practice programs

-----------------

practice programs

-----------------

basic application:

class Tests

{

public static void main(String[] args)

{

System.out.println("Hello World!");

}

}

E:\java\java practice programs>javac Tests.java

E:\java\java practice programs>java Tests

Hello World!

example 2:instance method

-------------------------

class Tests1

{

void m1()

{

int a=10;

System.out.println(a);

}

public static void main(String[] args)

{

Tests1 t = new Tests1(); // memory allocatee when objec is created

t.m1();

}

}

E:\java\java practice programs>javac Tests1.java

E:\java\java practice programs>java Tests1

10

example 3 : static method

--------------------------

class Tests1

{

static void m1()

{

int a=20;

System.out.println(a);

}

public static void main(String[] args)

{

Tests1.m1(); // memory is created when .class file is loading

}

}

E:\java\java practice programs>javac Tests1.java

E:\java\java practice programs>java Tests1

20

example 4: instance and static variables and methods

---------------------------------------------------

class Tests1

{

int a=10;//instance variables

int b=20;

static int c=30;//static variables

void m1()

{

System.out.println(a);

System.out.println(b);

System.out.println(Tests1.c);// classname.variable since it is the ststic variable

}

static void m2()

{

Tests1 t = new Tests1();

System.out.println(t.a);//This is static area so to execute instance variables we give it as

System.out.println(t.b); // reference variable.variable name

System.out.println(Tests1.c);

}

public static void main( String [] args)

{

Tests1 t = new Tests1();// This is for calling the void method

t.m1();

Tests1.m2();// this is for calling the static method

}

}

E:\java\java practice programs>javac Tests1.java

E:\java\java practice programs>java Tests1

10

20

30

10

20

30

Example 5: constructors //1) used to write the logics of application. 2) to intitialize instance values.

----------------------------------------------------------

class Testsc

{

void m1()

{

System.out.println("m1 method");

}

Testsc()

{

System.out.println("0-arg constructor");

}

Testsc(int a)

{

System.out.println("1-arg constructor");

}

public static void main(String[] args)

{

Testsc t = new Testsc();

Testsc t1 = new Testsc(10);

t.m1();

}

}

E:\java\java practice programs>javac Testsc.java

E:\java\java practice programs>java Testsc

0-arg constructor

1-arg constructor

m1 method

Example 6: to call 1 constructor using another by "this method"

---------------------------------------------------------------

class Testsc

{

Testsc()

{

this(10); //indicates the 1 arg constructor

System.out.println("0 arg constuctor");

}

Testsc(int a )

{

System.out.println("1 arg constuctor");

}

public static void main(String[] args)

{

new Testsc();

}

}

E:\java\java practice programs>java Testsc

1 arg constuctor

0 arg constuctor

Example 7: to assaign different values to different emp

---------------------------------------------------------

class Emp

{ // instance variables

int eid;

String ename;

float esal;

Emp(int eid, String ename, float esal)//converstion of local values to instance values

{

this.eid=eid;

this.ename=ename;

this.esal=esal;}

void disp()

{

System.out.println("eid="+eid);

System.out.println("ename="+ename);

System.out.println("esal="+esal);

}

public static void main (String[] args)

{

Emp e1 = new Emp(111,"teja",11500.65f);

e1.disp();

Emp e2 = new Emp(222,"ajay",22120.73f);

e2.disp();

}

}

E:\java\java practice programs>java Emp

eid=111

ename=teja

esal=11500.65

eid=222

ename=ajay

esal=22120.73

Example 8: Instance block

-----------------------------

class Testsi

{

Testsi()

{

this(10);

System.out.println("o arg constructor");

}

Testsi(int a)

{

System.out.println("1 arg constructor");

}

{System.out.println("instance block");//instance bloclk

}

public static void main(String[] args)

{

new Testsi();

}

}

E:\java\java practice programs>java Testsi

instance block

1 arg constructor

o arg constructor

Example 9: the insatnce block is executed the no of times object is created

------------------------------------------------------------

class Testsi

{

void m1() //insatnce method

{

System.out.println("m1 method");

}

Testsi()//constructor 1

{

this(10);

System.out.println("o arg constructor");

}

Testsi(int a) //constucter 2

{

System.out.println("1 arg constructor");

}

{System.out.println("instance block");//instance block

}

public static void main(String[] args)

{

new Testsi();//to call constructor

Testsi t1 = new Testsi();// to call instance method

t1.m1();

}

}

E:\java\java practice programs>java Testsi

instance block

1 arg constructor

o arg constructor

instance block

1 arg constructor

o arg constructor

m1 method

Example !0: to implement all keywords,variables,methods,constructors,blocks

----------------------------------------------------------------

class Testsi

{

int a=20;// instance variables

static int b=30;//static variables

void m1(int a)//instance method

{

System.out.println("instance method");

}

static void m2(String str)//static method

{

System.out.println("static method");

}

Testsi()//constructor

{

this(10);//to call 2 coonstructor without creating a new object

System.out.println("o arg constructor");

}

Testsi(int a)// 2 constructor

{

System.out.println("1 arg constructor");

}

{System.out.println("instance block");//instance block

}

static // static block is executed only once for the class

{

System.out.println("static block");//static block

}

public static void main(String[] args)

{

new Testsi();//call constructor 1

Testsi t1 = new Testsi();//call instance method

t1.m1(40);

Testsi.m2("teja");//to call static method

}

}

output:

---------

static block

instance block

1 arg constructor

o arg constructor

instance block

1 arg constructor

o arg constructor

instance method

static method

OOPS :1)inheritance 2)polymorphism 3)abstraction 4)encapsulation

Example 11: Inheritance //the property of acquiring properties from one class to another is called inheritance

----------------------------------------------------------------

class parent

{

int a=10;

int b=20;

}

class child extends Parent

{

int x=100;

int y=200;

void add(int i, int j)

{

System.out.println(i+j);//local variables

System.out.println(x+y);//current

System.out.println(a+b);//super

}

public static void main(String[] args)

{

new child().add(1000,2000);

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

3000

300

30

Example 12: inheritance if all the variable names are same use "this" and "super" keywords

----------------------------------------------------------------

class Parent

{

int a=10;

int b=20;

}

class child extends Parent

{

int a=100;

int b=200;

void add(int a, int b)

{

System.out.println(a+b);

System.out.println(this.a+this.b);

System.out.println(super.a+super.b);

}

public static void main(String[] args)

{

new child().add(1000,2000);

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

3000

300

30

Example 13: parent methods in inheritance

---------------------------------------------

class Parent

{

void m1()

{

System.out.println("parent method");

}

}

class child extends Parent

{

void m1()

{

System.out.println("child method");

}

void m2()

{

this.m1();

super.m1();

System.out.println("child 2 method");

}

public static void main(String[] args)

{

new child().m2();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

child method

parent method

child 2 method

Example 13: parent constructors in inheritance

--------------------------------------------------

class Parent

{

void m1()

{

System.out.println("parent method");

}

Parent()

{

System.out.println("0 arg constructor");

}

}

class child extends Parent

{

void m1()

{

System.out.println("child method");

}

void m2()

{

this.m1();

super.m1();

System.out.println("child 2 method");

}

child()

{

System.out.println("child constructor");

}

public static void main(String[] args)

{

new child().m2();//object created and method called

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

0 arg constructor

child constructor

child method

parent method

child 2 method

Exampele: important for real time

---------------------------------------

class Parent

{

Parent()

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

child()// super(); compiler generated contructor

{

System.out.println("child constructor 1");

}

child(int a)//super();generated by compiler

{

System.out.println("child contructor 2");

}

public static void main(String[] args)

{

new child();//object creation

new child(10);//object creation

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent constructor

child constructor 1

parent constructor

child contructor 2

Example 14:no child constructor then directly parent constructor is called

---------------------------------------------------------------------------

class Parent

{

Parent()

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

/\*code is generated by compiler

child()

{

super();

}

\*/

public static void main(String[] args)

{

new child();//object creation

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent constructor

Example 1: //Insatance blocks in inheritance

----------------------------------------------

class Parent

{

{//insatnce block

System.out.println("parent class instance block");

}

Parent()//constructor

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

{//instance block

System.out.println("child class instance block");

}

child()//constructor

{

System.out.println("child constructor");

}

public static void main(String[] args)

{

new child();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent class instance block

parent constructor

child class instance block

child constructor

Example 1: //Static blocks in inheritance

----------------------------------------------

class Parent

{

{

System.out.println("parent class instance block");

}

static

{

System.out.println("parent class static block");

}

Parent()

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

{

System.out.println("child class instance block");

}

static

{

System.out.println("child class static block");

}

child()

{

System.out.println("child constructor");

}

public static void main(String[] args)

{

new child();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent class static block

child class static block

parent class instance block

parent constructor

child class instance block

child constructor

Example2: static block is executed once

insatnce blocks are executed the no.of times the objects are created

----------------------------------------------------------------------------------

class Parent

{

{

System.out.println("parent class instance block");

}

static

{

System.out.println("parent class static block");

}

Parent()

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

{

System.out.println("child class instance block");

}

static

{

System.out.println("child class static block");

}

child()

{

System.out.println("child constructor");

}

public static void main(String[] args)

{

new child();

new child();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent class static block

child class static block

parent class instance block

parent constructor

child class instance block

child constructor

parent class instance block

parent constructor

child class instance block

child constructor

// Polymorphysm : the abiliy to appear in many forms

---------------------------------------------------

1)Compile time polymorphysm

2)run time polymorphysm

IN the compile time polymorphysm

OVERLOADING:// TO achieve this 1 class is required

------------------------------------------------------

1) method overloading

----------------------------------------------

class Parent

{

void m1(int a)//overloading method

{

System.out.println("int m1 method");

}

void m1(int a, int b)//same method name with diff no.of args

{

System.out.println("int, int m1 method");

}

void m1(char ch)//same method with diff data types

{

System.out.println("char m1 method");

}

public static void main(String[] args)

{

Parent t = new Parent();

t.m1(10);

t.m1(10,20);

t.m1('T');

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Parent

int m1 method

int, int m1 method

char m1 method

Example: 2 constructor overloading

-----------------------------------

class Parent

{

Parent(int a)

{

System.out.println("int arg constructor");

}

Parent(int a, int b)

{

System.out.println("int, int arg constructor");

}

Parent(char ch)

{

System.out.println("char arg constructor");

}

public static void main(String[] args)

{

new Parent(10);

new Parent(10,20);

new Parent('T');

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Parent

int arg constructor

int, int arg constructor

char arg constructor

Example : 3 operator overloading only ("+") operator is supporteed by java

------------------------------------------------------------------------

class Parent

{

public static void main( String[] args)

{

System.out.println(10+20);

System.out.println("amma"+"nanna");

System.out.println(100+"amma");

System.out.println(10+"amma"+"nanna"+10+20);

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Parent

30

ammananna

100amma

10ammananna1020

OVERRIDING: // to achieve the overriding concept 2 classes are required and should have the parent and child relation in inheritance

---------------------------------------------------------------------------------------------------------------------------

Rule 1: Method overriding// signatures of overrriden and overriding methods should be same

-------------------------------------------------------------------------------------------------

class Parent

{

void marry()// overriden method

{

System.out.println("black girl");

}

}

class child extends Parent

{

void marry()//overriding method

{

System.out.println("white girl");

}

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

white girl

Example 2: if no overriding

-----------------------------

class Parent

{

void marry()

{

System.out.println("black girl");

}

}

class child extends Parent

{

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

black girl

Rule2: //the return type must be same at primitive level

------------------------------------------------

class Parent

{

int marry()//return type is int

{

System.out.println("black girl");

return 6.7f;

}

}

class child extends Parent

{

float marry()//return type is float

{

System.out.println("white girl");

return 6.7f;

}

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:6: error: incompatible types: possible lossy conversion from float to int

return 6.7f;

^

Parent.java:11: error: marry() in child cannot override marry() in Parent

float marry()

^

return type float is not compatible with int

Rule 3 : Covarient return type it is possible to change the return type //the child class return type is child and parent class return type is parent

-----------------------------------------------------------------------------------------------------------------

class Animal

{

}

class Dog extends Animal

{

}

class Parent

{

Animal marry()

{

System.out.println("black girl");

return new Animal();

}

}

class child extends Parent

{

Dog marry()

{

System.out.println("white girl");

return new Dog();

}

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

white girl

Rule 4: //final methods cannot be overriden

--------------------------------------------

class Parent

{

final void marry()

{

System.out.println("black girl");

}

}

class child extends Parent

{

void marry()

{

System.out.println("white girl");

}

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:11: error: marry() in child cannot override marry() in Parent

void marry()

^

overridden method is final

//Final

------------------------

//final classes cannot be extended

//final methods cannot be overridden

//final variables cannot be modified

new type of object creation

-----------------------------------------

class Parent

{

void m1()

{

System.out.println("parent method");

}

}

class child extends Parent

{

void m1()

{

System.out.println("child overrriding method");

}

void m2()

{

System.out.println("child direct method");

}

public static void main(String[] args)

{

Parent p = new child();

p.m1();// compile checks the parent class and executes overrriden method of child

p.m2();// compile time it checks the parent class

// since their is no m2 method in parent class

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:22: error: cannot find symbol

p.m2();

^

symbol: method m2()

location: variable p of type Parent

1 error

Example 2 :// child c = (child)p;

-----------------------------------------

class Parent

{

void m1()

{

System.out.println("parent method");

}

}

class child extends Parent

{

void m1()

{

System.out.println("child overrriding method");

}

void m2()

{

System.out.println("child direct method");

}

public static void main(String[] args)

{

Parent p = new child();

p.m1();// compile checks the parent class and run time executes method of child

p.m2();// compiler checks parent class and error

child c = (child)p;// to overcome the above problem we create this object

c.m2();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

child overrriding method

child direct method

// sellinium program // doubt

----------------------------------------------------

interface WebDriver

{

void get();

void window();

}

class FirefoxDriver implements WebDriver

{

void get();

void window();

void x();

void y();

WebDiver driver = new FirefoxDriver();

driver.get();

driver.window();

FirefoxDriver f = (FirefoxDriver)driver;

f.x();

f.y();

}

RULE 5 : Static methods are bonded with class

-----------------------------------------------

class Parent

{

static void m1()

{

System.out.println("parent method");

}

}

class child extends Parent

{

static void m1()

{

System.out.println("child overrriding method");

}

public static void main(String[] args)

{

Parent p = new child();

p.m1();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent method

Rule 6: //private methods are accesible within the class

-----------------------------------------------------------------

class Parent

{

private void m1()// acess only within inthe class

{

System.out.println("parent method");}

}

class child extends Parent

{

private void m2()

{

System.out.println("child method");

}

public static void main(String[] args)

{}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

Rule 7: modifiers

----------------------------------------

public-var methods,class//accessible in any package

protected- var, method //accesible within package and out only in the class

private - var,method//access within in the class

default - var,method,class//acess within the package

same level

---------------

class Parent

{

void m1()

{

System.out.println("parent method");}

}

class child extends Parent

{

void m1()

{

System.out.println("child method");

}

public static void main(String[] args)

{}

}

increase the level

----------------------------

class Parent

{

protected void m1()

{

System.out.println("parent method");}

}

class child extends Parent

{

public void m1()

{

System.out.println("child method");

}

public static void main(String[] args)

{}

}

decrease the level

-------------------------

class Parent

{

public void m1()

{

System.out.println("parent method");}

}

class child extends Parent

{

void m1()

{

System.out.println("child method");

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:10: error: m1() in child cannot override m1() in Parent

void m1()

^

attempting to assign weaker access privileges; was public

1 error

ABSTRACTION : the process of hiding the structure and highlighting the implimentations is called abstraction

----------------------------------------------------------------------------------------------------------------

we can achieve abstraction by using abstract and interface

1)normal methods and abstract methods

void m1(){}

in abstract

abstract void m1();

2) normal class and abstract class

class test

{

void m1(){}

void m2() {}

}

in abstract

abstract class test

{

void m1(){}

abstract void m3();{}

}

Example 1: abstract class hav some implimentations and some declarartions

---------------------------------------------------------------------------

abstract class Parent

{

abstract void m1();

abstract void m2();

void m3()

{

System.out.println("m3 method");

}

}

class child extends Parent

{

void m1()

{System.out.println("m1 method");}

void m2()

{System.out.println("m2 method");}

public static void main(String[] args)

{

child c = new child() ;

c.m1();

c.m2();

c.m3();

}}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

m1 method

m2 method

m3 method

Example 2:

-------------

abstract class Parent

{

abstract void m1();

abstract void m2();

void m3()

{

System.out.println("m3 method");

}

}

abstract class child extends Parent

{

void m1(){System.out.println("m1 method");}

}

class child1 extends child

{

void m2(){System.out.println("m2 method");}

public static void main(String[] args)

{

child1 c = new child1();

c.m1();

c.m2();

c.m3();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child1

m1 method

m2 method

m3 method

Example 3: vari,method.constructoer,insatnce and static blocks elements are executed in the absatract class also

object creation is not possible

----------------------

abstract class Parent

{

Parent()// parent class constructor

{System.out.println("parent class constructor");}

static {System.out.println("parent class static block");}//static block

}

class child extends Parent

{

{System.out.println("child class instance block");}// instance block

void m1()// child method

{System.out.println("m1 method");}

public static void main(String[] args)

{

child c = new child();

c.m1();

}}

E:\java\java practice programs>java child

parent class static block

parent class constructor

child class instance block

m1 method

PACKAGES :def: physical folder structure

types

--------------------------------------

Step 1 :- write the application with package statements

package com.sai.java.corejava;

class Testp

{

public static void main(String[] args)

{System.out.println("packages example 1");}

}

class A

{

}

class B

{

}

interface In

{

}

E:\java\java practice programs>javac Testp.java

E:\java\java practice programs>javac -d . Testp.java

Step 2:- compile using statement

javac -d . Testp.java

javac---------> java compiler

-d------------> create folder sttructure

. ------------> place the folder structure in current working directory

Testp.java----> file name

STEP 3:-it is placed as

com

--->sai

--->java

-----> core java

----------->Test.class

------------>A.class

------------->B.class

-------------->In.class

STEP 4:- Execution process

E:\java\java practice programs>java com.sai.java.corejava.Testp

packages example 1

Example 2:-

---------------------

error 1 : import

error 2: it is not public

Interface : it highlits the set of functionalities and implimenations classes impliments methods

-----------------------------------------------------------------------------------------------------

interface it// they are abstract so object creatioon is not possible

{void m1();//only abstract methods

// by default they are public and

}

Example 1 :-

-------------------------

interface It // abstract

{

void m1();// all methods in the interface are

//public and abstract

void m2();

void m3();

}

class Test implements It

{

public void m1(){System.out.println("m1 method");}

public void m2(){System.out.println("m2 method");}

public void m3(){System.out.println("m3 method");}

public static void main(String[] args)

{

Test t =new Test();

t.m1();

t.m2();

t.m3();

}

}

E:\java\java practice programs> javac It.java

E:\java\java practice programs>java Test

m1 method

m2 method

m3 method

Example 2:-// one method at a time

----------------------------------------

interface It // abstract

{

void m4();// all methods in the interface are

//public and abstract

void m5();

void m6();

}

abstract class Test implements It

{

public void m4(){System.out.println("m4 method");}

}

abstract class Test1 extends Test

{

public void m5(){System.out.println("m5 method");}

}

class Test2 extends Test1

{

public void m6(){System.out.println("m6 method");}

public static void main(String[] args)

{

Test2 t =new Test2();

t.m4();

t.m5();

t.m6();

}

}

E:\java\java practice programs>javac It.java

E:\java\java practice programs>java Test2

m4 method

m5 method

m6 method

INHERITANCE IN INTERFACE

-------------------------------------

class extends class

interface extends interface

class implements interface

class A extends B--------->valid

class A extends B,C-------->invalid

class A implements It------->valid

class A implements It1,It2------>valid

class A extends A --------------> invalid

interface It1 extends It2 ------->valid

interface It1 extends It2,IT3----->valid

interface It extends A ----------->invalid

interface It extends It---------> invalid

(extends keyword must be first keyword)

class A extends B impliments It1,It2-------> valid

class A impliments It1,It2 extends B ------->invalid

NESTED interface :- Definig interface inside interface is called nested

----------------------------------------------------------

interface Parent

{

interface Parent2

{void m1();

}

}

class Tests implements Parent.Parent2

{public void m1()

{

System.out.println("i love family");}

public static void main(String[] args)

{

new Tests().m1();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Tests

i love family

Example 2:- if the variable is same them their will be a problem while implimentations

--------------------------------------------------------------

interface Parent

{ int a=10;

}

interface Parent2

{int a=100;

}

class Tests implements Parent,Parent2

{public void m1()

{

System.out.println(a);}// conflit problems

public static void main(String[] args)

{

new Tests().m1();

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:10: error: reference to a is ambiguous

System.out.println(a);}

^

both variable a in Parent and variable a in Parent2 match

1 error

Example 3:-to over come the above problem

-----------------------------------------

interface Parent

{ int a=10;

}

interface Parent2

{int a=100;

}

class Tests implements Parent,Parent2

{public void m1()

{

System.out.println(Parent.a);

System.out.println(Parent2.a);}

public static void main(String[] args)

{

new Tests().m1();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Tests

10

100

Example 4:- Problem with this interface concepts so

to override the required methods their is a concept called adopterclass

-----------------------------------------------------------

adopterclass contains the empty implimentations of the interface

---------------------------------------------------------------

interface Parent

{ void m1();

void m2();

void m3();

void m4();

}

class X implements Parent // adopterclass

{ public void m1(){};//empty implimentations

public void m2(){};

public void m3(){};

public void m4(){};

}

class Tests implements Parent //child class for implimentations

{public void m1()

{System.out.println(" m1 method");}

public void m2()

{System.out.println("m2 method");}

public void m3()

{System.out.println("m3 method");}

public void m4()

{System.out.println("m4 method");}

public static void main(String[] args)

{

Tests t = new Tests();//object creation

t.m1();//method calling while creating an object

t.m2();

t.m3();

t.m4();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Tests

m1 method

m2 method

m3 method

m4 method

// javap is used to know the how many methods in particular class

E:\java\java practice programs>javap Parent

Compiled from "Parent.java"

interface Parent {

public abstract void m1();

public abstract void m2();

public abstract void m3();

public abstract void m4();}

MARKED INTERFACE :

----------------------------------------

javap package name gives empty methods but can acqiure somE CAPABILITIES from class this is called as marked interface

EXAMPLE

-----------

E:\java\java practice programs>javap java.io.Serializable

Compiled from "Serializable.java"

public interface java.io.Serializable {

}

For duplicate object creation we use cloning techinque This is used in marked interfaces to aquire capabilities

-------------------------------------------------------------------------------------------------------------------

Syntax : clone()

class Test implements Cloneable // this is cloneable is required to impliment the clone process

{

int a=10;

int b=20;

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

{

Test t = new Test();// object creation

System.out.println(t.a);

System.out.println(t.b);

/\*E:\java\java practice programs>javac Test.java

E:\java\java practice programs>java Test

10

20 \*/

t.a=100;

t.b=200;

System.out.println(t.a);

System.out.println(t.b);

/\* E:\java\java practice programs>javac Test.java

E:\java\java practice programs>java Test

10

20

100

200 \*/

Test t1 = (Test)t.clone();// duplicate object creation of 100 200

t.a=1000;

t.b=2000;

System.out.println(t.a);

System.out.println(t.b);

// here i want the values of 100 200

System.out.println(t1.a);//calling the duplicate objects of 100 200

System.out.println(t1.b);

}

}

/\*E:\java\java practice programs>javac Test.java

E:\java\java practice programs>java Test

10

20

100

200

1000

2000

100

200 \*/

REAL TIME EXAMPLE :

------------------------------------------

//interface : contains only declarations

package com.sai.java.realtime;

public interface Message

{ void morn();

void even();

void night();

}

// abstract class: contains method implimentions and declarations

package com.sai.java.abstrac;

import com.sai.java.realtime.Message;

public abstract class Helper implements Message

{

public void even()

{ System.out.println("good evening");}

}

// com[ilation of packages are done as follows

E:\java\java practice programs>javac -d . Message.java

E:\java\java practice programs>javac -d . Helper.java

E:\java\java practice programs>javac -d . TestClient1.java

E:\java\java practice programs>javac -d . TestClient2.java

// normal class : contains all the implimentions

package com.sai.java.TestClient1;//creating a new package

import com.sai.java.realtime.Message;// importing the interface

public class TestClient1 implements Message// class implements a interface

{

public void morn()

{ System.out.println("good morning");}

public void even()

{ System.out.println("gud evening");}

public void night()

{ System.out.println("good night");}

public static void main(String[] args)

{

TestClient1 t = new TestClient1();//object creation

t.morn();// method calling

t.even();

t.night();

}

}

/\*E:\java\java practice programs>java com.sai.java.TestClient1.TestClient1

good morning

gud evening

good night\*/

// normal class : contains all implimentations some in current class and some from extended class

package com.sai.java.TestClient2;// package creation

import com.sai.java.abstrac.Helper;// importing a class

public class TestClient2 extends Helper// class extends another class

{

public void morn()

{ System.out.println("good morning");}

public void night()

{ System.out.println("good night");}

public static void main(String[] args)

{

TestClient2 t = new TestClient2();// object creation

t.morn();

t.even();// imlimentations are done in parent class

t.night();

}

}

/\* E:\java\java practice programs>java com.sai.java.TestClient2.TestClient2

good morning

good evening

good night \*/

EXCEPTION: it is an event that terminates the event abnormally

whenever it happenss the rest application cannot be exected

---------------------------------------------------------------------

for normal termination of program and rest of application is executed normally is called exception handling

keywords: try ,cacth ,finally ,trow ,trows.

types : checked, unchecked , error exceptions

unchecked : the exceptions which are not checked by compiler but checked by runtime

so handle exception in 3 ways // try ,catch ,trows

CHECKED : the execptions that are caught by compiler nad code not compiled so use //

so use // try catch trows

FINALLY block: this is used to write the piece of code which is executed always irrespective of try catch blocks

---------------------------------------------------------------------

1)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println("try");}

catch ( ArithmeticException ae)

{System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>javac Testf.java

E:\java\java practice programs>java Testf

try

finally

2)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println(10/0);}

catch ( ArithmeticException ae)

{System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

catch

finally

3)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println(10/0);}

catch (NullPointerException ae)

{System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

finally

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.main(Testf.java:6)

4)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println(10/0);}

catch (ArithmeticException ae)

{System.out.println(10/0);}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

finally

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.main(Testf.java:8)

5)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println(10/0);}

catch (ArithmeticException ae)

{System.out.println("catch");}

finally

{System.out.println(10/0);}

}

}

E:\java\java practice programs>java Testf

catch

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.main(Testf.java:10)

6)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println("try");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

try

finally

whwnever we use system.exit(0); the finally block is not executed

------------------------------------------------------------------------

class Testf

{

public static void main(String[] args)

{

try

{System.out.println("try");

System.exit(0);

}

catch (ArithmeticException ae)

{ System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

try

2) finally can execute only if execution starts from tryblock only

-------------------------------------------------------------------

class Testf

{

public static void main(String[] args)

{ System.out.println(10/0);}

{

try

{System.out.println("try");

System.exit(0);

}

catch (ArithmeticException ae)

{ System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.main(Testf.java:4)

TO PRINT EXCEPTION INFO 3 methods

---------------------------------------

1) toString()

2)getMessage()

3)printStacktrace()

example 1:-

------------

class Testf

{

void m1()

{m2();} // m1 calling m2 method

void m2()

{m3();}// m2 calling m3 method

void m3()

{ try

{ System.out.println(10/0);}

catch (ArithmeticException ae)

{

System.out.println(ae.toString()); // 2nd important

System.out.println(ae.getMessage());

ae.printStackTrace(); // usually preferd

}

}

public static void main(String[] args)

{ new Testf().m1();

}

}

E:\java\java practice programs>java Testf

java.lang.ArithmeticException: / by zero // tostring()

/ by zero //getmessege()

java.lang.ArithmeticException: / by zero //printstacktrace()

at Testf.m3(Testf.java:9)

at Testf.m2(Testf.java:6)

at Testf.m1(Testf.java:4)

at Testf.main(Testf.java:19)

example2 : by default jvm uses printstacktrace()

------------------------------------------------

class Testf

{

void m1()

{m2();}

void m2()

{m3();}

void m3()

{ System.out.println(10/0);}

public static void main(String[] args)

{ new Testf().m1();

}

}

E:\java\java practice programs>java Testf // printstcaktrace()

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.m3(Testf.java:8)

at Testf.m2(Testf.java:6)

at Testf.m1(Testf.java:4)

at Testf.main(Testf.java:10)

TRY BLOCK : to write the exceptional code

CATCH BLOCK : to write the alternative code

FINALLY BLOCK : it is executed always irrespective of try catch

THROWS : It deligates the responcibilities of exception handling to caller method

THROW :it is used to handover the user created exception to JVM

TROWS() : It deligates the responcibilities of exception handling to caller method

--------------------------------------------------------------------------------

1)

class Testf

{void mro()throws InterruptedException

{System.out.println("sleeping");

Thread.sleep(3000);

System.out.println("working");

}

void moofficer() throws InterruptedException

{mro();}

void eseva() throws InterruptedException

{ moofficer();}

public static void main(String[] args)throws InterruptedException

{

Testf t = new Testf();

t.eseva();

}

}

E:\java\java practice programs>javac Testf.java

E:\java\java practice programs>java Testf

sleeping

working

2) one method can throw multpile exceptions at a time

----------------------------------------------------

class Testf

{void mro()throws InterruptedException

{System.out.println("sleeping");

Thread.sleep(3000); // 3 sec

System.out.println("working");

}

void moofficer() throws InterruptedException

{mro();}

void eseva()

{

try{ moofficer();}

catch(InterruptedException ie){ie.printStackTrace();}

}

public static void main(String[] args)

{

Testf t = new Testf();

t.eseva();

}

}

E:\java\java practice programs>java Testf

sleeping

working

STRING MANIPULATION :

-------------------------------------------------------

1) string : group of caracters or charachter array enclosed within doble codes // immutable

2)string bufferclass // mutable

3)tostring class

4)

STRING VS STRINGBUFFER

-------------------------------------

CASE 1) THESE ARE FINAL CLASSES

CASE 2) // 2 aproches to creat a string class and only one aproach for stringbuffer

class Test

{

public static void main(String[] args)

{

String s1 = "teja"; // without using new operator stored in scp memory (string constant pool) in this overriding is not possible

System.out.println(s1);

String s2 = new String("ajay");

System.out.println(s2);

StringBuffer sb1 = new StringBuffer ("feroz");//stringbuffer class using new operator stored in heap memory in this overriding is possible

System.out.println(sb1);

}

}

E:\java\java practice programs>java Test

teja

ajay

feroz

case 3) : string lo concatination it is "immutable", stringbuffer we use append "mutable"

class Test

{

public static void main(String[] args)

{

String s2 = new String("ajay");

s1 concatinate("good");

System.out.println(s2);

StringBuffer sb1 = new StringBuffer ("teja");

s1b append("bad");

System.out.println(sb1);

}

}

case 4) tostring() /\*this generally belongs to object which exeute hascode

first it test in curent class if tostring() not found it executes object class tostring() i.e., hashcode\*/

class Test

{

public static void main(String[] args)

{

Test t = new Test();

System.out.println(t);

System.out.println(t.toString());

}

}

E:\java\java practice programs>java Test

Test@15db9742 //

Test@15db9742

class Object

{ public String toString()

{

retrun "class-name@hashcode";

}

}

hashcode: is the unique identification number of object generated by jvm to identify object.

example 2: //to print the required datd we need to override the object class methods using tostring()

//this is general declaration of constructors

class Teste

{

int eid;

String ename;

Teste(int eid, String ename)

{

this.eid = eid;

this.ename = ename;

System.out.println(eid);

System.out.println(ename);

}

public static void main(String[] args)

{

Teste t = new Teste(216, "teja");

Teste t1 = new Teste(1110, "ajay");

Teste t2 = new Teste(013, "mom");

}

}

E:\java\java practice programs>java Teste

216

teja

1110

ajay

11

mom

class Teste

{

int eid;

String ename;

Teste(int eid, String ename)

{

this.eid = eid;

this.ename = ename;

System.out.println(eid);

System.out.println(ename);

}

public static void main(String[] args)

{

Teste t = new Teste(216, "teja");

Teste t1 = new Teste(1110, "ajay");

Teste t2 = new Teste(013, "mom");

System.out.println(t); // here it declares hashcode

// System.out.println(t.eid); // here it declares the value of eid

}

}

E:\java\java practice programs>java Teste

216

teja

1110

ajay

11

mom

Teste@15db9742

//to overcome this we can declare the tostring method

class Teste

{

int eid; //insatnce variables

String ename;

Teste(int eid, String ename) //constructor with 2 args

{

this.eid = eid;// declaring the local to insatnce variables

this.ename = ename;

}

// overriding the object class hascode

public String toString() // tosrting method which is declared as public

{

return "Teste eid="+eid+" Teste ename="+ename; //this is retuntype or output format

}

public static void main(String[] args)

{

Teste t = new Teste(216, "teja");

Teste t1 = new Teste(1110, "ajay");

System.out.println(t);

System.out.println(t1); //tostring method implimentations are executed

}

}

E:\java\java practice programs>java Teste

Teste eid=216 Teste ename=teja

Teste eid=1110 Teste ename=ajay

example 3: here no tostring method and only s1 or sib are declared but the output is not hashcode

this is possible is the "string objects or stringuffer objects" are created in the class

--------------------------------------------------

class Teste

{

public static void main(String[] args)

{

String s1 = "teja";

System.out.println(s1);

StringBuffer s1b = new StringBuffer ("ajay");

System.out.println(s1b);

}

}

E:\java\java practice programs>java Teste

teja

ajay

CASE 5: EQUALs : it returns boolean based on reference

------------------------------------------------------------

equalls() present in object class and object class return type is refence comparison

but string classs can overrride the equal method and return type is content refference

stringbuffer class cannot override the equal method so object class return type is execute ie.,ref comparison

class Teste

{

Teste(String str) {}// 1-arg constructor

public static void main(String[] args)

{

Teste t1 = new Teste("mom"); // constructor object creation

Teste t2 = new Teste("mom");//does not override equal method so object classequal method

System.out.println(t1.equals(t2));//// the return type is "referene comparision"

String s1 = "teja";

String s2 = "teja";// overides the equal method

System.out.println(s1.equals(s2));// return typr is "context reference"

StringBuffer s1b = new StringBuffer ("ajay");

StringBuffer s2b = new StringBuffer ("ajay");//does not override equal method so object classequal method

System.out.println(s1b.equals(s2b));// the return type is "referene comparision"

}

}

E:\java\java practice programs>javac Teste.java

E:\java\java practice programs>java Teste

false// heap memory

true//sccp memory (string constant pool)

false// heap memory

case 6: converstion of string to stringbuffer and stringbuffer to string

-------------------------------------------------------------------------

this string to stringbuffer because in java the reverse method is present only in the strningbuffer class not in the string class

example 1:

class Teste

{

public static void main(String[] args)

{

// string to stringbuffer because reverse is not present in string object

String str = "ratna";

StringBuffer sb = new StringBuffer(str);

System.out.println(sb.reverse());

// stringbuffer to string object the tostring() return type is string

StringBuffer sb1 = new StringBuffer("bhoomareddy");

String ss= sb1.toString();

System.out.println(ss);

}

}

E:\java\java practice programs>javac Teste.java

E:\java\java practice programs>java Teste

antar

bhoomareddy

COMPARE TO: its return type is integer value

-------------------------------------------------

if two strings are equall it returns "0" and if two strings not equall it returns either "+" or "-" based on unicode values

if ratan and anu the unicode value of r=115 and a=95 so it returns + and viceversa

if we have to compare ratna and ratan then check next character

equals() vs compareTo()

-------------------------------------

class Teste

{

public static void main(String[] args)

{

String s1 = "teja";

String s2 = "ajay";

String s3 = "teja";

System.out.println(s1.equals(s2));//false

System.out.println(s1.equals(s3));//true

System.out.println(s3.equals(s2));//false

System.out.println("TEJA".equals("teja"));//false

System.out.println("TEJA".equalsIgnoreCase("teja"));//true

System.out.println(s1.compareTo(s2));//+

System.out.println(s1.compareTo(s3));//+

System.out.println(s3.compareTo(s2));//0

System.out.println("TEJA".compareTo("teja"));//-

System.out.println("TEJA".compareToIgnoreCase("teja"));//0

}

}

E:\java\java practice programs>javac Teste.java

E:\java\java practice programs>java Teste

false

true

false

false

true

19

0

19

-32

0

equall() vs ==

-------------------------------

equal()is amethod present in object class whose return type is comparision refference

== is a assignment operator always used as refference

lenght() vs lenght

---------------------------------

lenght() is a method which used to find the length of the string

length is a variable which is used to find length of the array []

class Teste

{

public static void main(String[] args)

{

int [] b = new int[100];

System.out.println(b.Length);

String str = "ratan";

System.out.println(str.Length());

}

}

methods in string

-----------------------------------

class Teste

{

public static void main(String[] args)

{

String str = "ratan";

System.out.println(str.Index('a'));

System.out.println(str.LastIndex('a'));

System.out.println(str.charAt(3));

System.out.printlnIndex(str.lenght());

String str1 = "16021993";

System.out.println(str1.contains("02"));

System.out.println(str1.startswith("216"));

System.out.println(str1.endswith("993"));

System.out.println(str1.endswith("193"));

String str2 = " 10111994 "

System.out.println(str.trim());

System.out.println(str.trim().length());

}

}

practice programs  
-----------------  
  
basic application:  
  
class Tests   
{  
 public static void main(String[] args)   
 {  
 System.out.println("Hello World!");  
 }  
}  
E:\java\java practice programs>javac Tests.java  
  
E:\java\java practice programs>java Tests  
Hello World!  
  
example 2:instance method  
-------------------------  
  
class Tests1  
{  
 void m1()  
   
 {  
 int a=10;  
 System.out.println(a);  
 }  
  
 public static void main(String[] args)   
 {  
 Tests1 t = new Tests1(); // memory allocatee when objec is created  
  
 t.m1();  
   
 }  
}  
  
E:\java\java practice programs>javac Tests1.java  
  
E:\java\java practice programs>java Tests1  
10  
  
example 3 : static method   
--------------------------  
  
class Tests1  
{  
 static void m1()  
   
 {  
 int a=20;  
 System.out.println(a);  
 }  
  
 public static void main(String[] args)   
 {  
 Tests1.m1(); // memory is created when .class file is loading  
   
 }  
}  
  
E:\java\java practice programs>javac Tests1.java  
  
E:\java\java practice programs>java Tests1  
20  
  
example 4: instance and static variables and methods  
---------------------------------------------------  
  
class Tests1  
{  
 int a=10;//instance variables  
 int b=20;  
 static int c=30;//static variables  
 void m1()  
 {  
 System.out.println(a);  
 System.out.println(b);  
 System.out.println(Tests1.c);// classname.variable since it is the ststic variable  
  
 }  
 static void m2()  
 {  
 Tests1 t = new Tests1();  
 System.out.println(t.a);//This is static area so to execute instance variables we give it as   
 System.out.println(t.b); // reference variable.variable name  
 System.out.println(Tests1.c);  
 }  
 public static void main( String [] args)  
 {  
 Tests1 t = new Tests1();// This is for calling the void method  
 t.m1();  
 Tests1.m2();// this is for calling the static method  
 }  
}  
  
   
E:\java\java practice programs>javac Tests1.java  
  
E:\java\java practice programs>java Tests1  
10  
20  
30  
10  
20  
30  
  
 Example 5: constructors //1) used to write the logics of application. 2) to intitialize instance values.  
----------------------------------------------------------  
  
class Testsc   
{  
 void m1()  
 {  
 System.out.println("m1 method");  
 }  
 Testsc()  
 {  
 System.out.println("0-arg constructor");  
 }  
 Testsc(int a)  
 {  
 System.out.println("1-arg constructor");  
 }  
  
 public static void main(String[] args)   
 {  
 Testsc t = new Testsc();  
 Testsc t1 = new Testsc(10);  
 t.m1();  
   
 }  
}  
  
E:\java\java practice programs>javac Testsc.java  
  
E:\java\java practice programs>java Testsc  
0-arg constructor  
1-arg constructor  
m1 method  
  
Example 6: to call 1 constructor using another by "this method"  
---------------------------------------------------------------  
class Testsc   
{  
Testsc()  
 {  
 this(10); //indicates the 1 arg constructor  
 System.out.println("0 arg constuctor");  
 }  
 Testsc(int a )  
 {  
 System.out.println("1 arg constuctor");  
 }  
 public static void main(String[] args)  
 {  
 new Testsc();  
   
  
 }  
}  
E:\java\java practice programs>java Testsc  
1 arg constuctor  
0 arg constuctor  
  
Example 7: to assaign different values to different emp  
---------------------------------------------------------  
  
class Emp   
{ // instance variables  
 int eid;  
 String ename;  
 float esal;  
 Emp(int eid, String ename, float esal)//converstion of local values to instance values  
 {  
 this.eid=eid;  
 this.ename=ename;  
 this.esal=esal;}  
 void disp()  
 {  
 System.out.println("eid="+eid);  
 System.out.println("ename="+ename);  
 System.out.println("esal="+esal);  
 }  
 public static void main (String[] args)  
 {  
 Emp e1 = new Emp(111,"teja",11500.65f);  
 e1.disp();  
 Emp e2 = new Emp(222,"ajay",22120.73f);  
 e2.disp();  
 }  
   
}  
  
E:\java\java practice programs>java Emp  
eid=111  
ename=teja  
esal=11500.65  
eid=222  
ename=ajay  
esal=22120.73  
  
Example 8: Instance block  
-----------------------------  
  
class Testsi   
{  
 Testsi()  
 {  
 this(10);  
   
 System.out.println("o arg constructor");  
 }  
 Testsi(int a)  
 {  
 System.out.println("1 arg constructor");  
 }  
 {System.out.println("instance block");//instance bloclk  
 }  
 public static void main(String[] args)   
 {  
 new Testsi();  
 }  
}  
E:\java\java practice programs>java Testsi  
instance block  
1 arg constructor  
o arg constructor  
  
Example 9: the insatnce block is executed the no of times object is created  
------------------------------------------------------------  
  
class Testsi   
{  
 void m1() //insatnce method  
 {  
 System.out.println("m1 method");  
 }  
 Testsi()//constructor 1  
 {  
 this(10);  
   
 System.out.println("o arg constructor");  
 }  
 Testsi(int a) //constucter 2  
 {  
 System.out.println("1 arg constructor");  
 }  
 {System.out.println("instance block");//instance block  
 }  
 public static void main(String[] args)   
 {  
 new Testsi();//to call constructor  
 Testsi t1 = new Testsi();// to call instance method  
 t1.m1();  
 }  
}  
  
E:\java\java practice programs>java Testsi  
instance block  
1 arg constructor  
o arg constructor  
instance block  
1 arg constructor  
o arg constructor  
m1 method  
  
Example !0: to implement all keywords,variables,methods,constructors,blocks  
----------------------------------------------------------------  
  
class Testsi   
{  
 int a=20;// instance variables  
 static int b=30;//static variables  
 void m1(int a)//instance method  
 {  
 System.out.println("instance method");  
 }  
 static void m2(String str)//static method  
 {  
 System.out.println("static method");  
 }  
 Testsi()//constructor  
 {  
 this(10);//to call 2 coonstructor without creating a new object  
   
 System.out.println("o arg constructor");  
 }  
 Testsi(int a)// 2 constructor  
 {  
 System.out.println("1 arg constructor");  
 }  
 {System.out.println("instance block");//instance block  
 }  
 static // static block is executed only once for the class  
 {  
 System.out.println("static block");//static block  
 }  
 public static void main(String[] args)  
 {  
 new Testsi();//call constructor 1  
 Testsi t1 = new Testsi();//call instance method  
 t1.m1(40);  
 Testsi.m2("teja");//to call static method  
 }  
}  
output:  
---------  
static block  
instance block  
1 arg constructor  
o arg constructor  
instance block  
1 arg constructor  
o arg constructor  
instance method  
static method  
  
OOPS :1)inheritance 2)polymorphism 3)abstraction 4)encapsulation  
  
Example 11: Inheritance //the property of acquiring properties from one class to another is called inheritance  
----------------------------------------------------------------  
class parent  
{  
 int a=10;  
 int b=20;  
}  
class child extends Parent  
{  
 int x=100;  
 int y=200;  
 void add(int i, int j)  
 {  
 System.out.println(i+j);//local variables  
 System.out.println(x+y);//current   
 System.out.println(a+b);//super  
 }  
   
 public static void main(String[] args)   
 {  
 new child().add(1000,2000);  
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
3000  
300  
30  
  
Example 12: inheritance if all the variable names are same use "this" and "super" keywords  
----------------------------------------------------------------  
  
class Parent   
{  
 int a=10;  
 int b=20;  
}  
class child extends Parent  
{  
 int a=100;  
 int b=200;  
 void add(int a, int b)  
 {  
 System.out.println(a+b);  
 System.out.println(this.a+this.b);  
 System.out.println(super.a+super.b);  
 }  
   
 public static void main(String[] args)   
 {  
 new child().add(1000,2000);  
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
3000  
300  
30  
  
Example 13: parent methods in inheritance  
---------------------------------------------  
class Parent   
{  
void m1()  
 {  
 System.out.println("parent method");  
 }  
}  
class child extends Parent  
{  
 void m1()  
 {  
 System.out.println("child method");  
 }  
 void m2()  
 {  
 this.m1();  
 super.m1();  
System.out.println("child 2 method");  
 }  
   
 public static void main(String[] args)   
 {  
 new child().m2();  
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
child method  
parent method  
child 2 method  
  
Example 13: parent constructors in inheritance  
--------------------------------------------------  
class Parent   
{  
void m1()  
 {  
 System.out.println("parent method");  
 }  
 Parent()  
 {  
 System.out.println("0 arg constructor");  
 }  
}  
class child extends Parent  
{  
 void m1()  
 {  
 System.out.println("child method");  
 }  
 void m2()  
 {  
 this.m1();  
 super.m1();  
System.out.println("child 2 method");  
 }  
 child()  
 {  
 System.out.println("child constructor");  
 }  
   
 public static void main(String[] args)   
 {  
 new child().m2();//object created and method called  
 }  
}  
  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
0 arg constructor  
child constructor  
child method  
parent method  
child 2 method  
  
Exampele: important for real time  
---------------------------------------  
class Parent   
{  
 Parent()  
 {  
 System.out.println("parent constructor");  
 }  
}  
class child extends Parent  
{  
 child()// super(); compiler generated contructor  
 {  
 System.out.println("child constructor 1");  
 }  
 child(int a)//super();generated by compiler  
 {  
 System.out.println("child contructor 2");  
 }  
   
 public static void main(String[] args)   
 {  
 new child();//object creation  
 new child(10);//object creation  
 }  
}  
  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
parent constructor  
child constructor 1  
parent constructor  
child contructor 2  
  
Example 14:no child constructor then directly parent constructor is called  
---------------------------------------------------------------------------  
  
class Parent   
{  
 Parent()  
 {  
 System.out.println("parent constructor");  
 }  
}  
class child extends Parent  
{  
 /\*code is generated by compiler  
 child()  
 {  
 super();  
}  
 \*/  
 public static void main(String[] args)   
 {  
 new child();//object creation  
   
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
parent constructor  
  
Example 1: //Insatance blocks in inheritance  
----------------------------------------------  
  
class Parent   
{  
 {//insatnce block  
 System.out.println("parent class instance block");  
 }  
 Parent()//constructor  
 {  
 System.out.println("parent constructor");  
 }  
}  
class child extends Parent  
{  
 {//instance block  
 System.out.println("child class instance block");  
}  
child()//constructor  
 {  
 System.out.println("child constructor");  
 }  
 public static void main(String[] args)   
 {  
 new child();  
   
 }  
}  
  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
parent class instance block  
parent constructor  
child class instance block  
child constructor  
  
Example 1: //Static blocks in inheritance  
----------------------------------------------  
class Parent   
{  
 {  
 System.out.println("parent class instance block");  
 }  
 static  
 {  
 System.out.println("parent class static block");  
 }  
 Parent()  
 {  
 System.out.println("parent constructor");  
 }  
}  
class child extends Parent  
{  
 {  
 System.out.println("child class instance block");  
}  
static  
 {  
 System.out.println("child class static block");  
 }  
child()  
 {  
 System.out.println("child constructor");  
 }  
 public static void main(String[] args)   
 {  
 new child();  
   
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
parent class static block  
child class static block  
parent class instance block  
parent constructor  
child class instance block  
child constructor  
  
Example2: static block is executed once  
 insatnce blocks are executed the no.of times the objects are created  
----------------------------------------------------------------------------------  
  
  
class Parent   
{  
 {  
 System.out.println("parent class instance block");  
 }  
 static  
 {  
 System.out.println("parent class static block");  
 }  
 Parent()  
 {  
 System.out.println("parent constructor");  
 }  
}  
class child extends Parent  
{  
 {  
 System.out.println("child class instance block");  
}  
static  
 {  
 System.out.println("child class static block");  
 }  
child()  
 {  
 System.out.println("child constructor");  
 }  
 public static void main(String[] args)   
 {  
 new child();  
 new child();  
   
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
parent class static block  
child class static block  
parent class instance block  
parent constructor  
child class instance block  
child constructor  
parent class instance block  
parent constructor  
child class instance block  
child constructor  
  
// Polymorphysm : the abiliy to appear in many forms   
---------------------------------------------------  
1)Compile time polymorphysm  
2)run time polymorphysm  
  
IN the compile time polymorphysm   
OVERLOADING:// TO achieve this 1 class is required  
------------------------------------------------------  
 1) method overloading  
----------------------------------------------  
class Parent   
{  
 void m1(int a)//overloading method  
 {  
 System.out.println("int m1 method");  
 }  
 void m1(int a, int b)//same method name with diff no.of args  
 {  
 System.out.println("int, int m1 method");  
 }  
 void m1(char ch)//same method with diff data types  
 {  
 System.out.println("char m1 method");  
 }  
public static void main(String[] args)  
 {  
 Parent t = new Parent();  
 t.m1(10);  
 t.m1(10,20);  
 t.m1('T');  
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java Parent  
int m1 method  
int, int m1 method  
char m1 method  
  
Example: 2 constructor overloading  
-----------------------------------  
  
class Parent   
{  
 Parent(int a)  
 {  
 System.out.println("int arg constructor");  
 }  
 Parent(int a, int b)  
 {  
 System.out.println("int, int arg constructor");  
 }  
 Parent(char ch)  
 {  
 System.out.println("char arg constructor");  
 }  
public static void main(String[] args)  
 {  
 new Parent(10);  
 new Parent(10,20);  
 new Parent('T');  
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java Parent  
int arg constructor  
int, int arg constructor  
char arg constructor  
  
Example : 3 operator overloading only ("+") operator is supporteed by java  
------------------------------------------------------------------------  
class Parent  
{  
 public static void main( String[] args)  
 {  
 System.out.println(10+20);  
 System.out.println("amma"+"nanna");  
 System.out.println(100+"amma");  
 System.out.println(10+"amma"+"nanna"+10+20);  
 }  
  
  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java Parent  
30  
ammananna  
100amma  
10ammananna1020  
  
OVERRIDING: // to achieve the overriding concept 2 classes are required and should have the parent and child relation in inheritance  
---------------------------------------------------------------------------------------------------------------------------  
Rule 1: Method overriding// signatures of overrriden and overriding methods should be same  
-------------------------------------------------------------------------------------------------  
class Parent  
{  
 void marry()// overriden method  
 {  
 System.out.println("black girl");  
 }  
}  
 class child extends Parent  
 {  
 void marry()//overriding method  
 {  
 System.out.println("white girl");   
 }  
 public static void main( String[] args)  
 {  
 new child().marry();  
 }  
 }  
 E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
white girl  
Example 2: if no overriding  
-----------------------------  
class Parent  
{  
 void marry()  
 {  
 System.out.println("black girl");  
 }  
}  
 class child extends Parent  
 {  
   
 public static void main( String[] args)  
 {  
 new child().marry();  
 }  
 }  
   
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
black girl  
  
Rule2: //the return type must be same at primitive level  
------------------------------------------------  
class Parent  
{  
 int marry()//return type is int  
 {  
 System.out.println("black girl");  
 return 6.7f;  
 }  
}  
 class child extends Parent  
 {  
 float marry()//return type is float  
 {  
 System.out.println("white girl");  
 return 6.7f;  
 }  
 public static void main( String[] args)  
 {  
 new child().marry();  
 }  
 }  
   
E:\java\java practice programs>javac Parent.java  
Parent.java:6: error: incompatible types: possible lossy conversion from float to int  
 return 6.7f;  
 ^  
Parent.java:11: error: marry() in child cannot override marry() in Parent  
 float marry()  
 ^  
 return type float is not compatible with int  
  
 Rule 3 : Covarient return type it is possible to change the return type //the child class return type is child and parent class return type is parent  
 -----------------------------------------------------------------------------------------------------------------  
  
 class Animal  
{  
}  
class Dog extends Animal  
{  
}  
class Parent  
{  
 Animal marry()  
 {  
 System.out.println("black girl");  
 return new Animal();  
 }  
}  
 class child extends Parent  
 {  
 Dog marry()  
 {  
 System.out.println("white girl");  
 return new Dog();  
 }  
 public static void main( String[] args)  
 {  
 new child().marry();  
 }  
 }  
 E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
white girl  
  
Rule 4: //final methods cannot be overriden  
--------------------------------------------  
class Parent  
{  
 final void marry()  
 {  
 System.out.println("black girl");  
 }  
}  
 class child extends Parent  
 {  
 void marry()  
 {  
 System.out.println("white girl");  
 }  
 public static void main( String[] args)  
 {  
 new child().marry();  
 }  
 }  
 E:\java\java practice programs>javac Parent.java  
Parent.java:11: error: marry() in child cannot override marry() in Parent  
 void marry()  
 ^  
 overridden method is final  
   
 //Final  
 ------------------------  
 //final classes cannot be extended  
 //final methods cannot be overridden  
 //final variables cannot be modified  
  
 new type of object creation  
 -----------------------------------------  
  
 class Parent  
{  
 void m1()  
 {  
 System.out.println("parent method");  
 }  
}  
class child extends Parent  
{  
 void m1()  
 {  
 System.out.println("child overrriding method");  
 }  
 void m2()  
 {  
 System.out.println("child direct method");  
 }  
 public static void main(String[] args)  
 {  
 Parent p = new child();  
 p.m1();// compile checks the parent class and executes overrriden method of child  
 p.m2();// compile time it checks the parent class   
 // since their is no m2 method in parent class  
   
}  
}  
  
E:\java\java practice programs>javac Parent.java  
Parent.java:22: error: cannot find symbol  
 p.m2();  
 ^  
 symbol: method m2()  
 location: variable p of type Parent  
1 error  
  
Example 2 :// child c = (child)p;  
-----------------------------------------  
 class Parent  
{  
 void m1()  
 {  
 System.out.println("parent method");  
 }  
}  
class child extends Parent  
{  
 void m1()  
 {  
 System.out.println("child overrriding method");  
 }  
 void m2()  
 {  
 System.out.println("child direct method");  
 }  
 public static void main(String[] args)  
 {  
 Parent p = new child();  
 p.m1();// compile checks the parent class and run time executes method of child  
 p.m2();// compiler checks parent class and error   
 child c = (child)p;// to overcome the above problem we create this object  
 c.m2();  
}  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
child overrriding method  
child direct method  
  
// sellinium program // doubt  
----------------------------------------------------  
  
interface WebDriver  
{  
 void get();  
 void window();  
 }  
 class FirefoxDriver implements WebDriver  
 {   
 void get();  
 void window();  
 void x();  
 void y();  
  
 WebDiver driver = new FirefoxDriver();  
 driver.get();  
 driver.window();  
 FirefoxDriver f = (FirefoxDriver)driver;  
 f.x();  
 f.y();  
 }  
  
RULE 5 : Static methods are bonded with class   
-----------------------------------------------  
 class Parent  
{  
 static void m1()  
 {  
 System.out.println("parent method");  
 }  
}  
class child extends Parent  
{  
 static void m1()  
 {  
 System.out.println("child overrriding method");  
   
 }  
 public static void main(String[] args)  
 {  
 Parent p = new child();  
 p.m1();  
}  
}  
  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
parent method  
  
Rule 6: //private methods are accesible within the class  
-----------------------------------------------------------------  
 class Parent  
{  
 private void m1()// acess only within inthe class  
 {  
 System.out.println("parent method");}  
}  
   
class child extends Parent  
{  
 private void m2()  
 {  
 System.out.println("child method");  
}  
public static void main(String[] args)  
 {}  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child  
  
Rule 7: modifiers  
----------------------------------------  
public-var methods,class//accessible in any package  
protected- var, method //accesible within package and out only in the class  
private - var,method//access within in the class  
default - var,method,class//acess within the package  
  
same level  
---------------  
 class Parent  
{  
 void m1()  
 {  
 System.out.println("parent method");}  
}  
   
class child extends Parent  
{  
 void m1()  
 {  
 System.out.println("child method");  
}  
public static void main(String[] args)  
 {}  
}  
  
increase the level  
----------------------------  
class Parent  
{  
 protected void m1()  
 {  
 System.out.println("parent method");}  
}  
   
class child extends Parent  
{  
 public void m1()  
 {  
 System.out.println("child method");  
}  
public static void main(String[] args)  
 {}  
}  
  
decrease the level  
-------------------------  
 class Parent  
{  
 public void m1()  
 {  
 System.out.println("parent method");}  
}  
   
class child extends Parent  
{  
 void m1()  
 {  
 System.out.println("child method");  
}  
}  
E:\java\java practice programs>javac Parent.java  
Parent.java:10: error: m1() in child cannot override m1() in Parent  
 void m1()  
 ^  
 attempting to assign weaker access privileges; was public  
1 error  
  
ABSTRACTION : the process of hiding the structure and highlighting the implimentations is called abstraction  
----------------------------------------------------------------------------------------------------------------  
we can achieve abstraction by using abstract and interface   
1)normal methods and abstract methods  
void m1(){}  
in abstract   
abstract void m1();  
2) normal class and abstract class  
class test  
{  
 void m1(){}  
 void m2() {}  
}  
in abstract  
abstract class test  
{  
 void m1(){}  
 abstract void m3();{}  
}  
  
Example 1: abstract class hav some implimentations and some declarartions  
---------------------------------------------------------------------------  
abstract class Parent  
{  
 abstract void m1();  
 abstract void m2();  
 void m3()  
 {  
 System.out.println("m3 method");  
}  
}  
 class child extends Parent  
{   
 void m1()  
 {System.out.println("m1 method");}  
void m2()  
 {System.out.println("m2 method");}  
 public static void main(String[] args)  
 {  
 child c = new child() ;  
 c.m1();  
 c.m2();  
 c.m3();  
}}  
E:\java\java practice programs>javac Parent.java  
E:\java\java practice programs>java child  
m1 method  
m2 method  
m3 method  
  
  
Example 2:  
-------------  
abstract class Parent  
{  
 abstract void m1();  
 abstract void m2();  
 void m3()  
 {  
 System.out.println("m3 method");  
}  
}  
abstract class child extends Parent  
{   
 void m1(){System.out.println("m1 method");}  
}  
class child1 extends child  
{  
void m2(){System.out.println("m2 method");}  
 public static void main(String[] args)  
 {  
 child1 c = new child1();  
 c.m1();  
 c.m2();  
 c.m3();  
}  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java child1  
m1 method  
m2 method  
m3 method  
  
Example 3: vari,method.constructoer,insatnce and static blocks elements are executed in the absatract class also  
 object creation is not possible  
----------------------  
abstract class Parent  
{  
 Parent()// parent class constructor  
 {System.out.println("parent class constructor");}  
 static {System.out.println("parent class static block");}//static block  
}  
 class child extends Parent  
{  
{System.out.println("child class instance block");}// instance block  
void m1()// child method  
 {System.out.println("m1 method");}  
 public static void main(String[] args)  
 {  
 child c = new child();  
 c.m1();  
   
}}  
E:\java\java practice programs>java child  
parent class static block  
parent class constructor  
child class instance block  
m1 method  
  
PACKAGES :def: physical folder structure  
 types  
--------------------------------------  
Step 1 :- write the application with package statements  
  
package com.sai.java.corejava;  
class Testp  
{  
 public static void main(String[] args)  
 {System.out.println("packages example 1");}  
}   
class A  
{  
}  
class B  
{  
}  
interface In  
{  
}  
E:\java\java practice programs>javac Testp.java  
  
E:\java\java practice programs>javac -d . Testp.java  
  
Step 2:- compile using statement  
  
javac -d . Testp.java  
javac---------> java compiler  
-d------------> create folder sttructure  
. ------------> place the folder structure in current working directory  
Testp.java----> file name  
  
STEP 3:-it is placed as  
com  
 --->sai  
 --->java  
 -----> core java  
 ----------->Test.class  
 ------------>A.class   
 ------------->B.class  
 -------------->In.class   
STEP 4:- Execution process  
  
E:\java\java practice programs>java com.sai.java.corejava.Testp  
packages example 1  
  
Example 2:-  
---------------------  
error 1 : import  
error 2: it is not public  
  
Interface : it highlits the set of functionalities and implimenations classes impliments methods  
-----------------------------------------------------------------------------------------------------  
 interface it// they are abstract so object creatioon is not possible  
 {void m1();//only abstract methods  
 // by default they are public and  
 }  
  
 Example 1 :-  
 -------------------------  
interface It // abstract  
{  
 void m1();// all methods in the interface are   
 //public and abstract  
void m2();  
 void m3();  
}  
class Test implements It  
{  
 public void m1(){System.out.println("m1 method");}  
 public void m2(){System.out.println("m2 method");}  
 public void m3(){System.out.println("m3 method");}  
 public static void main(String[] args)  
 {  
 Test t =new Test();  
 t.m1();  
 t.m2();  
 t.m3();  
 }  
}  
E:\java\java practice programs> javac It.java  
  
E:\java\java practice programs>java Test  
m1 method  
m2 method  
m3 method  
  
Example 2:-// one method at a time  
----------------------------------------  
interface It // abstract  
{  
 void m4();// all methods in the interface are   
 //public and abstract  
void m5();  
 void m6();  
}  
abstract class Test implements It  
{  
 public void m4(){System.out.println("m4 method");}  
}  
abstract class Test1 extends Test  
{  
 public void m5(){System.out.println("m5 method");}  
}  
class Test2 extends Test1  
{  
 public void m6(){System.out.println("m6 method");}  
 public static void main(String[] args)  
 {  
 Test2 t =new Test2();  
 t.m4();  
 t.m5();  
 t.m6();  
 }  
}  
E:\java\java practice programs>javac It.java  
  
E:\java\java practice programs>java Test2  
m4 method  
m5 method  
m6 method  
  
INHERITANCE IN INTERFACE  
-------------------------------------  
class extends class  
interface extends interface  
class implements interface  
  
class A extends B--------->valid  
class A extends B,C-------->invalid  
class A implements It------->valid  
class A implements It1,It2------>valid  
class A extends A --------------> invalid  
  
interface It1 extends It2 ------->valid  
interface It1 extends It2,IT3----->valid  
interface It extends A ----------->invalid  
interface It extends It---------> invalid  
  
  
(extends keyword must be first keyword)  
class A extends B impliments It1,It2-------> valid  
class A impliments It1,It2 extends B ------->invalid  
  
NESTED interface :- Definig interface inside interface is called nested  
----------------------------------------------------------  
interface Parent  
{  
interface Parent2  
 {void m1();  
 }  
}  
class Tests implements Parent.Parent2  
{public void m1()  
 {  
 System.out.println("i love family");}  
 public static void main(String[] args)  
 {  
 new Tests().m1();  
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java Tests  
i love family  
  
Example 2:- if the variable is same them their will be a problem while implimentations  
--------------------------------------------------------------  
interface Parent  
{ int a=10;  
}  
interface Parent2  
 {int a=100;  
 }  
class Tests implements Parent,Parent2  
{public void m1()  
 {  
 System.out.println(a);}// conflit problems  
 public static void main(String[] args)  
 {  
 new Tests().m1();  
 }  
}  
E:\java\java practice programs>javac Parent.java  
Parent.java:10: error: reference to a is ambiguous  
 System.out.println(a);}  
 ^  
 both variable a in Parent and variable a in Parent2 match  
1 error  
  
Example 3:-to over come the above problem  
-----------------------------------------  
interface Parent  
{ int a=10;  
}  
interface Parent2  
 {int a=100;  
 }  
class Tests implements Parent,Parent2  
{public void m1()  
 {  
 System.out.println(Parent.a);  
 System.out.println(Parent2.a);}  
 public static void main(String[] args)  
 {  
 new Tests().m1();  
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java Tests  
10  
100  
  
Example 4:- Problem with this interface concepts so   
 to override the required methods their is a concept called adopterclass  
-----------------------------------------------------------  
adopterclass contains the empty implimentations of the interface  
---------------------------------------------------------------  
   
interface Parent  
{ void m1();  
void m2();  
void m3();  
void m4();  
}  
class X implements Parent // adopterclass   
{ public void m1(){};//empty implimentations  
public void m2(){};  
public void m3(){};  
 public void m4(){};  
}  
class Tests implements Parent //child class for implimentations  
{public void m1()  
 {System.out.println(" m1 method");}  
 public void m2()  
 {System.out.println("m2 method");}  
 public void m3()  
 {System.out.println("m3 method");}  
 public void m4()  
 {System.out.println("m4 method");}  
  
 public static void main(String[] args)  
 {  
 Tests t = new Tests();//object creation  
 t.m1();//method calling while creating an object  
 t.m2();  
 t.m3();  
 t.m4();  
 }  
}  
E:\java\java practice programs>javac Parent.java  
  
E:\java\java practice programs>java Tests  
 m1 method  
m2 method  
m3 method  
m4 method  
  
// javap is used to know the how many methods in particular class  
   
   
E:\java\java practice programs>javap Parent  
Compiled from "Parent.java"  
interface Parent {  
 public abstract void m1();  
 public abstract void m2();  
 public abstract void m3();  
 public abstract void m4();}  
  
 MARKED INTERFACE :  
 ----------------------------------------  
 javap package name gives empty methods but can acqiure somE CAPABILITIES from class this is called as marked interface   
 EXAMPLE  
 -----------  
 E:\java\java practice programs>javap java.io.Serializable  
Compiled from "Serializable.java"  
public interface java.io.Serializable {  
}  
  
For duplicate object creation we use cloning techinque This is used in marked interfaces to aquire capabilities  
-------------------------------------------------------------------------------------------------------------------  
Syntax : clone()  
  
class Test implements Cloneable // this is cloneable is required to impliment the clone process   
{  
 int a=10;  
 int b=20;  
public static void main(String[] args)throws Exception  
{  
 Test t = new Test();// object creation   
 System.out.println(t.a);  
 System.out.println(t.b);  
 /\*E:\java\java practice programs>javac Test.java  
E:\java\java practice programs>java Test  
10  
20 \*/  
 t.a=100;  
 t.b=200;  
 System.out.println(t.a);  
 System.out.println(t.b);  
 /\* E:\java\java practice programs>javac Test.java  
E:\java\java practice programs>java Test  
10  
20  
100  
200 \*/   
 Test t1 = (Test)t.clone();// duplicate object creation of 100 200  
 t.a=1000;  
 t.b=2000;  
 System.out.println(t.a);  
 System.out.println(t.b);  
 // here i want the values of 100 200  
 System.out.println(t1.a);//calling the duplicate objects of 100 200  
 System.out.println(t1.b);   
}  
}  
/\*E:\java\java practice programs>javac Test.java  
E:\java\java practice programs>java Test  
10  
20  
100  
200  
1000  
2000  
100  
200 \*/  
  
REAL TIME EXAMPLE :  
------------------------------------------  
//interface : contains only declarations  
package com.sai.java.realtime;  
public interface Message  
{ void morn();  
 void even();  
 void night();  
}  
// abstract class: contains method implimentions and declarations  
package com.sai.java.abstrac;  
import com.sai.java.realtime.Message;  
public abstract class Helper implements Message  
{  
 public void even()  
 { System.out.println("good evening");}  
}  
// com[ilation of packages are done as follows  
E:\java\java practice programs>javac -d . Message.java  
  
E:\java\java practice programs>javac -d . Helper.java  
  
E:\java\java practice programs>javac -d . TestClient1.java  
  
E:\java\java practice programs>javac -d . TestClient2.java  
  
// normal class : contains all the implimentions  
package com.sai.java.TestClient1;//creating a new package  
import com.sai.java.realtime.Message;// importing the interface  
public class TestClient1 implements Message// class implements a interface  
{  
public void morn()  
 { System.out.println("good morning");}  
public void even()  
 { System.out.println("gud evening");}  
public void night()  
 { System.out.println("good night");}  
 public static void main(String[] args)   
 {  
 TestClient1 t = new TestClient1();//object creation  
 t.morn();// method calling  
 t.even();  
 t.night();  
 }  
}  
 /\*E:\java\java practice programs>java com.sai.java.TestClient1.TestClient1  
good morning  
gud evening  
good night\*/  
  
// normal class : contains all implimentations some in current class and some from extended class  
package com.sai.java.TestClient2;// package creation  
import com.sai.java.abstrac.Helper;// importing a class  
public class TestClient2 extends Helper// class extends another class   
{  
 public void morn()  
 { System.out.println("good morning");}  
 public void night()  
 { System.out.println("good night");}   
 public static void main(String[] args)   
 {  
 TestClient2 t = new TestClient2();// object creation  
 t.morn();  
 t.even();// imlimentations are done in parent class  
 t.night();  
 }  
}  
/\* E:\java\java practice programs>java com.sai.java.TestClient2.TestClient2  
good morning  
good evening  
good night \*/  
  
EXCEPTION: it is an event that terminates the event abnormally   
 whenever it happenss the rest application cannot be exected  
---------------------------------------------------------------------  
for normal termination of program and rest of application is executed normally is called exception handling  
keywords: try ,cacth ,finally ,trow ,trows.  
types : checked, unchecked , error exceptions  
  
unchecked : the exceptions which are not checked by compiler but checked by runtime  
 so handle exception in 3 ways // try ,catch ,trows  
CHECKED : the execptions that are caught by compiler nad code not compiled so use //  
 so use // try catch trows  
FINALLY block: this is used to write the piece of code which is executed always irrespective of try catch blocks  
---------------------------------------------------------------------  
1)  
class Testf   
{  
 public static void main(String[] args)   
 {  
 try  
 {System.out.println("try");}  
 catch ( ArithmeticException ae)  
 {System.out.println("catch");}  
 finally  
 {System.out.println("finally");}  
 }  
}  
E:\java\java practice programs>javac Testf.java  
  
E:\java\java practice programs>java Testf  
try  
finally  
   
2)  
class Testf   
{  
 public static void main(String[] args)   
 {  
 try  
 {System.out.println(10/0);}  
 catch ( ArithmeticException ae)  
 {System.out.println("catch");}  
 finally  
 {System.out.println("finally");}  
 }  
}  
E:\java\java practice programs>java Testf  
catch  
finally  
  
3)  
class Testf   
{  
 public static void main(String[] args)   
 {  
 try  
 {System.out.println(10/0);}  
 catch (NullPointerException ae)  
 {System.out.println("catch");}  
 finally  
 {System.out.println("finally");}  
 }  
}  
  
E:\java\java practice programs>java Testf  
finally  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
 at Testf.main(Testf.java:6)  
  
4)  
class Testf   
{  
 public static void main(String[] args)   
 {  
 try  
 {System.out.println(10/0);}  
 catch (ArithmeticException ae)  
 {System.out.println(10/0);}  
 finally  
 {System.out.println("finally");}  
 }  
}  
E:\java\java practice programs>java Testf  
finally  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
 at Testf.main(Testf.java:8)  
5)  
  
class Testf   
{  
 public static void main(String[] args)   
 {  
 try  
 {System.out.println(10/0);}  
 catch (ArithmeticException ae)  
 {System.out.println("catch");}  
 finally  
 {System.out.println(10/0);}  
 }  
}  
E:\java\java practice programs>java Testf  
catch  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
 at Testf.main(Testf.java:10)  
6)  
  
class Testf   
{  
 public static void main(String[] args)   
 {  
 try  
 {System.out.println("try");}  
 finally  
 {System.out.println("finally");}  
 }  
}  
E:\java\java practice programs>java Testf  
try  
finally  
  
whwnever we use system.exit(0); the finally block is not executed  
------------------------------------------------------------------------  
class Testf   
{  
 public static void main(String[] args)   
 {  
 try  
 {System.out.println("try");  
 System.exit(0);  
 }  
 catch (ArithmeticException ae)  
 { System.out.println("catch");}  
 finally  
 {System.out.println("finally");}  
 }  
}  
E:\java\java practice programs>java Testf  
try  
  
2) finally can execute only if execution starts from tryblock only  
-------------------------------------------------------------------  
class Testf   
{  
 public static void main(String[] args)   
 { System.out.println(10/0);}  
 {  
 try  
 {System.out.println("try");  
 System.exit(0);  
 }  
 catch (ArithmeticException ae)  
 { System.out.println("catch");}  
 finally  
 {System.out.println("finally");}  
 }  
}  
E:\java\java practice programs>java Testf  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
 at Testf.main(Testf.java:4)  
  
TO PRINT EXCEPTION INFO 3 methods  
---------------------------------------  
1) toString()  
2)getMessage()  
3)printStacktrace()  
  
example 1:-  
------------  
  
class Testf   
{  
 void m1()  
 {m2();} // m1 calling m2 method  
 void m2()  
 {m3();}// m2 calling m3 method  
 void m3()  
 { try  
 { System.out.println(10/0);}  
 catch (ArithmeticException ae)  
 {  
 System.out.println(ae.toString()); // 2nd important  
 System.out.println(ae.getMessage());  
 ae.printStackTrace(); // usually preferd  
 }  
 }  
   
 public static void main(String[] args)  
 { new Testf().m1();  
 }  
 }  
   
E:\java\java practice programs>java Testf  
java.lang.ArithmeticException: / by zero // tostring()  
/ by zero //getmessege()  
java.lang.ArithmeticException: / by zero //printstacktrace()  
 at Testf.m3(Testf.java:9)  
 at Testf.m2(Testf.java:6)  
 at Testf.m1(Testf.java:4)  
 at Testf.main(Testf.java:19)  
  
  
example2 : by default jvm uses printstacktrace()   
------------------------------------------------  
class Testf   
{  
 void m1()  
 {m2();}  
 void m2()  
 {m3();}  
 void m3()  
 { System.out.println(10/0);}  
 public static void main(String[] args)  
 { new Testf().m1();  
 }  
}  
E:\java\java practice programs>java Testf // printstcaktrace()  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
 at Testf.m3(Testf.java:8)  
 at Testf.m2(Testf.java:6)  
 at Testf.m1(Testf.java:4)  
 at Testf.main(Testf.java:10)  
TRY BLOCK : to write the exceptional code  
CATCH BLOCK : to write the alternative code  
FINALLY BLOCK : it is executed always irrespective of try catch  
THROWS : It deligates the responcibilities of exception handling to caller method  
THROW :it is used to handover the user created exception to JVM  
  
TROWS() : It deligates the responcibilities of exception handling to caller method  
--------------------------------------------------------------------------------  
1)  
class Testf  
{void mro()throws InterruptedException  
 {System.out.println("sleeping");  
 Thread.sleep(3000);  
 System.out.println("working");  
 }  
 void moofficer() throws InterruptedException  
 {mro();}  
void eseva() throws InterruptedException  
 { moofficer();}  
   
public static void main(String[] args)throws InterruptedException  
 {  
Testf t = new Testf();  
t.eseva();  
}  
}  
E:\java\java practice programs>javac Testf.java  
  
E:\java\java practice programs>java Testf  
sleeping  
working  
  
2) one method can throw multpile exceptions at a time  
----------------------------------------------------  
class Testf  
{void mro()throws InterruptedException  
 {System.out.println("sleeping");  
 Thread.sleep(3000); // 3 sec  
 System.out.println("working");  
 }  
 void moofficer() throws InterruptedException  
 {mro();}  
void eseva()   
 {  
 try{ moofficer();}  
catch(InterruptedException ie){ie.printStackTrace();}  
 }  
   
public static void main(String[] args)  
 {  
Testf t = new Testf();  
t.eseva();  
}  
}  
  
E:\java\java practice programs>java Testf  
sleeping  
working  
  
STRING MANIPULATION :  
-------------------------------------------------------  
1) string : group of caracters or charachter array enclosed within doble codes // immutable  
2)string bufferclass // mutable  
3)tostring class   
4)   
 STRING VS STRINGBUFFER  
 -------------------------------------  
  
CASE 1) THESE ARE FINAL CLASSES  
CASE 2) // 2 aproches to creat a string class and only one aproach for stringbuffer   
  
class Test   
{  
 public static void main(String[] args)   
 {  
 String s1 = "teja"; // without using new operator stored in scp memory (string constant pool) in this overriding is not possible  
 System.out.println(s1);  
 String s2 = new String("ajay");   
 System.out.println(s2);  
 StringBuffer sb1 = new StringBuffer ("feroz");//stringbuffer class using new operator stored in heap memory in this overriding is possible  
 System.out.println(sb1);  
 }  
}  
E:\java\java practice programs>java Test  
teja  
ajay  
feroz  
  
case 3) : string lo concatination it is "immutable", stringbuffer we use append "mutable"  
  
class Test   
{  
 public static void main(String[] args)   
 {  
   
 String s2 = new String("ajay");  
 s1 concatinate("good");  
 System.out.println(s2);  
 StringBuffer sb1 = new StringBuffer ("teja");  
 s1b append("bad");  
 System.out.println(sb1);  
 }  
}  
  
case 4) tostring() /\*this generally belongs to object which exeute hascode  
 first it test in curent class if tostring() not found it executes object class tostring() i.e., hashcode\*/  
   
class Test   
{  
 public static void main(String[] args)   
 {  
 Test t = new Test();  
 System.out.println(t);  
 System.out.println(t.toString());  
 }  
}  
  
E:\java\java practice programs>java Test  
Test@15db9742 //   
Test@15db9742  
  
class Object   
{ public String toString()  
 {  
 retrun "class-name@hashcode";  
}  
}  
hashcode: is the unique identification number of object generated by jvm to identify object.  
  
example 2: //to print the required datd we need to override the object class methods using tostring()  
  
//this is general declaration of constructors  
class Teste   
{  
 int eid;  
 String ename;  
 Teste(int eid, String ename)  
 {  
 this.eid = eid;  
 this.ename = ename;  
 System.out.println(eid);  
 System.out.println(ename);  
  
 }  
  
 public static void main(String[] args)   
 {  
 Teste t = new Teste(216, "teja");  
 Teste t1 = new Teste(1110, "ajay");  
 Teste t2 = new Teste(013, "mom");  
   
}  
}  
E:\java\java practice programs>java Teste  
216  
teja  
1110  
ajay  
11  
mom  
  
class Teste   
{  
 int eid;  
 String ename;  
 Teste(int eid, String ename)  
 {  
 this.eid = eid;  
 this.ename = ename;  
 System.out.println(eid);  
 System.out.println(ename);  
  
 }  
  
 public static void main(String[] args)   
 {  
 Teste t = new Teste(216, "teja");  
 Teste t1 = new Teste(1110, "ajay");  
 Teste t2 = new Teste(013, "mom");  
 System.out.println(t); // here it declares hashcode  
 // System.out.println(t.eid); // here it declares the value of eid  
   
}  
}  
E:\java\java practice programs>java Teste  
216  
teja  
1110  
ajay  
11  
mom  
Teste@15db9742  
  
//to overcome this we can declare the tostring method  
class Teste   
{  
 int eid; //insatnce variables  
 String ename;  
 Teste(int eid, String ename) //constructor with 2 args  
 {  
 this.eid = eid;// declaring the local to insatnce variables   
 this.ename = ename;  
 }  
 // overriding the object class hascode  
 public String toString() // tosrting method which is declared as public  
 {  
 return "Teste eid="+eid+" Teste ename="+ename; //this is retuntype or output format  
 }  
  
 public static void main(String[] args)   
 {  
 Teste t = new Teste(216, "teja");  
 Teste t1 = new Teste(1110, "ajay");  
 System.out.println(t);  
 System.out.println(t1); //tostring method implimentations are executed  
}  
}  
  
E:\java\java practice programs>java Teste  
Teste eid=216 Teste ename=teja  
Teste eid=1110 Teste ename=ajay  
  
example 3: here no tostring method and only s1 or sib are declared but the output is not hashcode  
 this is possible is the "string objects or stringuffer objects" are created in the class   
--------------------------------------------------  
class Teste  
{  
 public static void main(String[] args)  
 {  
 String s1 = "teja";  
 System.out.println(s1);  
 StringBuffer s1b = new StringBuffer ("ajay");  
 System.out.println(s1b);  
  
 }  
}  
E:\java\java practice programs>java Teste  
teja  
ajay  
  
CASE 5: EQUALs : it returns boolean based on reference   
------------------------------------------------------------  
equalls() present in object class and object class return type is refence comparison  
 but string classs can overrride the equal method and return type is content refference  
 stringbuffer class cannot override the equal method so object class return type is execute ie.,ref comparison  
class Teste  
{  
 Teste(String str) {}// 1-arg constructor  
 public static void main(String[] args)  
 {  
 Teste t1 = new Teste("mom"); // constructor object creation  
 Teste t2 = new Teste("mom");//does not override equal method so object classequal method  
System.out.println(t1.equals(t2));//// the return type is "referene comparision"  
 String s1 = "teja";  
 String s2 = "teja";// overides the equal method  
 System.out.println(s1.equals(s2));// return typr is "context reference"  
 StringBuffer s1b = new StringBuffer ("ajay");  
 StringBuffer s2b = new StringBuffer ("ajay");//does not override equal method so object classequal method  
 System.out.println(s1b.equals(s2b));// the return type is "referene comparision"   
  
 }  
}  
E:\java\java practice programs>javac Teste.java  
  
E:\java\java practice programs>java Teste  
false// heap memory  
true//sccp memory (string constant pool)  
false// heap memory  
  
case 6: converstion of string to stringbuffer and stringbuffer to string  
-------------------------------------------------------------------------  
this string to stringbuffer because in java the reverse method is present only in the strningbuffer class not in the string class   
  
example 1:  
  
  
class Teste  
{  
 public static void main(String[] args)  
 {   
 // string to stringbuffer because reverse is not present in string object   
 String str = "ratna";  
 StringBuffer sb = new StringBuffer(str);  
 System.out.println(sb.reverse());  
 // stringbuffer to string object the tostring() return type is string  
 StringBuffer sb1 = new StringBuffer("bhoomareddy");  
 String ss= sb1.toString();  
 System.out.println(ss);  
 }  
}  
E:\java\java practice programs>javac Teste.java  
  
E:\java\java practice programs>java Teste  
antar  
bhoomareddy  
  
COMPARE TO: its return type is integer value  
-------------------------------------------------  
if two strings are equall it returns "0" and if two strings not equall it returns either "+" or "-" based on unicode values  
if ratan and anu the unicode value of r=115 and a=95 so it returns + and viceversa  
if we have to compare ratna and ratan then check next character  
  
equals() vs compareTo()  
-------------------------------------  
class Teste   
{  
 public static void main(String[] args)  
 {  
 String s1 = "teja";  
 String s2 = "ajay";  
 String s3 = "teja";  
 System.out.println(s1.equals(s2));//false  
 System.out.println(s1.equals(s3));//true  
 System.out.println(s3.equals(s2));//false  
 System.out.println("TEJA".equals("teja"));//false  
 System.out.println("TEJA".equalsIgnoreCase("teja"));//true  
   
 System.out.println(s1.compareTo(s2));//+  
 System.out.println(s1.compareTo(s3));//+  
 System.out.println(s3.compareTo(s2));//0  
 System.out.println("TEJA".compareTo("teja"));//-   
 System.out.println("TEJA".compareToIgnoreCase("teja"));//0   
}  
}  
E:\java\java practice programs>javac Teste.java  
  
E:\java\java practice programs>java Teste  
false  
true  
false  
false  
true  
19  
0  
19  
-32  
0  
  
equall() vs ==  
-------------------------------  
equal()is amethod present in object class whose return type is comparision refference  
== is a assignment operator always used as refference  
  
lenght() vs lenght  
---------------------------------  
lenght() is a method which used to find the length of the string   
length is a variable which is used to find length of the array []  
  
class Teste  
{  
 public static void main(String[] args)  
 {  
 int [] b = new int[100];  
 System.out.println(b.Length);  
 String str = "ratan";  
 System.out.println(str.Length());  
}  
}  
  
methods in string  
-----------------------------------  
class Teste  
{  
 public static void main(String[] args)  
 {  
 String str = "ratan";  
 System.out.println(str.Index('a'));  
 System.out.println(str.LastIndex('a'));  
  
 System.out.println(str.charAt(3));  
 System.out.printlnIndex(str.lenght());  
  
 String str1 = "16021993";  
 System.out.println(str1.contains("02"));  
 System.out.println(str1.startswith("216"));  
 System.out.println(str1.endswith("993"));  
 System.out.println(str1.endswith("193"));  
  
 String str2 = " 10111994 "  
 System.out.println(str.trim());  
 System.out.println(str.trim().length());  
}  
}

basic application:

class Tests

{

public static void main(String[] args)

{

System.out.println("Hello World!");

}

}

E:\java\java practice programs>javac Tests.java

E:\java\java practice programs>java Tests

Hello World!

example 2:instance method

-------------------------

class Tests1

{

void m1()

{

int a=10;

System.out.println(a);

}

public static void main(String[] args)

{

Tests1 t = new Tests1(); // memory allocatee when objec is created

t.m1();

}

}

E:\java\java practice programs>javac Tests1.java

E:\java\java practice programs>java Tests1

10

example 3 : static method

--------------------------

class Tests1

{

static void m1()

{

int a=20;

System.out.println(a);

}

public static void main(String[] args)

{

Tests1.m1(); // memory is created when .class file is loading

}

}

E:\java\java practice programs>javac Tests1.java

E:\java\java practice programs>java Tests1

20

example 4: instance and static variables and methods

---------------------------------------------------

class Tests1

{

int a=10;//instance variables

int b=20;

static int c=30;//static variables

void m1()

{

System.out.println(a);

System.out.println(b);

System.out.println(Tests1.c);// classname.variable since it is the ststic variable

}

static void m2()

{

Tests1 t = new Tests1();

System.out.println(t.a);//This is static area so to execute instance variables we give it as

System.out.println(t.b); // reference variable.variable name

System.out.println(Tests1.c);

}

public static void main( String [] args)

{

Tests1 t = new Tests1();// This is for calling the void method

t.m1();

Tests1.m2();// this is for calling the static method

}

}

E:\java\java practice programs>javac Tests1.java

E:\java\java practice programs>java Tests1

10

20

30

10

20

30

Example 5: constructors //1) used to write the logics of application. 2) to intitialize instance values.

----------------------------------------------------------

class Testsc

{

void m1()

{

System.out.println("m1 method");

}

Testsc()

{

System.out.println("0-arg constructor");

}

Testsc(int a)

{

System.out.println("1-arg constructor");

}

public static void main(String[] args)

{

Testsc t = new Testsc();

Testsc t1 = new Testsc(10);

t.m1();

}

}

E:\java\java practice programs>javac Testsc.java

E:\java\java practice programs>java Testsc

0-arg constructor

1-arg constructor

m1 method

Example 6: to call 1 constructor using another by "this method"

---------------------------------------------------------------

class Testsc

{

Testsc()

{

this(10); //indicates the 1 arg constructor

System.out.println("0 arg constuctor");

}

Testsc(int a )

{

System.out.println("1 arg constuctor");

}

public static void main(String[] args)

{

new Testsc();

}

}

E:\java\java practice programs>java Testsc

1 arg constuctor

0 arg constuctor

Example 7: to assaign different values to different emp

---------------------------------------------------------

class Emp

{ // instance variables

int eid;

String ename;

float esal;

Emp(int eid, String ename, float esal)//converstion of local values to instance values

{

this.eid=eid;

this.ename=ename;

this.esal=esal;}

void disp()

{

System.out.println("eid="+eid);

System.out.println("ename="+ename);

System.out.println("esal="+esal);

}

public static void main (String[] args)

{

Emp e1 = new Emp(111,"teja",11500.65f);

e1.disp();

Emp e2 = new Emp(222,"ajay",22120.73f);

e2.disp();

}

}

E:\java\java practice programs>java Emp

eid=111

ename=teja

esal=11500.65

eid=222

ename=ajay

esal=22120.73

Example 8: Instance block

-----------------------------

class Testsi

{

Testsi()

{

this(10);

System.out.println("o arg constructor");

}

Testsi(int a)

{

System.out.println("1 arg constructor");

}

{System.out.println("instance block");//instance bloclk

}

public static void main(String[] args)

{

new Testsi();

}

}

E:\java\java practice programs>java Testsi

instance block

1 arg constructor

o arg constructor

Example 9: the insatnce block is executed the no of times object is created

------------------------------------------------------------

class Testsi

{

void m1() //insatnce method

{

System.out.println("m1 method");

}

Testsi()//constructor 1

{

this(10);

System.out.println("o arg constructor");

}

Testsi(int a) //constucter 2

{

System.out.println("1 arg constructor");

}

{System.out.println("instance block");//instance block

}

public static void main(String[] args)

{

new Testsi();//to call constructor

Testsi t1 = new Testsi();// to call instance method

t1.m1();

}

}

E:\java\java practice programs>java Testsi

instance block

1 arg constructor

o arg constructor

instance block

1 arg constructor

o arg constructor

m1 method

Example !0: to implement all keywords,variables,methods,constructors,blocks

----------------------------------------------------------------

class Testsi

{

int a=20;// instance variables

static int b=30;//static variables

void m1(int a)//instance method

{

System.out.println("instance method");

}

static void m2(String str)//static method

{

System.out.println("static method");

}

Testsi()//constructor

{

this(10);//to call 2 coonstructor without creating a new object

System.out.println("o arg constructor");

}

Testsi(int a)// 2 constructor

{

System.out.println("1 arg constructor");

}

{System.out.println("instance block");//instance block

}

static // static block is executed only once for the class

{

System.out.println("static block");//static block

}

public static void main(String[] args)

{

new Testsi();//call constructor 1

Testsi t1 = new Testsi();//call instance method

t1.m1(40);

Testsi.m2("teja");//to call static method

}

}

output:

---------

static block

instance block

1 arg constructor

o arg constructor

instance block

1 arg constructor

o arg constructor

instance method

static method

OOPS :1)inheritance 2)polymorphism 3)abstraction 4)encapsulation

Example 11: Inheritance //the property of acquiring properties from one class to another is called inheritance

----------------------------------------------------------------

class parent

{

int a=10;

int b=20;

}

class child extends Parent

{

int x=100;

int y=200;

void add(int i, int j)

{

System.out.println(i+j);//local variables

System.out.println(x+y);//current

System.out.println(a+b);//super

}

public static void main(String[] args)

{

new child().add(1000,2000);

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

3000

300

30

Example 12: inheritance if all the variable names are same use "this" and "super" keywords

----------------------------------------------------------------

class Parent

{

int a=10;

int b=20;

}

class child extends Parent

{

int a=100;

int b=200;

void add(int a, int b)

{

System.out.println(a+b);

System.out.println(this.a+this.b);

System.out.println(super.a+super.b);

}

public static void main(String[] args)

{

new child().add(1000,2000);

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

3000

300

30

Example 13: parent methods in inheritance

---------------------------------------------

class Parent

{

void m1()

{

System.out.println("parent method");

}

}

class child extends Parent

{

void m1()

{

System.out.println("child method");

}

void m2()

{

this.m1();

super.m1();

System.out.println("child 2 method");

}

public static void main(String[] args)

{

new child().m2();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

child method

parent method

child 2 method

Example 13: parent constructors in inheritance

--------------------------------------------------

class Parent

{

void m1()

{

System.out.println("parent method");

}

Parent()

{

System.out.println("0 arg constructor");

}

}

class child extends Parent

{

void m1()

{

System.out.println("child method");

}

void m2()

{

this.m1();

super.m1();

System.out.println("child 2 method");

}

child()

{

System.out.println("child constructor");

}

public static void main(String[] args)

{

new child().m2();//object created and method called

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

0 arg constructor

child constructor

child method

parent method

child 2 method

Exampele: important for real time

---------------------------------------

class Parent

{

Parent()

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

child()// super(); compiler generated contructor

{

System.out.println("child constructor 1");

}

child(int a)//super();generated by compiler

{

System.out.println("child contructor 2");

}

public static void main(String[] args)

{

new child();//object creation

new child(10);//object creation

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent constructor

child constructor 1

parent constructor

child contructor 2

Example 14:no child constructor then directly parent constructor is called

---------------------------------------------------------------------------

class Parent

{

Parent()

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

/\*code is generated by compiler

child()

{

super();

}

\*/

public static void main(String[] args)

{

new child();//object creation

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent constructor

Example 1: //Insatance blocks in inheritance

----------------------------------------------

class Parent

{

{//insatnce block

System.out.println("parent class instance block");

}

Parent()//constructor

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

{//instance block

System.out.println("child class instance block");

}

child()//constructor

{

System.out.println("child constructor");

}

public static void main(String[] args)

{

new child();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent class instance block

parent constructor

child class instance block

child constructor

Example 1: //Static blocks in inheritance

----------------------------------------------

class Parent

{

{

System.out.println("parent class instance block");

}

static

{

System.out.println("parent class static block");

}

Parent()

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

{

System.out.println("child class instance block");

}

static

{

System.out.println("child class static block");

}

child()

{

System.out.println("child constructor");

}

public static void main(String[] args)

{

new child();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent class static block

child class static block

parent class instance block

parent constructor

child class instance block

child constructor

Example2: static block is executed once

insatnce blocks are executed the no.of times the objects are created

----------------------------------------------------------------------------------

class Parent

{

{

System.out.println("parent class instance block");

}

static

{

System.out.println("parent class static block");

}

Parent()

{

System.out.println("parent constructor");

}

}

class child extends Parent

{

{

System.out.println("child class instance block");

}

static

{

System.out.println("child class static block");

}

child()

{

System.out.println("child constructor");

}

public static void main(String[] args)

{

new child();

new child();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent class static block

child class static block

parent class instance block

parent constructor

child class instance block

child constructor

parent class instance block

parent constructor

child class instance block

child constructor

// Polymorphysm : the abiliy to appear in many forms

---------------------------------------------------

1)Compile time polymorphysm

2)run time polymorphysm

IN the compile time polymorphysm

OVERLOADING:// TO achieve this 1 class is required

------------------------------------------------------

1) method overloading

----------------------------------------------

class Parent

{

void m1(int a)//overloading method

{

System.out.println("int m1 method");

}

void m1(int a, int b)//same method name with diff no.of args

{

System.out.println("int, int m1 method");

}

void m1(char ch)//same method with diff data types

{

System.out.println("char m1 method");

}

public static void main(String[] args)

{

Parent t = new Parent();

t.m1(10);

t.m1(10,20);

t.m1('T');

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Parent

int m1 method

int, int m1 method

char m1 method

Example: 2 constructor overloading

-----------------------------------

class Parent

{

Parent(int a)

{

System.out.println("int arg constructor");

}

Parent(int a, int b)

{

System.out.println("int, int arg constructor");

}

Parent(char ch)

{

System.out.println("char arg constructor");

}

public static void main(String[] args)

{

new Parent(10);

new Parent(10,20);

new Parent('T');

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Parent

int arg constructor

int, int arg constructor

char arg constructor

Example : 3 operator overloading only ("+") operator is supporteed by java

------------------------------------------------------------------------

class Parent

{

public static void main( String[] args)

{

System.out.println(10+20);

System.out.println("amma"+"nanna");

System.out.println(100+"amma");

System.out.println(10+"amma"+"nanna"+10+20);

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Parent

30

ammananna

100amma

10ammananna1020

OVERRIDING: // to achieve the overriding concept 2 classes are required and should have the parent and child relation in inheritance

---------------------------------------------------------------------------------------------------------------------------

Rule 1: Method overriding// signatures of overrriden and overriding methods should be same

-------------------------------------------------------------------------------------------------

class Parent

{

void marry()// overriden method

{

System.out.println("black girl");

}

}

class child extends Parent

{

void marry()//overriding method

{

System.out.println("white girl");

}

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

white girl

Example 2: if no overriding

-----------------------------

class Parent

{

void marry()

{

System.out.println("black girl");

}

}

class child extends Parent

{

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

black girl

Rule2: //the return type must be same at primitive level

------------------------------------------------

class Parent

{

int marry()//return type is int

{

System.out.println("black girl");

return 6.7f;

}

}

class child extends Parent

{

float marry()//return type is float

{

System.out.println("white girl");

return 6.7f;

}

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:6: error: incompatible types: possible lossy conversion from float to int

return 6.7f;

^

Parent.java:11: error: marry() in child cannot override marry() in Parent

float marry()

^

return type float is not compatible with int

Rule 3 : Covarient return type it is possible to change the return type //the child class return type is child and parent class return type is parent

-----------------------------------------------------------------------------------------------------------------

class Animal

{

}

class Dog extends Animal

{

}

class Parent

{

Animal marry()

{

System.out.println("black girl");

return new Animal();

}

}

class child extends Parent

{

Dog marry()

{

System.out.println("white girl");

return new Dog();

}

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

white girl

Rule 4: //final methods cannot be overriden

--------------------------------------------

class Parent

{

final void marry()

{

System.out.println("black girl");

}

}

class child extends Parent

{

void marry()

{

System.out.println("white girl");

}

public static void main( String[] args)

{

new child().marry();

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:11: error: marry() in child cannot override marry() in Parent

void marry()

^

overridden method is final

//Final

------------------------

//final classes cannot be extended

//final methods cannot be overridden

//final variables cannot be modified

new type of object creation

-----------------------------------------

class Parent

{

void m1()

{

System.out.println("parent method");

}

}

class child extends Parent

{

void m1()

{

System.out.println("child overrriding method");

}

void m2()

{

System.out.println("child direct method");

}

public static void main(String[] args)

{

Parent p = new child();

p.m1();// compile checks the parent class and executes overrriden method of child

p.m2();// compile time it checks the parent class

// since their is no m2 method in parent class

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:22: error: cannot find symbol

p.m2();

^

symbol: method m2()

location: variable p of type Parent

1 error

Example 2 :// child c = (child)p;

-----------------------------------------

class Parent

{

void m1()

{

System.out.println("parent method");

}

}

class child extends Parent

{

void m1()

{

System.out.println("child overrriding method");

}

void m2()

{

System.out.println("child direct method");

}

public static void main(String[] args)

{

Parent p = new child();

p.m1();// compile checks the parent class and run time executes method of child

p.m2();// compiler checks parent class and error

child c = (child)p;// to overcome the above problem we create this object

c.m2();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

child overrriding method

child direct method

// sellinium program // doubt

----------------------------------------------------

interface WebDriver

{

void get();

void window();

}

class FirefoxDriver implements WebDriver

{

void get();

void window();

void x();

void y();

WebDiver driver = new FirefoxDriver();

driver.get();

driver.window();

FirefoxDriver f = (FirefoxDriver)driver;

f.x();

f.y();

}

RULE 5 : Static methods are bonded with class

-----------------------------------------------

class Parent

{

static void m1()

{

System.out.println("parent method");

}

}

class child extends Parent

{

static void m1()

{

System.out.println("child overrriding method");

}

public static void main(String[] args)

{

Parent p = new child();

p.m1();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

parent method

Rule 6: //private methods are accesible within the class

-----------------------------------------------------------------

class Parent

{

private void m1()// acess only within inthe class

{

System.out.println("parent method");}

}

class child extends Parent

{

private void m2()

{

System.out.println("child method");

}

public static void main(String[] args)

{}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

Rule 7: modifiers

----------------------------------------

public-var methods,class//accessible in any package

protected- var, method //accesible within package and out only in the class

private - var,method//access within in the class

default - var,method,class//acess within the package

same level

---------------

class Parent

{

void m1()

{

System.out.println("parent method");}

}

class child extends Parent

{

void m1()

{

System.out.println("child method");

}

public static void main(String[] args)

{}

}

increase the level

----------------------------

class Parent

{

protected void m1()

{

System.out.println("parent method");}

}

class child extends Parent

{

public void m1()

{

System.out.println("child method");

}

public static void main(String[] args)

{}

}

decrease the level

-------------------------

class Parent

{

public void m1()

{

System.out.println("parent method");}

}

class child extends Parent

{

void m1()

{

System.out.println("child method");

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:10: error: m1() in child cannot override m1() in Parent

void m1()

^

attempting to assign weaker access privileges; was public

1 error

ABSTRACTION : the process of hiding the structure and highlighting the implimentations is called abstraction

----------------------------------------------------------------------------------------------------------------

we can achieve abstraction by using abstract and interface

1)normal methods and abstract methods

void m1(){}

in abstract

abstract void m1();

2) normal class and abstract class

class test

{

void m1(){}

void m2() {}

}

in abstract

abstract class test

{

void m1(){}

abstract void m3();{}

}

Example 1: abstract class hav some implimentations and some declarartions

---------------------------------------------------------------------------

abstract class Parent

{

abstract void m1();

abstract void m2();

void m3()

{

System.out.println("m3 method");

}

}

class child extends Parent

{

void m1()

{System.out.println("m1 method");}

void m2()

{System.out.println("m2 method");}

public static void main(String[] args)

{

child c = new child() ;

c.m1();

c.m2();

c.m3();

}}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child

m1 method

m2 method

m3 method

Example 2:

-------------

abstract class Parent

{

abstract void m1();

abstract void m2();

void m3()

{

System.out.println("m3 method");

}

}

abstract class child extends Parent

{

void m1(){System.out.println("m1 method");}

}

class child1 extends child

{

void m2(){System.out.println("m2 method");}

public static void main(String[] args)

{

child1 c = new child1();

c.m1();

c.m2();

c.m3();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java child1

m1 method

m2 method

m3 method

Example 3: vari,method.constructoer,insatnce and static blocks elements are executed in the absatract class also

object creation is not possible

----------------------

abstract class Parent

{

Parent()// parent class constructor

{System.out.println("parent class constructor");}

static {System.out.println("parent class static block");}//static block

}

class child extends Parent

{

{System.out.println("child class instance block");}// instance block

void m1()// child method

{System.out.println("m1 method");}

public static void main(String[] args)

{

child c = new child();

c.m1();

}}

E:\java\java practice programs>java child

parent class static block

parent class constructor

child class instance block

m1 method

PACKAGES :def: physical folder structure

types

--------------------------------------

Step 1 :- write the application with package statements

package com.sai.java.corejava;

class Testp

{

public static void main(String[] args)

{System.out.println("packages example 1");}

}

class A

{

}

class B

{

}

interface In

{

}

E:\java\java practice programs>javac Testp.java

E:\java\java practice programs>javac -d . Testp.java

Step 2:- compile using statement

javac -d . Testp.java

javac---------> java compiler

-d------------> create folder sttructure

. ------------> place the folder structure in current working directory

Testp.java----> file name

STEP 3:-it is placed as

com

--->sai

--->java

-----> core java

----------->Test.class

------------>A.class

------------->B.class

-------------->In.class

STEP 4:- Execution process

E:\java\java practice programs>java com.sai.java.corejava.Testp

packages example 1

Example 2:-

---------------------

error 1 : import

error 2: it is not public

Interface : it highlits the set of functionalities and implimenations classes impliments methods

-----------------------------------------------------------------------------------------------------

interface it// they are abstract so object creatioon is not possible

{void m1();//only abstract methods

// by default they are public and

}

Example 1 :-

-------------------------

interface It // abstract

{

void m1();// all methods in the interface are

//public and abstract

void m2();

void m3();

}

class Test implements It

{

public void m1(){System.out.println("m1 method");}

public void m2(){System.out.println("m2 method");}

public void m3(){System.out.println("m3 method");}

public static void main(String[] args)

{

Test t =new Test();

t.m1();

t.m2();

t.m3();

}

}

E:\java\java practice programs> javac It.java

E:\java\java practice programs>java Test

m1 method

m2 method

m3 method

Example 2:-// one method at a time

----------------------------------------

interface It // abstract

{

void m4();// all methods in the interface are

//public and abstract

void m5();

void m6();

}

abstract class Test implements It

{

public void m4(){System.out.println("m4 method");}

}

abstract class Test1 extends Test

{

public void m5(){System.out.println("m5 method");}

}

class Test2 extends Test1

{

public void m6(){System.out.println("m6 method");}

public static void main(String[] args)

{

Test2 t =new Test2();

t.m4();

t.m5();

t.m6();

}

}

E:\java\java practice programs>javac It.java

E:\java\java practice programs>java Test2

m4 method

m5 method

m6 method

INHERITANCE IN INTERFACE

-------------------------------------

class extends class

interface extends interface

class implements interface

class A extends B--------->valid

class A extends B,C-------->invalid

class A implements It------->valid

class A implements It1,It2------>valid

class A extends A --------------> invalid

interface It1 extends It2 ------->valid

interface It1 extends It2,IT3----->valid

interface It extends A ----------->invalid

interface It extends It---------> invalid

(extends keyword must be first keyword)

class A extends B impliments It1,It2-------> valid

class A impliments It1,It2 extends B ------->invalid

NESTED interface :- Definig interface inside interface is called nested

----------------------------------------------------------

interface Parent

{

interface Parent2

{void m1();

}

}

class Tests implements Parent.Parent2

{public void m1()

{

System.out.println("i love family");}

public static void main(String[] args)

{

new Tests().m1();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Tests

i love family

Example 2:- if the variable is same them their will be a problem while implimentations

--------------------------------------------------------------

interface Parent

{ int a=10;

}

interface Parent2

{int a=100;

}

class Tests implements Parent,Parent2

{public void m1()

{

System.out.println(a);}// conflit problems

public static void main(String[] args)

{

new Tests().m1();

}

}

E:\java\java practice programs>javac Parent.java

Parent.java:10: error: reference to a is ambiguous

System.out.println(a);}

^

both variable a in Parent and variable a in Parent2 match

1 error

Example 3:-to over come the above problem

-----------------------------------------

interface Parent

{ int a=10;

}

interface Parent2

{int a=100;

}

class Tests implements Parent,Parent2

{public void m1()

{

System.out.println(Parent.a);

System.out.println(Parent2.a);}

public static void main(String[] args)

{

new Tests().m1();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Tests

10

100

Example 4:- Problem with this interface concepts so

to override the required methods their is a concept called adopterclass

-----------------------------------------------------------

adopterclass contains the empty implimentations of the interface

---------------------------------------------------------------

interface Parent

{ void m1();

void m2();

void m3();

void m4();

}

class X implements Parent // adopterclass

{ public void m1(){};//empty implimentations

public void m2(){};

public void m3(){};

public void m4(){};

}

class Tests implements Parent //child class for implimentations

{public void m1()

{System.out.println(" m1 method");}

public void m2()

{System.out.println("m2 method");}

public void m3()

{System.out.println("m3 method");}

public void m4()

{System.out.println("m4 method");}

public static void main(String[] args)

{

Tests t = new Tests();//object creation

t.m1();//method calling while creating an object

t.m2();

t.m3();

t.m4();

}

}

E:\java\java practice programs>javac Parent.java

E:\java\java practice programs>java Tests

m1 method

m2 method

m3 method

m4 method

// javap is used to know the how many methods in particular class

E:\java\java practice programs>javap Parent

Compiled from "Parent.java"

interface Parent {

public abstract void m1();

public abstract void m2();

public abstract void m3();

public abstract void m4();}

MARKED INTERFACE :

----------------------------------------

javap package name gives empty methods but can acqiure somE CAPABILITIES from class this is called as marked interface

EXAMPLE

-----------

E:\java\java practice programs>javap java.io.Serializable

Compiled from "Serializable.java"

public interface java.io.Serializable {

}

For duplicate object creation we use cloning techinque This is used in marked interfaces to aquire capabilities

-------------------------------------------------------------------------------------------------------------------

Syntax : clone()

class Test implements Cloneable // this is cloneable is required to impliment the clone process

{

int a=10;

int b=20;

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

{

Test t = new Test();// object creation

System.out.println(t.a);

System.out.println(t.b);

/\*E:\java\java practice programs>javac Test.java

E:\java\java practice programs>java Test

10

20 \*/

t.a=100;

t.b=200;

System.out.println(t.a);

System.out.println(t.b);

/\* E:\java\java practice programs>javac Test.java

E:\java\java practice programs>java Test

10

20

100

200 \*/

Test t1 = (Test)t.clone();// duplicate object creation of 100 200

t.a=1000;

t.b=2000;

System.out.println(t.a);

System.out.println(t.b);

// here i want the values of 100 200

System.out.println(t1.a);//calling the duplicate objects of 100 200

System.out.println(t1.b);

}

}

/\*E:\java\java practice programs>javac Test.java

E:\java\java practice programs>java Test

10

20

100

200

1000

2000

100

200 \*/

REAL TIME EXAMPLE :

------------------------------------------

//interface : contains only declarations

package com.sai.java.realtime;

public interface Message

{ void morn();

void even();

void night();

}

// abstract class: contains method implimentions and declarations

package com.sai.java.abstrac;

import com.sai.java.realtime.Message;

public abstract class Helper implements Message

{

public void even()

{ System.out.println("good evening");}

}

// com[ilation of packages are done as follows

E:\java\java practice programs>javac -d . Message.java

E:\java\java practice programs>javac -d . Helper.java

E:\java\java practice programs>javac -d . TestClient1.java

E:\java\java practice programs>javac -d . TestClient2.java

// normal class : contains all the implimentions

package com.sai.java.TestClient1;//creating a new package

import com.sai.java.realtime.Message;// importing the interface

public class TestClient1 implements Message// class implements a interface

{

public void morn()

{ System.out.println("good morning");}

public void even()

{ System.out.println("gud evening");}

public void night()

{ System.out.println("good night");}

public static void main(String[] args)

{

TestClient1 t = new TestClient1();//object creation

t.morn();// method calling

t.even();

t.night();

}

}

/\*E:\java\java practice programs>java com.sai.java.TestClient1.TestClient1

good morning

gud evening

good night\*/

// normal class : contains all implimentations some in current class and some from extended class

package com.sai.java.TestClient2;// package creation

import com.sai.java.abstrac.Helper;// importing a class

public class TestClient2 extends Helper// class extends another class

{

public void morn()

{ System.out.println("good morning");}

public void night()

{ System.out.println("good night");}

public static void main(String[] args)

{

TestClient2 t = new TestClient2();// object creation

t.morn();

t.even();// imlimentations are done in parent class

t.night();

}

}

/\* E:\java\java practice programs>java com.sai.java.TestClient2.TestClient2

good morning

good evening

good night \*/

EXCEPTION: it is an event that terminates the event abnormally

whenever it happenss the rest application cannot be exected

---------------------------------------------------------------------

for normal termination of program and rest of application is executed normally is called exception handling

keywords: try ,cacth ,finally ,trow ,trows.

types : checked, unchecked , error exceptions

unchecked : the exceptions which are not checked by compiler but checked by runtime

so handle exception in 3 ways // try ,catch ,trows

CHECKED : the execptions that are caught by compiler nad code not compiled so use //

so use // try catch trows

FINALLY block: this is used to write the piece of code which is executed always irrespective of try catch blocks

---------------------------------------------------------------------

1)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println("try");}

catch ( ArithmeticException ae)

{System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>javac Testf.java

E:\java\java practice programs>java Testf

try

finally

2)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println(10/0);}

catch ( ArithmeticException ae)

{System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

catch

finally

3)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println(10/0);}

catch (NullPointerException ae)

{System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

finally

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.main(Testf.java:6)

4)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println(10/0);}

catch (ArithmeticException ae)

{System.out.println(10/0);}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

finally

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.main(Testf.java:8)

5)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println(10/0);}

catch (ArithmeticException ae)

{System.out.println("catch");}

finally

{System.out.println(10/0);}

}

}

E:\java\java practice programs>java Testf

catch

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.main(Testf.java:10)

6)

class Testf

{

public static void main(String[] args)

{

try

{System.out.println("try");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

try

finally

whwnever we use system.exit(0); the finally block is not executed

------------------------------------------------------------------------

class Testf

{

public static void main(String[] args)

{

try

{System.out.println("try");

System.exit(0);

}

catch (ArithmeticException ae)

{ System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

try

2) finally can execute only if execution starts from tryblock only

-------------------------------------------------------------------

class Testf

{

public static void main(String[] args)

{ System.out.println(10/0);}

{

try

{System.out.println("try");

System.exit(0);

}

catch (ArithmeticException ae)

{ System.out.println("catch");}

finally

{System.out.println("finally");}

}

}

E:\java\java practice programs>java Testf

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.main(Testf.java:4)

TO PRINT EXCEPTION INFO 3 methods

---------------------------------------

1) toString()

2)getMessage()

3)printStacktrace()

example 1:-

------------

class Testf

{

void m1()

{m2();} // m1 calling m2 method

void m2()

{m3();}// m2 calling m3 method

void m3()

{ try

{ System.out.println(10/0);}

catch (ArithmeticException ae)

{

System.out.println(ae.toString()); // 2nd important

System.out.println(ae.getMessage());

ae.printStackTrace(); // usually preferd

}

}

public static void main(String[] args)

{ new Testf().m1();

}

}

E:\java\java practice programs>java Testf

java.lang.ArithmeticException: / by zero // tostring()

/ by zero //getmessege()

java.lang.ArithmeticException: / by zero //printstacktrace()

at Testf.m3(Testf.java:9)

at Testf.m2(Testf.java:6)

at Testf.m1(Testf.java:4)

at Testf.main(Testf.java:19)

example2 : by default jvm uses printstacktrace()

------------------------------------------------

class Testf

{

void m1()

{m2();}

void m2()

{m3();}

void m3()

{ System.out.println(10/0);}

public static void main(String[] args)

{ new Testf().m1();

}

}

E:\java\java practice programs>java Testf // printstcaktrace()

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Testf.m3(Testf.java:8)

at Testf.m2(Testf.java:6)

at Testf.m1(Testf.java:4)

at Testf.main(Testf.java:10)

TRY BLOCK : to write the exceptional code

CATCH BLOCK : to write the alternative code

FINALLY BLOCK : it is executed always irrespective of try catch

THROWS : It deligates the responcibilities of exception handling to caller method

THROW :it is used to handover the user created exception to JVM

TROWS() : It deligates the responcibilities of exception handling to caller method

--------------------------------------------------------------------------------

1)

class Testf

{void mro()throws InterruptedException

{System.out.println("sleeping");

Thread.sleep(3000);

System.out.println("working");

}

void moofficer() throws InterruptedException

{mro();}

void eseva() throws InterruptedException

{ moofficer();}

public static void main(String[] args)throws InterruptedException

{

Testf t = new Testf();

t.eseva();

}

}

E:\java\java practice programs>javac Testf.java

E:\java\java practice programs>java Testf

sleeping

working

2) one method can throw multpile exceptions at a time

----------------------------------------------------

class Testf

{void mro()throws InterruptedException

{System.out.println("sleeping");

Thread.sleep(3000); // 3 sec

System.out.println("working");

}

void moofficer() throws InterruptedException

{mro();}

void eseva()

{

try{ moofficer();}

catch(InterruptedException ie){ie.printStackTrace();}

}

public static void main(String[] args)

{

Testf t = new Testf();

t.eseva();

}

}

E:\java\java practice programs>java Testf

sleeping

working

STRING MANIPULATION :

-------------------------------------------------------

1) string : group of caracters or charachter array enclosed within doble codes // immutable

2)string bufferclass // mutable

3)tostring class

4)

STRING VS STRINGBUFFER

-------------------------------------

CASE 1) THESE ARE FINAL CLASSES

CASE 2) // 2 aproches to creat a string class and only one aproach for stringbuffer

class Test

{

public static void main(String[] args)

{

String s1 = "teja"; // without using new operator stored in scp memory (string constant pool) in this overriding is not possible

System.out.println(s1);

String s2 = new String("ajay");

System.out.println(s2);

StringBuffer sb1 = new StringBuffer ("feroz");//stringbuffer class using new operator stored in heap memory in this overriding is possible

System.out.println(sb1);

}

}

E:\java\java practice programs>java Test

teja

ajay

feroz

case 3) : string lo concatination it is "immutable", stringbuffer we use append "mutable"

class Test

{

public static void main(String[] args)

{

String s2 = new String("ajay");

s1 concatinate("good");

System.out.println(s2);

StringBuffer sb1 = new StringBuffer ("teja");

s1b append("bad");

System.out.println(sb1);

}

}

case 4) tostring() /\*this generally belongs to object which exeute hascode

first it test in curent class if tostring() not found it executes object class tostring() i.e., hashcode\*/

class Test

{

public static void main(String[] args)

{

Test t = new Test();

System.out.println(t);

System.out.println(t.toString());

}

}

E:\java\java practice programs>java Test

Test@15db9742 //

Test@15db9742

class Object

{ public String toString()

{

retrun "class-name@hashcode";

}

}

hashcode: is the unique identification number of object generated by jvm to identify object.

example 2: //to print the required datd we need to override the object class methods using tostring()

//this is general declaration of constructors

class Teste

{

int eid;

String ename;

Teste(int eid, String ename)

{

this.eid = eid;

this.ename = ename;

System.out.println(eid);

System.out.println(ename);

}

public static void main(String[] args)

{

Teste t = new Teste(216, "teja");

Teste t1 = new Teste(1110, "ajay");

Teste t2 = new Teste(013, "mom");

}

}

E:\java\java practice programs>java Teste

216

teja

1110

ajay

11

mom

class Teste

{

int eid;

String ename;

Teste(int eid, String ename)

{

this.eid = eid;

this.ename = ename;

System.out.println(eid);

System.out.println(ename);

}

public static void main(String[] args)

{

Teste t = new Teste(216, "teja");

Teste t1 = new Teste(1110, "ajay");

Teste t2 = new Teste(013, "mom");

System.out.println(t); // here it declares hashcode

// System.out.println(t.eid); // here it declares the value of eid

}

}

E:\java\java practice programs>java Teste

216

teja

1110

ajay

11

mom

Teste@15db9742

//to overcome this we can declare the tostring method

class Teste

{

int eid; //insatnce variables

String ename;

Teste(int eid, String ename) //constructor with 2 args

{

this.eid = eid;// declaring the local to insatnce variables

this.ename = ename;

}

// overriding the object class hascode

public String toString() // tosrting method which is declared as public

{

return "Teste eid="+eid+" Teste ename="+ename; //this is retuntype or output format

}

public static void main(String[] args)

{

Teste t = new Teste(216, "teja");

Teste t1 = new Teste(1110, "ajay");

System.out.println(t);

System.out.println(t1); //tostring method implimentations are executed

}

}

E:\java\java practice programs>java Teste

Teste eid=216 Teste ename=teja

Teste eid=1110 Teste ename=ajay

example 3: here no tostring method and only s1 or sib are declared but the output is not hashcode

this is possible is the "string objects or stringuffer objects" are created in the class

--------------------------------------------------

class Teste

{

public static void main(String[] args)

{

String s1 = "teja";

System.out.println(s1);

StringBuffer s1b = new StringBuffer ("ajay");

System.out.println(s1b);

}

}

E:\java\java practice programs>java Teste

teja

ajay

CASE 5: EQUALs : it returns boolean based on reference

------------------------------------------------------------

equalls() present in object class and object class return type is refence comparison

but string classs can overrride the equal method and return type is content refference

stringbuffer class cannot override the equal method so object class return type is execute ie.,ref comparison

class Teste

{

Teste(String str) {}// 1-arg constructor

public static void main(String[] args)

{

Teste t1 = new Teste("mom"); // constructor object creation

Teste t2 = new Teste("mom");//does not override equal method so object classequal method

System.out.println(t1.equals(t2));//// the return type is "referene comparision"

String s1 = "teja";

String s2 = "teja";// overides the equal method

System.out.println(s1.equals(s2));// return typr is "context reference"

StringBuffer s1b = new StringBuffer ("ajay");

StringBuffer s2b = new StringBuffer ("ajay");//does not override equal method so object classequal method

System.out.println(s1b.equals(s2b));// the return type is "referene comparision"

}

}

E:\java\java practice programs>javac Teste.java

E:\java\java practice programs>java Teste

false// heap memory

true//sccp memory (string constant pool)

false// heap memory

case 6: converstion of string to stringbuffer and stringbuffer to string

-------------------------------------------------------------------------

this string to stringbuffer because in java the reverse method is present only in the strningbuffer class not in the string class

example 1:

class Teste

{

public static void main(String[] args)

{

// string to stringbuffer because reverse is not present in string object

String str = "ratna";

StringBuffer sb = new StringBuffer(str);

System.out.println(sb.reverse());

// stringbuffer to string object the tostring() return type is string

StringBuffer sb1 = new StringBuffer("bhoomareddy");

String ss= sb1.toString();

System.out.println(ss);

}

}

E:\java\java practice programs>javac Teste.java

E:\java\java practice programs>java Teste

antar

bhoomareddy

COMPARE TO: its return type is integer value

-------------------------------------------------

if two strings are equall it returns "0" and if two strings not equall it returns either "+" or "-" based on unicode values

if ratan and anu the unicode value of r=115 and a=95 so it returns + and viceversa

if we have to compare ratna and ratan then check next character

equals() vs compareTo()

-------------------------------------

class Teste

{

public static void main(String[] args)

{

String s1 = "teja";

String s2 = "ajay";

String s3 = "teja";

System.out.println(s1.equals(s2));//false

System.out.println(s1.equals(s3));//true

System.out.println(s3.equals(s2));//false

System.out.println("TEJA".equals("teja"));//false

System.out.println("TEJA".equalsIgnoreCase("teja"));//true

System.out.println(s1.compareTo(s2));//+

System.out.println(s1.compareTo(s3));//+

System.out.println(s3.compareTo(s2));//0

System.out.println("TEJA".compareTo("teja"));//-

System.out.println("TEJA".compareToIgnoreCase("teja"));//0

}

}

E:\java\java practice programs>javac Teste.java

E:\java\java practice programs>java Teste

false

true

false

false

true

19

0

19

-32

0

equall() vs ==

-------------------------------

equal()is amethod present in object class whose return type is comparision refference

== is a assignment operator always used as refference

lenght() vs lenght

---------------------------------

lenght() is a method which used to find the length of the string

length is a variable which is used to find length of the array []

class Teste

{

public static void main(String[] args)

{

int [] b = new int[100];

System.out.println(b.Length);

String str = "ratan";

System.out.println(str.Length());

}

}

methods in string

-----------------------------------

class Teste

{

public static void main(String[] args)

{

String str = "ratan";

System.out.println(str.Index('a'));

System.out.println(str.LastIndex('a'));

System.out.println(str.charAt(3));

System.out.printlnIndex(str.lenght());

String str1 = "16021993";

System.out.println(str1.contains("02"));

System.out.println(str1.startswith("216"));

System.out.println(str1.endswith("993"));

System.out.println(str1.endswith("193"));

String str2 = " 10111994 "

System.out.println(str.trim());

System.out.println(str.trim().length());

}

}