unique element only like hashset
Should homogenous data

```java
package set;

import java.util.TreeSet;

public class Treeset {

	public static void main(String[] args) {
		TreeSet s=new TreeSet();
		s.add(4);
		s.add(34);
		s.add(21);
		s.add(21);
		System.out.println(s);

	}

}
```

String sort

```java
package set;

import java.util.TreeSet;

public class B {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		TreeSet s=new TreeSet();
		s.add("as");
		s.add("hfg");
		s.add("pradip");
		s.add("raju");
		System.out.println(s);

	}
	}
```

Queue

The queue interface basically orders the element in fifo manner

Priority

The priority queue class provides the facility of using queue

But it does not orders the element in fifo manner

```java
package queue;

import java.util.ArrayList;
import java.util.PriorityQueue;

public class A {

	public static void main(String[] args) {
		PriorityQueue a=new PriorityQueue();
		a.add(12);
		a.add(20);
		a.add(25);
		a.add(122);
		System.out.println(a);
		System.out.println(a.poll());
		System.out.println(a.poll());
		System.out.println(a.poll());
		System.out.println(a.poll());

}
}
```

Peek only top element give output

```java
package queue;

import java.util.PriorityQueue;

public class B {
	public static void main(String[] args) {
		PriorityQueue a=new PriorityQueue();
		a.add(12);
		a.add(20);
		a.add(25);
```

```
            a.add(122);
            System.out.println(a);
            System.out.println(a.peek());
System.out.println(a.peek());


}

}
```

Peek and poll both used

```
package queue;

import java.util.PriorityQueue;

public class B {
	public static void main(String[] args) {
		PriorityQueue a=new PriorityQueue();
		a.add(12);
		a.add(20);
		a.add(25);
		a.add(122);
		System.out.println(a);
		System.out.println(a.peek());//10
	a.poll();
System.out.println(a.peek());20
	a.poll();
	System.out.println(a.peek());//25

}

}
```

Vector

Inner class

Creating a class whith in anaogher class it class inner class

Local inner class :

```
package innerclass;

public class A {
 class B{
	 int i=45;

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

B a=new B(); // error
System.out.println(a.i);
}
}
```

How call inner class local variable

to access member of local inner class firstly create obj of outer class then using of reference of outer class create obj of inner class
then access the member of inner class as soon below

```
package innerclass;

public class A {
 class B{
       int i=45;

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

A a1=new A();//  obj create A class first
B a=a1.new B(); // take reference
System.out.println(a.i);
}
}
```

Inner class better security

```
package innerclass;

public class A {
 class B{
       int i=45;
       public void test ()
       {
             System.out.println("test ");
       }

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

A a1=new A();//  obj create A class first
B a=a1.new B(); // takev reference
a.test();
System.out.println(a.i);
}
}
```

We can not create static member into local inner class
Static member can not used inside local inner class

```
package innerclass;

import innerclass.B.A;

public class B {
       class A{
             static int j=10;

              public static  void test ()// error
              {
                    System.out.println("test ");
              }

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

       B a1=new B();
       A a= a1.new A();
       a.test();
       System.out.println(a.i);// not acees static variable local inner class
```

```
}
}
}
```

Creating object of local inner class we can not access member of outer class

```
package innerclass;

import innerclass.C.B;

public class C {
	int j=90;
	class B{
		int i=45;
		public void test ()
		{
			System.out.println("test ");
		}

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

	C a1=new C();//  obj create A class first
	B a=a1.new B(); // takev reference
	a.test();
	System.out.println(a.i);
	System.out.println(a.j);// can not access outer class variable using inner
class reference
	}
	}
```

Inside constructor inner class

```
package innerclass;

public class A1 {
	class A11{
		A11()
	{
		System.out.println("from a1");
	}
	}
	public static void main(String[] args) {
	A1 a1=new A1();
	A11 b1=a1.new A11();

	}

}
```

We can inherited member of outer class into inner class and then access this member by creating obj of inner class

```
package innerclass;

public class A2 {
	int i=10;
```

```
class A3 extends A2{
	int j=20;

}
public static void main(String[] args) {
	A2 a=new A2();
	A3 a1 =a.new A3();
	System.out.println(a1.i);
	System.out.println(a1.j);
}

}
```

Static member of outer class can be inherited into local inner class but then static member can not create local inner class

```
package innerclass;

public class A2 {
	static int i=10;
class A3 extends A2{
	int j=20;

}
public static void main(String[] args) {
	A2 a=new A2();
	A3 a1 =a.new A3();
	System.out.println(a1.i);//
	System.out.println(a1.j);
}

}
```

We can create more then one inner class inside the same class

```
package innerclass;

public class A5 {
class B
{
	int i=12;
}
class C
{
	int j=10;
}
	public static void main(String[] args) {
	A5  a= new A5();
	B a1=a.new B();

	System.out.println(a1.i);
	C c1=a.new C();
	System.out.println(c1.j);
	}
```

```
}
```

When will complile above program following three class but above program save single a.java file

A$B.class

A$C.class

A.class

Inheritance btw two local inner classes is possible

```
package innerclass;

public class A5 {
class B
{
	int i=12;
}
class C extends B
{
	int j=10;
}
	public static void main(String[] args) {
	A5  a= new A5();
C a1=a.new C();

	System.out.println(a1.i);//12

	System.out.println(a1.j);//10
	}
}
```

Sib ans cons and iib

Iib can be create inside local inner class

```
package innerclass;

public class A6 {
	class B
	{
		int i;
	{i=122;
		System.out.println("illb ");

	}
	static
	{
		System.out.println("from static  ");
	}
 B()
 {
	 System.out.println("frm cons ");
 }

	}
```

```
	public static void main(String[] args) {
		A6 a =new A6();
		B a1=a.new B();
	System.out.println(a1.i);

	}

}
```

Static initialization
Block can be keep local inner class

Interface vs class
Interface it give incomplete method
Class it give complete method

Inner class can implement interface

```
package innerclass;
public interface E
{
	public void test();
}
public class A7 {
	class B implements E
	{
		public void test()
		{
			System.out.println("test");
		}
	}
	public static void main(String[] args) {
		A7 a=new A7();
		B a1=a.new B();
		a1.test();
	}

}
```

Static inner class

In this class we can keep static and non static member both

```
package staticinnerclass;

import innerclass.B;

public class A {
	static class B
	{
		static int i;
		int j=23;
```

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

		B b1=new B();
		System.out.println(b1.j);
		System.out.println(b1.i);
		System.out.println(B.i);

	}

}
```

Inside static class we can create sib ,iib ,constructor

```
package staticinnerclass;

import staticinnerclass.A.B;

public class A1 {
	static class B
	{
		static {
			System.out.println("sib");
		}
		B()
		{
			System.out.println("constructor");
		}
		{
			System.out.println("iib ");
		}
	}
	public static void main(String[] args) {

		B b1=new B();

	}
}
```

We can create obj of static inner class and access of member of outer class with out inhertance not possible

```
static class B
	{
		static int i;
		int j=23;

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

		B b1=new B();
		System.out.println(b1.j);
		System.out.println(b1.i);
		System.out.println(B.i);

	}
```

Access two static class inside outer class

```
package staticinnerclass;

import staticinnerclass.A3.B;

public class A4 {

	static class B
	{
		int i=23;
	}
	static class C
	{
		int j=45;
	}
	public static void main(String[] args) {

		B b1=new B();
		C c1=new C();

		System.out.println(b1.i);
	System.out.println(c1.j);

	}
}
```

Inheritance btw two static inner class is possible

```
package staticinnerclass;

import staticinnerclass.A4.B;
import staticinnerclass.A4.C;

public class A6 {
	static class B
	{
		int i=23;
	}
	static class C  extends B
	{
		int j=45;
	}
	public static void main(String[] args) {


		C c1=new C();

		System.out.println(c1.i);
	System.out.println(c1.j);

	}
}
```

Creating object an inner class can not acess outer class

```
package staticinnerclass;

import staticinnerclass.A4.B;
import staticinnerclass.A4.C;
```

```
public class A6 {
int i=20;
	static class B
	{
		Int j=23;
	}

	public static void main(String[] args) {


		B c1=new B();

		System.out.println(c1.i);// not possible get error
	System.out.println(c1.j);

	}

}
```

Interface of static class

Static inner class can implement interface

```
package staticinnerclass;

public interface C {
	public void test ();


}
```

```
package staticinnerclass;

public class D {
static class B implements C
{
	public void test()
	{
		System.out.println("test ");
	}
}
	public static void main(String[] args) {
	B a1=new B();
	a1.test();



	}

}
```

Inheriting local class member into static class is not possible

```
public class A4 {

	 class B// local class
	{
		int i=23;
	}
	static class C extends B // error
	{
		int j=45;
	}
```

Inheriting form static inner class to local inner class possible

```
package staticinnerclass;

public class A7 {
	static class B
	{
		int i=19;
		static int j=10;
	}
class C extends B{

}
public static void main(String[] args) {
	A7 a1 = new A7();

	C c1=a1.new C();
	System.out.println(c1.i);
	System.out.println(C.j);
}
}
```

We can not create main method inside local inner class because it is static

We can create main method inside static inner class but then it will exceute when call it

```
package staticinnerclass;

public class B {
 static class C
 {
	 int  j=10;
	 public static void main(String[] args) {
		int i=100;
		System.out.println(i);
	}
 }
 public static void main(String[] args) {
	C a1= new C();
	System.out.println(a1.j);
	C.main(null);
}
}
```

Annony class

A class with out any name it call annonamtous class

```
package annonious;

public interface B {
public void test();
}
```

```
package annonious;

public class A {
	public static void main(String[] args) {
		B b1=new B() // implements
		{
```

```
		public void test() {
			System.out.println("from test B");
		}
		};
	b1.test();
	}


}
```

Anonomatic anutomatic inherited the member of the obj before yet

```
package annonious;

public class C {
	public void test1()
	{
		System.out.println("from test  class method ");
	}

}
package annonious;

public class A {
	public static void main(String[] args) {
		C b1=new C() // extends
		{


		public void test1() {
			System.out.println("from test B");
		}
		};
	b1.test1();
	}
}
```

Annonious class can inheried the member of local inner class as well

```
package annonious;

public class A1 {
	class B
	{
		int i=90;
	}
	public static void main(String[] args) {
		A1 a1=new A1();
		B b1=a1.new B()
				{

				};
				System.out.println(b1.i);//90
	}

}
```

Anony classes can inherited the members of any other class ,any other interface ,local

and static inner class but can not inherited member of outer class

```
package annonious;

import annonious.A1.B;

public class A3 {
	static class B
	{
		public void test ()
		{
			System.out.println("test from B");
		}
	}
	public static void main(String[] args) {

		B b1=new B()
				{
			public void test()
			{
				System.out.println("test override anony class ");
			}
				};
				b1.test();
	}


}
```

Threads :

Multitasing done at the program level is call threads
By using thread class :
Here we inherited run method from thread class then we override it user define class
To start run thread we used start method of thread class
Run thread is user define thread whrer means thread dy default always main thread run first and then user define thread
Which thread will executed how much time can not predicted by programmer as it is decided by the processor dependent on avability

```
package threads;

public class A  extends Thread{
public void run()
{
	for( int i=0;i<1000;i++)
	{
		System.out.println("test1");
	}
}
public static void main(String[] args) {
	A a1=new A();
```

```
        a1.start();
        for( int i=0;i<1000;i++)
        {
                System.out.println("test2");
        }

}
}
```

Creating a thread using runnable interface

```
package threads;

public class B  implements Runnable {
        public void run()
        {
                for (int i=0;i<100;i++)
                {
                        System.out.println("from test 1");
                }
        }
        public static void main(String[] args) {
            B b1 =new B();
            Thread t1=new Thread(b1);
            t1.start();
                for (int i=0;i<100;i++)
                {
                        System.out.println("from test 3");
                }
                }
}
```

Thread synchronization

When two thread are operating on a common data in a non synchronized then data will get corrupted

```
package threadsynch;

public class A {
        int balance;
        public static void main(String[] args) {
                 A a1=new A();
                 a1.account();
                 System.out.println(a1.balance);

        }
        public  void account()
        {
                Thread t1=new Thread(new Runnable ()
                                {
                        public void run()
                        {
                        add();
                        }
                        });
                                Thread t2=new Thread(new Runnable ()
                                {
                                        public void run()
                                        {
```

```
                        sub();
                                }
                        });
                        t1.start();
                        t2.start();
                        try
                        {
                                t1.join();
                                t2.join();
                        }
                        catch(Exception e)
                        {
                                System.out.println(e);
                        }

        }
        public void add()
        {
                for (int i=10;i<1000;i++)
                {
                        balance+=i;
                }
        }
        public void sub()
        {
                for (int i=10;i<1000;i++)
                {
                        balance-=i;
                }
        }

}
```

Synchronized method :

Synchronization if a method made sync then a thread which aqueous are obj lak can only execte this method

Every obj only one lock which ever thread will executed sncy method while other thread has to wait for the lock release

Thread will release the lock only after complete execution by doing this we are sure advance a particular thread executed completed only then next can start this will avoid data corruption of common data

Thead joined

It help us to join operation or task of two or more thread

```
package threadsynch;

public class A {
```

```
	int balance;
	public static void main(String[] args) {
		 A a1=new A();
		 a1.account();
		 System.out.println(a1.balance);

	}
	public  void account()
	{
		Thread t1=new Thread(new Runnable ()
				{
			public synchronized void run()
			{
			add();
			}
			});
				Thread t2=new Thread(new Runnable ()
				{
					public synchronized void run()
					{
			sub();
					}
				});
				t1.start();
				t2.start();
				try
				{
					t1.join();
					t2.join();
				}
				catch(Exception e)
				{
					System.out.println(e);
				}

	}
	public void add()
	{
		for (int i=10;i<1000;i++)
		{
			balance+=i;
		}
	}
	public void sub()
	{
		for (int i=10;i<1000;i++)
		{
			balance-=i;
		}
	}

}
```

Wait() and notify() ,notifyAll()

Wait: it will hold the execution of thread

Notify :

It will end the wait of any one thread so that it can resume its execution
notifuAll
it will end the wait of all the thread at ones which are in wating status the thread now executed in queue one after another
 wait notify notifyall belong to object class

```
package waitnotify;

public class A {

	public static void main(String[] args) {
	B a1=new B();
	a1.start();
	synchronized(a1)
	{
		try
		{
			a1.wait();
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
	System.out.println(a1.balance);

	}

}
```

```
package waitnotify;

public class B extends Thread {
	int balance;
	public void run()
	{
	for ( int i=0;i<1222;i++)
		{
		balance+=i;
		}
	notify();
		}
}
```

Deadlock: when two tread are waiting each other to release lock and nither of thread is release the lock then this tread has got into date lock state
Lifecycle :

To see the current state we get state method of thread class

```java
package threads;

public class D  extends Thread {
	public void run()
	{
		System.out.println("running ");
	}
	public static void main(String[] args) {
		D a=new D();
		System.out.println(a.getState());
		a.start();
		System.out.println(a.getState());
		try {


		Thread.sleep(5000);
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
		System.out.println("teminated ");
	}

}
```

```
New
Runnable
running
terminated
```

which thread will be executed first is dependent priority
NORM_PRIORITY value 5
MAX_PRIORITY value 10
MIN_PRIORITY VALUE 1
WHEN A programmer set a priority it is a request
Priority set inernally it is comment

```java
package threads;

public class E extends Thread {

	public void run()
	{
		System.out.println("running ");
	}
	public static void main(String[] args) {
		E a=new E();

		a.start();
		a.setPriority(MAX_PRIORITY);
		System.out.println(a.getPriority());
		a.setPriority(MIN_PRIORITY);
		System.out.println(a.getPriority());
	}
}
```

Thread concept are idial used for server side implementation to build application like tomcat
For every incoming request a thread will be pick up and will be assinge request
Thread pool is collection of thread when the request handle using thread multitasking can be perform and hands it will fill the to user that all the request are being process at ones
 Ones the request handle we put the thread back thread poll
Every time we don’t create thread and destroy because that will reduse efficiency
Rather after uses the thread it store thread poll for feather re used

It increasing efficiency and perform of server
assert
assert help us tto check the business condition only if business condition true assert will continue program execution but if condition is fail assert is not continue with the execution
assert was introduce jdk 1.4
step to configure anable a asset
go to run drop down select run configuration and go two arg under VM arg set -ea it means its check assert condition the work assert and -da its means assert condition not check

example

```
!program
package assert1;

public class A {
	public static void main(String[] args) {
		int age =20;
		assert age>20;
		System.out.println("register your self " );
	}

}
```

@program

```
package assert1;

public class A {
	public static void main(String[] args) {
		int age =200;
		try
		{
		assert age>20;
		System.out.println("register your self " );
		}
		catch(Exception e) {
			System.out.println(e);
		}
		}
}
```

Multiple assert statement can used one class

```
package assert1;

public class B {
	public static void main(String[] args) {
		assert true;
		System.out.println("hello");
		assert  false;
		System.out.println("error ");
	}

}
3program
package assert1;

public class A1 {

	public static void main(String[] args) {

assert test();
System.out.println("from main");// second from main

	}
public static boolean test()
{
```

```
		System.out.println("from test");// if true then first from test
		//return false ;
		return true;
}
}
```

Generics

A generic class help us to create a variable such that any kind of value can be store in the variable as the data type of variable is decided base on the kind of value is in it
This was intrigated jdk 1.5

Program

```
package generics;

public class A<x>{
	x i;

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		A a=new A();
		a.i=10.3;
		a.i="a";
		System.out.println(a.i);

	}

}
```

2program

```
	package generics;

public class B<x> {
	x i;
	x j;
	x k;

	public static void main(String[] args) {
		B a=new B();
		a.i=12;
		a.j="a";
		a.k=20.2f;
		System.out.println(a.i);
		System.out.println(a.j);
		System.out.println(a.k);
	}


}
```

3program

```
package generics;

public class B<x,y,z> {
	x i;
```

```
	y j;
	z k;

	public static void main(String[] args) {
		B a=new B();
		a.i=12;
		a.j="a";
		a.k=20.2f;
		System.out.println(a.i);
		System.out.println(a.j);
		System.out.println(a.k);
	}


}
```

4 program

Generics can not apply local variable and static varible

It only for non static variable and method also not static method

```
package generics;

public class C<X> {
	X i;
X j;
	C c1;
	public X test()
	{
		c1=new C();
		c1.i=30.3f;
		return i;
	}

	public static void main(String[] args) {
	C a2 =new C();
	c2.j=a2.test();
	System.out.println(c2.j);
	}

}
```

HashCode

Hashcode give me interger representation of obj memory address

```
package hashcode;

public class A {
	public static void main(String[] args) {
		A a1=new A();
		System.out.println(a1);
		System.out.println(a1.hashCode());
	}
}
```

System.out.println()

System :

It a class

Out: it static final print stream reference variable

Println:

Is a non static memthod

Annotation

Annotation are set introduction given to the compiler during compilation

Annotation over introduce in jdk version 1.5

Following annotation of java

@Override: this annotation help us to check weather ewe are overriding a method are not

@ serpace warring @ depecated

```
package annotationapp;

public class A {
	public void test()
	{
		System.out.println("from A");
	}

}
package annotationapp;

public class B extends A{
	@Override// check in compile time
public void test()
{
	System.out.println("test B");
}
	public static void main(String[] args) {

	}
}
```

@SuppressWarnings()

This annotation help us to suppress warning messages on the program

```
package annotationapp;

public class C {
	@SuppressWarnings("unused ")

	public static void main(String[] args) {
		int i;

	}

}
```

@Deprecated

This annontation Help us notify that particular method not in used

```
package annotationapp;

public class D {

	public static void main(String[] args) {
		D a= new D();
		a.test();

	}
	@Deprecated
	public void test()
	{
		System.out.println("test ");
	}

}
```

String class

String is a class in java which consist of several build in method using which we can manupulated string data easy

```
package stringclass;

public class A {

	public static void main(String[] args) {
		String s=" Pradip Love ";
```

```
                        System.out.println(s.toLowerCase());
                        System.out.println(s.toUpperCase());
                        System.out.println(s.trim());

        }

}
```

Remove white space benning and end string used trim()

Sting split ()

```
package stringclass;

public class A {

        public static void main(String[] args) {
                String s="Pradip Love to you  ";

        String[] s2   =s.split(" ");
System.out.println(s2[0]);
System.out.println(s2[1]);
System.out.println(s2[2]);
System.out.println(s2[3]);
        }

}
Length()
package stringclass;

public class A {

        public static void main(String[] args) {
                String s="Pradip Love to you  ";
                System.out.println(s.length());// 20

        String[] s2   =s.split(" ");
System.out.println(s2[0].length());//6
System.out.println(s2[1]);
System.out.println(s2[2]);
System.out.println(s2[3]);
        }

}
CharAt()
package stringclass;

public class A {

        public static void main(String[] args) {
                String s="Pradip Love to you  ";
                System.out.println(s.length());// 20

        String[] s2   =s.split(" ");
System.out.println(s2[0].length());//6
System.out.println(s2[1]);
System.out.println(s2[2]);
System.out.println(s2[3]);
        }
```

```
}
Startwith() and endwith()
package stringclass;

public class B {

	public static void main(String[] args) {

String s1="pradip giri";
System.out.println(s1.charAt(2));// a
System.out.println(s1.startsWith("a"));// false
System.out.println(s1.endsWith("i"));// true
}
}
charAt();
package stringclass;

public class A1 {

	public static void main(String[] args) {
		String s1="pradip";
//		for(int i=0;i<s1.length();i++)
//		{
//			System.out.println(s1.charAt(i));
//		}
		for(int i=s1.length()-1;i>=0;i--)
		{
		System.out.print(s1.charAt(i));
			//System.out.println(i);
		}

	}

}

Print a given string reverse manner with out build function
Reverse string
package interview;

public class Reverse {
	public static void main(String[] args) {
		String s="testing";
		int size=s.length();

	String		rev=" ";
		for(int i=size-1;i>=0;i--)
		{
			rev=rev+s.charAt(i);

			System.out.print(s.charAt(i));
		}	System.out.println(rev);
	}

}

Bigest a word and print a string
package interview;

public class Bigword {
```

```
	public static void main(String[] args) {
		String s1="pradip pradip";

		String[] s2=s1.split(" ");
		String temp=null;
		if(s2[0].length()>s2[1].length())
		{
			temp=s2[0];
		}
		if (s2[0].length()<s2[1].length())
		{
		 temp = s2[1];
		}
		if (temp!=null)
		{
			System.out.println("bigest wors:"+temp);

		}
		else
		{
			System.out.println("many word same ");
		}
	}


}
Palindrome
package stringclass;

public class Plindroom {

	public static void main(String[] args) {
		String s1="mama";
		int j=s1.length()-1;
		int count=0;
		for(int i=0;i<s1.length();i++)
		{
			if(s1.charAt(i)==s1.charAt(j--))
			{
				count++;
			}
		}
		if(count == s1.length())
		{
			System.out.println("palindrome");
		}
		else
		{
			System.out.println("not palindrome ");
		}
	}

}
 Time call run op tome call nur
package interview;

public class A1 {
	static String reverse=" ";
```

```
	static String temp=null;
	public static void main(String[] args) {


String s1="time call run ";
String[] d=s1.split(" ");
if(d[0].equals("run"))
{
	for (int i=d[0].length()-1;i>=0;i--)
	{
		reverse=reverse+d[0].charAt(i);
	}
	temp=reverse + " "+d[1]+" "+d[2];
}
if(d[1].equals("run"))
{
	for (int i=d[1].length()-1;i>=0;i--)
	{
		reverse=reverse+d[1].charAt(i);
	}	temp=d[0] + " "+reverse+" "+d[2];
}
if(d[2].equals("run"))
{
	for (int i=d[2].length()-1;i>=0;i--)
	{
		reverse=reverse+d[2].charAt(i);
	}	temp=d[0]+ " "+d[1]+" "+reverse;
}
	System.out.println(temp);




	}

}
```

Mutable and immutable

Seter and getter

```
package interview;

public class A2 {
	int age;
public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}

public static void main(String[] args) {
	A2 a=new A2();
	a.setAge(23);
```

```
		System.out.println(a.getAge());
}
}
```

Mutable object are the one which value keep changing

Immutable value are the one which value can never alter

Rule to design immutable

1)make class final
2)make variable final and private
3) initialize variable through constructer
4)we only getter not used setter

```
package mutable;

public class A {
	final private int i;
	final private int j;
	A(int i,int j)
	{
		this.i=i;
		this.j=j;
	}
	public int getI() {
		return i;
	}
	public int getJ() {
		return j;
	}
public static void main(String[] args) {
	A a=new A(10,20);
	System.out.println(a.getI());//0
	System.out.println(a.getJ());//20
}
}
```

String class immutable

== and equal

== :

compare memory address of string obj

Equal method :compare values of string

```
package mutable;

public class B {

	public static void main(String[] args) {

String s1=new String("xyz");
String s2=new String("xyz");
System.out.println(s1==s2);// false
System.out.println(s1.equals(s2));//ture

	}

}
```

case sensitive string compare

```
package immutable;

public class A {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		String s1=new String("hello");
		String s2=new String("Hello");
		System.out.println(s1==s2);// false
		System.out.println(s1.equals(s2));// false


	}

}
```