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 p**ublic** **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

**p**ublic **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 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 auto matic place super keyword dush 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();

}

}

f

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 moditify 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 orriding 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;

**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

**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

**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

**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

**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

**package** throwandthrows;

**class** A33 {

**public** **static** **void** test() **throws** Exception, NullPointerException

{

**int** i=10/ 0;

System.***out***.println(i);

}

}

**public** **class** A3

{

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

A33 a =**new** A33();

**try**

{

a.*test*();

}

**catch**(Exception e)

{

System.***out***.println(e);

}

System.***out***.println("ghdjk");

}

}

Throw

Throw keyword help us to take the memory address of the obj and give to catch

Ex1 **package** throwandthrows;

**class** InsufficientFunds **extends** Exception {

}

**public** **class** A4

{

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

**int** bal=50000;

**int** amt=10000;

**if**(amt>bal)

{**try** {

**throw** **new** InsufficientFunds();

}

**catch**(InsufficientFunds e)

{

System.***out***.println(e);

}

}

**else** {

System.***err***.println("you have many");

}

}

}

Ex2

**package** throwandthrows;

**class** InsufficientFunds **extends** Exception {

InsufficientFunds()

{

System.***out***.println("low balance ");

}

}

**public** **class** A4

{

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

**int** bal=5000;

**int** amt=10000;

**if**(amt>bal)

{**try** {

**throw** **new** InsufficientFunds();

}

**catch**(InsufficientFunds e)

{

System.***out***.println(e);

}

}

**else** {

System.***err***.println("you have many");

}

}

}

Ex3

**package** throwandthrows;

**class** InsufficientFunds **extends** Exception {

InsufficientFunds()

{

System.***out***.println("low balance ");

}

}

**public** **class** A5

{

**public** **static** **void** main(String[] args) **throws** Exception {

**throw** **new** InsufficientFunds();

System.***out***.println("helo");//error after throw we write it error not run program

}

}

Ex4

**package** throwandthrows;

**class** InsufficientFunds **extends** Exception {

InsufficientFunds()

{

System.***out***.println("low balance ");

}

}

**public** **class** A5

{

**public** **static** **void** main(String[] args) **throws** Exception {

System.***out***.println("helo");// before throw keyword u write anything it work

**throw** **new** InsufficientFunds();

}

}

Reflection

It help us analysis the member of class develop by some one else

Jar

Jar is a clocetion of .class file and interface

Step to generated jar file :

Right click in project select export then jar file click in next and fnish

Linking the jar into current program

Right click in project properties java build path

Library path and click add external path