**Java**

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

- Java is programming language was developed by the sun microsystem in 1995, now it is part of oracle.

- Java is Object Oriented Programming language

- In Java everything is an object.

- Java is platform independent languages (We can same programs in Window OS, Mac OS or Unix OS).

- Java is simple and easy to understand.

- Java is secure language

- Using Java we can developer different types of application

1) Desktop Based Application

2) Web Based application

3) Mobile Based Application

4) Embedded Based Application

5) Smart Carts

6) Games Application

7) Test Automation

- Java Syntax Rules

- Java is case Sensitive Language.

- First letter of class name must be capital.

- Method name should be start with small letter

- Java program file name should be exactly match with class name.

- Java program execution starts with main method.

- Every java statement ends with semicolon (;)

- Every java code enclosed with { }

Datatypes in Java

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

- it is used to store the values.

there are 2 main datatypes in java

1) Primitive Datatype

2) Non Primitive Datatype

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

1) Primitive Datatype

there are 8 types of primitive datatypes

1) byte

2) short

3) int

4) long

5) float

6) double

7) boolean

8) char

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

if u want to store numbers in java program

1) byte

2) short

3) int

4) long

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

1) byte

- it is used to store numbers between -128 to 127 values.

syntax:

datatype variablename=value;

e.g

byte abc=10;

System.out.println(abc);

byte datatype

abc variable name

10 is value for abc variable

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

package Tutorial2;

public class Demo1 {

public static void main(String[] args) {

byte abc=20;

System.out.println(abc);

}

}

===============================================

package Tutorial2;

public class Demo2 {

public static void main(String[] args) {

byte pune=-50;

System.out.println(pune);

}

}

===========================================================

2) short datatype

- it used to store the numbers between -32768 to 32767

syntax:

datatype variablename=value;

e.g

short abc=4000;

System.out.println(abc);

short datatype

abc variablename

1000 is value for abc variable

package Tutorial2;

public class Demo3 {

public static void main(String[] args) {

// TODO Auto-generated method stub

short pqr = -500;

System.out.println(pqr);

}

}

package Tutorial2;

public class Demo4 {

public static void main(String[] args) {

byte a = 10;

System.out.println(a);

short b = 3000;

System.out.println(b);

}

}

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

3) int datatype

- it is used to store the numbers between -2,147,483,648 to 2,147,483,647 values

syntax:

datatype variablename=values;

e.g

int abc =100000;

System.out.println(abc);

package Tutorial2;

public class Demo5 {

public static void main(String[] args) {

int abc=555000;

System.out.println(abc);

}

}

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

4) long datatype

it used to store the values between -9,223,372,036,854,775,808 to 9,223,372,038,775,807 values

-if we have large values then at end of values we use upper case L or lower case l.

syntax:

datatype variablename=valuesl;

e.g

long a=190;

long abc = 909000000000L;

long xyz = 909000000000l;

System.out.println(abc);

package Tutorial2;

public class Demo6 {

public static void main(String[] args) {

long abc = 190000000000L;

System.out.println(abc);

}

}

package Tutorial2;

public class Demo7 {

public static void main(String[] args) {

byte a=100;

short b=2000;

int c=1900000;

long d=100000000000L;

System.out.println(a);

System.out.println(b);

System.out.println(c);

System.out.println(d);

}

}

==========================================================

If want to store decial values then we use

1) float datatype

2) double datatype

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

1) float datatype

- it used to store the decimal values between 6 to 7 digits

- at the end of values we have use upper case F or lower case f.

syntax:

datatype variablename=valuef;

e,g

float abc = 190.55f;

System.out.println(abc);

package Tutorial2;

public class Demo8 {

public static void main(String[] args) {

float abc = 155.55666F;

System.out.println(abc);

}

}

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

2) double datatype

- it used to store the decimal values upto 15 digits

syntax:

datatype variablename=value;

e.g

double abc = 8999.717374744;

System.out.println(abc);

package Tutorial2;

public class Demo9 {

public static void main(String[] args) {

double abc = 989188343.3555234335;

System.out.println(abc);

}

}

=============================================================

If u want to store boolean values

1) boolean datatype

- it used to store the boolean values (true or false)

syntax

datatype variablename=value;

e.g.

boolean abc = false;

System.out.println(abc);

package Tutorial2;

public class Demo10 {

public static void main(String[] args) {

boolean abc = true;

System.out.println(abc);

}

}

===========================================

if u want to store single character then we use char datatype

1) char datatype

- it used to store the single character in single quote.

- Always we have to use single quote for values.

syntax:

datatype variablename='value';

e.

char abc = 'p';

System.out.println(abc);

package Tutorial2;

public class Demo11 {

public static void main(String[] args) {

char abc = 'a';

System.out.println(abc);

}

}

======================================================================

Primitive datatype

store the numbers

1)byte

datatype variablename=value;

2) short

datatype variablename=value;

3) int

datatype variablename=value;

4) long

datatype variablename=valueL;

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

store the decimal values

5) float

datatype variablename=valueF;

6) double

datatype variablename=value;

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

store the boolean values

7) boolean

datatype variablename=value;//true or false

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

store the single character

8) char

datatype variablename='value';

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

package Tutorial2;

public class Demo12 {

public static void main(String[] args) {

byte a=100;

short b=2000;

int c =90000;

long d=1900999999999990l;

float e=82983.35f;

double f=23484.34432;

char g = '4';

boolean h= false;

System.out.println(a);

System.out.println(b);

System.out.println(c);

System.out.println(d);

System.out.println(e);

System.out.println(f);

System.out.println(g);

System.out.println(h);

}

}

Datatypes:

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

there are 2 types of datatypes

1) Primitive Datatype

2) Non Primitive Datatype

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

1) Primitive Datatype

there are 8 types of primitive datatypes

1) byte

2) short

3) int

4) long

5) float

6) double

7) char

8) boolean

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

syntax:

datatype variablename=value;

Variable Name declaration Rules

abc Valid

ABC valid

\_abc valid

\_ABC valid

$abc valid

$ABC valid

abcXYZ valid

abc1828 valid

2122abc invalid ----> syntax Errors

@abc invalid ----> syntax Errors

package Tutorial3;

public class Demo1 {

public static void main(String[] args) {

int abc=10; //Valid variable name

int ABC =20; //Valid variable name

int \_abc=30; //Valid variable name

int \_ABC=40; //Valid variable name

int $abc=300; //Valid variable name

int $ABC=3043; //Valid variable name

int abc134=2340; //Valid variable name

int ABC122=2340; //Valid variable name

int 123=3455; //Invalid variable name

int 134abc=344; //Invalid variable name

int #abc=234; //Invalid variable name

int abc#$$=345; //Invalid variable name

int abc xyz=34330; //Invalid variable name

}

}

===========================================================

Can we declare multiple variable in same line with same datatypes

yes

Datatype variablename1=value1, variablename2=value2, variableName3=value3;

int a=100,b=200,c=300;

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

Can we declare multiple variable in same line with different datatypes

No,

int a=200,b=23.34f,c='A'; //syntax Error

package Tutorial3;

public class Demo2 {

public static void main(String[] args) {

int a=10,b=20,c=30;

System.out.println(c);

System.out.println(a);

System.out.println(b);

}

}

package Tutorial3;

public class Demo3 {

public static void main(String[] args) {

char a='P',b='S',c='R',d='A',e='S';

System.out.println(a);

System.out.println(b);

System.out.println(c);

System.out.println(d);

System.out.println(e);

}

}

==================================

Operator in java

there are 4 types operator in java

1) Arithemetic operator

2 Relational Operator

3) logical Operator

4) Assignment Operator

1) Arithemetic operator

there are 7 types of Arithemetic operator

i) addition +

ii) substraction -

iii) Multiplication \*

iv) Division /

v) module %

vi) increment ++

viii) decrement --

e.g.

int a=20,b=10;

System.out.println(a+b);

System.out.println(a-b);

System.out.println(a\*b);

System.out.println(a/b);

System.out.println(a%b);

a++;//a = a+1;

System.out.println(a);

b--;//b=b-1

System.out.println(b);

package Tutorial3;

public class Demo4 {

public static void main(String[] args) {

int a=20,b=10;

System.out.println(a+b);//30

System.out.println(a-b);//10

System.out.println(a\*b);//200

System.out.println(a/b);//2

System.out.println(a%b);//0

a++;

System.out.println(a);//21

b--;

System.out.println(b);//9

}

}

package Tutorial3;

public class Demo5 {

public static void main(String[] args) {

int a=5;

float b=10.0f;

System.out.println(a+b);//15.0

System.out.println(a-b);//-5.0

System.out.println(a\*b);//50.0

System.out.println(a/b);//0.5

System.out.println(a%b);//5

a++;

System.out.println(a);//6

b--;

System.out.println(b);//9.0

}

}

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

package Tutorial3;

public class Demo6 {

public static void main(String[] args) {

int a=10,b=20;

System.out.println(a+b);//30

System.out.println(a-b);//-10

System.out.println(a\*b);//200

System.out.println(a/b);//0

System.out.println(a%b);//10

a++;

System.out.println(a);//1

b--;

System.out.println(b);//19

}

}

====================================================

2) Relational Operator

there are 6 types in Relational Operator

i) equal to ==

ii) not equal to !=

iii) greater than >

iv) greater than and equal to >=

v) less than <

vi) less than and equal to <=

Result will be show in boolean (true/false)

int a=100;

int b=200;

System.out.println(a==b); //false

System.out.println(a!=b); //true

System.out.println(a>b); .//false

System.out.println(a>=b); //false

System.out.println(a<b); //true

System.out.println(a<=b); //true

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

int a=100;

int b=100;

System.out.println(a==b); //T

System.out.println(a!=b); //F

System.out.println(a>b); //F

System.out.println(a>=b); //TRUE

System.out.println(a<b); //false

System.out.println(a<=b); //TRUE

package Tutorial3;

public class Demo7 {

public static void main(String[] args) {

int a = 100, b = 200;

System.out.println(a == b);// False

System.out.println(a != b);// True

System.out.println(a > b);// False

System.out.println(a >= b);// False

System.out.println(a < b);// true

System.out.println(a <= b);// true

}

}

package Tutorial3;

public class Demo8 {

public static void main(String[] args) {

int a = 100;

int b = 100;

System.out.println(a == b);// true

System.out.println(a != b);// false

System.out.println(a>b);//false

System.out.println(a>=b);//true

System.out.println(a<b);//false

System.out.println(a<=b);//True

}

}

package Tutorial3;

public class Demo9 {

public static void main(String[] args) {

int a=100;

float b=100.00f;

System.out.println(a==b);//true

System.out.println(a!=b);//false

System.out.println(a>b);//false

System.out.println(a>=b);//true

System.out.println(a<b);//false

System.out.println(a<=b);//true

}

}

=======================================================

3) Assignment operator

i) assignment = we can assign the values for variable

ii) add and Assign +=

iii) sub and Assign -=

iv) div and assign /=

v) mul and assign \*=

vi) module and assign %=

int a=100;

System.out.println(a); //100

a=2000; We are re assign the value for variable a.

System.out.println(a); //2000

a=-10;

System.out.println(a); //-10

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

ii) add and Assign +=

int a=5; //12

a+=7; //a=a+7= 5+7=12

a+=3; //a=a+3=12+3=15

a+=9; //a=a+9 = 15+9= 24

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

iii) sub and assign

int a=20;

a-=5; //a=a-5=20-5=15

a-=7; //a=a-7=15-7=8

package Tutorial3;

public class Demo10 {

public static void main(String[] args) {

int a = 100;

System.out.println(a);

a = 299;

System.out.println(a);

a=-145;

System.out.println(a);

}

}

package Tutorial3;

public class Demo11 {

public static void main(String[] args) {

int a = 10;

System.out.println(a);

a+=100;

System.out.println(a);//110

a+=20;

System.out.println(a);//130

a-=115; //a=a-115= 130-115= 15

System.out.println(a);//15

a-=3;

System.out.println(a);//12

a\*=10;

System.out.println(a);//120

a/=10;

System.out.println(a);//12

a%=6;

System.out.println(a);//0

}

}

package Tutorial3;

public class Demo12 {

public static void main(String[] args) {

int a = 15;

System.out.println(a);

a \*= 10; // a=a\*10=15\*10=150

System.out.println(a);// 150

a += 50;// a=a+50=150+50=200

System.out.println(a);// 200

a-=140;//a=a-140=200-140=60

System.out.println(a);//60

a/=20;//a=a/20=60/20=3

System.out.println(a);//3

a%=10;//

System.out.println(a);//3

}

}

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

4) Logical operator

there are 3 types Logical operator

1) and operator &&

2) or operator ||

3) not operator !

1) and operator &&

if both condition are true then and then only result is true.

condition1 condition2 and Operator result

true true true

true false false

false true false

false false false

System.out.println(true&&true);

System.out.println(true&&false);//false

System.out.println(false&&true);//false

System.out.println(false&&false);//false

package Tutorial3;

public class Demo13 {

public static void main(String[] args) {

System.out.println(true && true);// true

System.out.println(true && false);// false

System.out.println(false && true);// false

System.out.println(false && false);// false

}

}

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

2) Or operator ||

if anyone condition is true then result is true

condition1 condition2 or operator result

true true true

true false true

false true true

false false false

System.out.println(true || true); true

System.out.println(true || false); true

System.out.println(false || true); true

System.out.println(false || false); false

package Tutorial3;

public class Demo14 {

public static void main(String[] args) {

System.out.println(true || true);// true

System.out.println(true || false);// true

System.out.println(false || true);// true

System.out.println(false || false);// false

}

}

====================================

Not operator

if condition is true, not operator mark the result is false

if condition is false, not operator mark the result is true

package Tutorial3;

public class Demo15 {

public static void main(String[] args) {

System.out.println(!true);

System.out.println(!false);

}

}

package Tutorial3;

public class Demo16 {

public static void main(String[] args) {

int a=40,b=30,c=20;

System.out.println((a>b)&&(a>c));//true

System.out.println((a>b)||(a>c));//true

System.out.println(!((a>b)&&(a>c)));//false

}

}

==============================================

package Tutorial3;

public class Demo16 {

public static void main(String[] args) {

int a = 10, b = 30, c = 10;

System.out.println((a > b) && (a == c));// false

System.out.println((a > b) || (a == c));// true

}

}

=========================================================

Datatypes:

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

there are 2 types of datatypes

1) Primitive Datatype

2) Non Primitive Datatype

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

1) Primitive Datatype

there are 8 types of primitive datatypes

1) byte

2) short

3) int

4) long

5) float

6) double

7) char

8) boolean

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

syntax:

datatype variablename=value;

Variable Name declaration Rules

abc Valid

ABC valid

\_abc valid

\_ABC valid

$abc valid

$ABC valid

abcXYZ valid

abc1828 valid

2122abc invalid ----> syntax Errors

@abc invalid ----> syntax Errors

package Tutorial3;

public class Demo1 {

public static void main(String[] args) {

int abc=10; //Valid variable name

int ABC =20; //Valid variable name

int \_abc=30; //Valid variable name

int \_ABC=40; //Valid variable name

int $abc=300; //Valid variable name

int $ABC=3043; //Valid variable name

int abc134=2340; //Valid variable name

int ABC122=2340; //Valid variable name

int 123=3455; //Invalid variable name

int 134abc=344; //Invalid variable name

int #abc=234; //Invalid variable name

int abc#$$=345; //Invalid variable name

int abc xyz=34330; //Invalid variable name

}

}

===========================================================

Can we declare multiple variable in same line with same datatypes

yes

Datatype variablename1=value1, variablename2=value2, variableName3=value3;

int a=100,b=200,c=300;

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

Can we declare multiple variable in same line with different datatypes

No,

int a=200,b=23.34f,c='A'; //syntax Error

package Tutorial3;

public class Demo2 {

public static void main(String[] args) {

int a=10,b=20,c=30;

System.out.println(c);

System.out.println(a);

System.out.println(b);

}

}

package Tutorial3;

public class Demo3 {

public static void main(String[] args) {

char a='P',b='S',c='R',d='A',e='S';

System.out.println(a);

System.out.println(b);

System.out.println(c);

System.out.println(d);

System.out.println(e);

}

}

==================================

Operator in java

there are 4 types operator in java

1) Arithemetic operator

2 Relational Operator

3) logical Operator

4) Assignment Operator

1) Arithemetic operator

there are 7 types of Arithemetic operator

i) addition +

ii) substraction -

iii) Multiplication \*

iv) Division /

v) module %

vi) increment ++

viii) decrement --

e.g.

int a=20,b=10;

System.out.println(a+b);

System.out.println(a-b);

System.out.println(a\*b);

System.out.println(a/b);

System.out.println(a%b);

a++;//a = a+1;

System.out.println(a);

b--;//b=b-1

System.out.println(b);

package Tutorial3;

public class Demo4 {

public static void main(String[] args) {

int a=20,b=10;

System.out.println(a+b);//30

System.out.println(a-b);//10

System.out.println(a\*b);//200

System.out.println(a/b);//2

System.out.println(a%b);//0

a++;

System.out.println(a);//21

b--;

System.out.println(b);//9

}

}

package Tutorial3;

public class Demo5 {

public static void main(String[] args) {

int a=5;

float b=10.0f;

System.out.println(a+b);//15.0

System.out.println(a-b);//-5.0

System.out.println(a\*b);//50.0

System.out.println(a/b);//0.5

System.out.println(a%b);//5

a++;

System.out.println(a);//6

b--;

System.out.println(b);//9.0

}

}

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

package Tutorial3;

public class Demo6 {

public static void main(String[] args) {

int a=10,b=20;

System.out.println(a+b);//30

System.out.println(a-b);//-10

System.out.println(a\*b);//200

System.out.println(a/b);//0

System.out.println(a%b);//10

a++;

System.out.println(a);//1

b--;

System.out.println(b);//19

}

}

====================================================

2) Relational Operator

there are 6 types in Relational Operator

i) equal to ==

ii) not equal to !=

iii) greater than >

iv) greater than and equal to >=

v) less than <

vi) less than and equal to <=

Result will be show in boolean (true/false)

int a=100;

int b=200;

System.out.println(a==b); //false

System.out.println(a!=b); //true

System.out.println(a>b); .//false

System.out.println(a>=b); //false

System.out.println(a<b); //true

System.out.println(a<=b); //true

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

int a=100;

int b=100;

System.out.println(a==b); //T

System.out.println(a!=b); //F

System.out.println(a>b); //F

System.out.println(a>=b); //TRUE

System.out.println(a<b); //false

System.out.println(a<=b); //TRUE

package Tutorial3;

public class Demo7 {

public static void main(String[] args) {

int a = 100, b = 200;

System.out.println(a == b);// False

System.out.println(a != b);// True

System.out.println(a > b);// False

System.out.println(a >= b);// False

System.out.println(a < b);// true

System.out.println(a <= b);// true

}

}

package Tutorial3;

public class Demo8 {

public static void main(String[] args) {

int a = 100;

int b = 100;

System.out.println(a == b);// true

System.out.println(a != b);// false

System.out.println(a>b);//false

System.out.println(a>=b);//true

System.out.println(a<b);//false

System.out.println(a<=b);//True

}

}

package Tutorial3;

public class Demo9 {

public static void main(String[] args) {

int a=100;

float b=100.00f;

System.out.println(a==b);//true

System.out.println(a!=b);//false

System.out.println(a>b);//false

System.out.println(a>=b);//true

System.out.println(a<b);//false

System.out.println(a<=b);//true

}

}

=======================================================

3) Assignment operator

i) assignment = we can assign the values for variable

ii) add and Assign +=

iii) sub and Assign -=

iv) div and assign /=

v) mul and assign \*=

vi) module and assign %=

int a=100;

System.out.println(a); //100

a=2000; We are re assign the value for variable a.

System.out.println(a); //2000

a=-10;

System.out.println(a); //-10

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

ii) add and Assign +=

int a=5; //12

a+=7; //a=a+7= 5+7=12

a+=3; //a=a+3=12+3=15

a+=9; //a=a+9 = 15+9= 24

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

iii) sub and assign

int a=20;

a-=5; //a=a-5=20-5=15

a-=7; //a=a-7=15-7=8

package Tutorial3;

public class Demo10 {

public static void main(String[] args) {

int a = 100;

System.out.println(a);

a = 299;

System.out.println(a);

a=-145;

System.out.println(a);

}

}

package Tutorial3;

public class Demo11 {

public static void main(String[] args) {

int a = 10;

System.out.println(a);

a+=100;

System.out.println(a);//110

a+=20;

System.out.println(a);//130

a-=115; //a=a-115= 130-115= 15

System.out.println(a);//15

a-=3;

System.out.println(a);//12

a\*=10;

System.out.println(a);//120

a/=10;

System.out.println(a);//12

a%=6;

System.out.println(a);//0

}

}

package Tutorial3;

public class Demo12 {

public static void main(String[] args) {

int a = 15;

System.out.println(a);

a \*= 10; // a=a\*10=15\*10=150

System.out.println(a);// 150

a += 50;// a=a+50=150+50=200

System.out.println(a);// 200

a-=140;//a=a-140=200-140=60

System.out.println(a);//60

a/=20;//a=a/20=60/20=3

System.out.println(a);//3

a%=10;//

System.out.println(a);//3

}

}

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

4) Logical operator

there are 3 types Logical operator

1) and operator &&

2) or operator ||

3) not operator !

1) and operator &&

if both condition are true then and then only result is true.

condition1 condition2 and Operator result

true true true

true false false

false true false

false false false

System.out.println(true&&true);

System.out.println(true&&false);//false

System.out.println(false&&true);//false

System.out.println(false&&false);//false

package Tutorial3;

public class Demo13 {

public static void main(String[] args) {

System.out.println(true && true);// true

System.out.println(true && false);// false

System.out.println(false && true);// false

System.out.println(false && false);// false

}

}

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

2) Or operator ||

if anyone condition is true then result is true

condition1 condition2 or operator result

true true true

true false true

false true true

false false false

System.out.println(true || true); true

System.out.println(true || false); true

System.out.println(false || true); true

System.out.println(false || false); false

package Tutorial3;

public class Demo14 {

public static void main(String[] args) {

System.out.println(true || true);// true

System.out.println(true || false);// true

System.out.println(false || true);// true

System.out.println(false || false);// false

}

}

====================================

Not operator

if condition is true, not operator mark the result is false

if condition is false, not operator mark the result is true

package Tutorial3;

public class Demo15 {

public static void main(String[] args) {

System.out.println(!true);

System.out.println(!false);

}

}

package Tutorial3;

public class Demo16 {

public static void main(String[] args) {

int a=40,b=30,c=20;

System.out.println((a>b)&&(a>c));//true

System.out.println((a>b)||(a>c));//true

System.out.println(!((a>b)&&(a>c)));//false

}

}

==============================================

package Tutorial3;

public class Demo16 {

public static void main(String[] args) {

int a = 10, b = 30, c = 10;

System.out.println((a > b) && (a == c));// false

System.out.println((a > b) || (a == c));// true

}

}

=========================================================

Java Conditional Statement

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

- there are 2 types of conditional statement in java

1) if else condition

2) switch statement

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

1) if else condition

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

i) if condition or single condition;

syntax:

if(condition)

{

//java statement

}

if(10>50)

{

System.out.println("Automation Testing class");

}

Note:

i) if block condition is true then and then only it will execute if block of code.

ii) if block condtion is false then it will skip the if block of code means it will not execute if block of code.

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

e.g

public class Demo1

{

public static void main(String [] args)

{

System.out.println("Main method is started");

if(100>800)

{

System.out.println("Automation Testing class");

}

System.out.println("Main method is Ended");

}

}

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

package Tutorial4;

public class Demo1 {

public static void main(String[] args) {

System.out.println("Main Method is started");

if(100>80)

{

System.out.println("Automation testing class");

}

System.out.println("Main Method is ended");

}

}

===============================================================================

ii) if else condition

syntax:

if(condition)

{

//java statememt

}

else

{

//java statement

}

note:

i) if block condition is true then it will execute if block of code and it ignore the else block of code.

ii) if block conditon is false then it will skip/ignore the if block of code and it will execute the else block of code.

e.g

if(10>70)

{

System.out.println("Pune");

}

else

{

System.out.println("Mumbai");

}

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

e.g

int a=1000;

int b=150;

if(a>b)

{

System.out.println("A is bigger than B");

}

else

{

System.out.println("B is bigger than A");

}

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

package Tutorial4;

public class Demo2 {

public static void main(String[] args) {

int a = 100, b = 100;

if (a > b) {

System.out.println("A is bigger than B");

} else {

System.out.println("B is bigger than A");

}

}

}

=====================================================

iii) else if condition

syntax:

if(condition1)

{

//java statement

}

else if(condition2)

{

//java statement

}

else

{

//java statement

}

Note;

i) if codition1 is true then it will execute the if block of code and it will ignore remaining block of code.

ii) if condition1 is false then it wil skip the if block of code and it will check condition2 and if condition 2 is true then it will execute the

else if block of code and it will ignore the else block of code.

iii) if condition1 is false then it will check condition2 and if condition2 is also false then it will execute else block of code.

e.g

int a=100,b=800,c=1000;

if(a>b)

{

System.out.println("Pune");

}

else if(b>c)

{

System.out.println("Mumbai");

}

else

{

System.out.println("Delhi");

}

package Tutorial4;

public class Demo3 {

public static void main(String[] args) {

int a=100,b=800,c=1000;

if(a>b)

{

System.out.println("Pune");

}

else if(b>c)

{

System.out.println("Mumbai");

}

else

{

System.out.println("Delhi");

}

}

}

package Tutorial4;

public class Demo4 {

public static void main(String[] args) {

int a=100,b=800,c=8000;

if((a>b)&&(a>c))

{

System.out.println("A is bigger than B and C");

}

else if(b>c)

{

System.out.println("B is bigger than A and C");

}

else

{

System.out.println("C is bigger than A and B");

}

}

}

=============================================

\package Tutorial4;

public class Demo5 {

public static void main(String[] args) {

// find max values out of 4 numbers

int a = 100000, b = 10030, c = 24000400, d = 34505;

if((a>b)&&(a>c)&&(a>d))

{

System.out.println("A is bigger than B,C,D");

}

else if((b>c)&&(b>d))

{

System.out.println("B is bigger than A,C,D");

}

else if(c>d)

{

System.out.println("C is bigger than A,B,D");

}

else

{

System.out.println("D is bigger than A,B,C");

}

}

}

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

Ass:

Declare the age variable

int age=values

conditions

if age is between 0 to 11 then print child keyword

if age is between 12 to 17 then print teen keyword

if age is between 18 to 64 print adult keyword

if age is above 64 the print senior keyword

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

Ass:

declare the int variable

int a=1;

conditions

i) if a number is between 1 to 100 then print a small number

ii) if a number is between 101 to 1000 then print a number is medium

iii) if a numbers is between 1001 to 10000 the print a number is large

iv) if a number is above 10001 then a number is biggest number

v) if a number is below 1 then a number is zero or negative number.

if((a>=1)&&(a<=100))

{

syso("A is small Number")

}

else if((a>=101)&&(a<=1000))

{

syso("A is Medium Number")

}

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

declare the marks int variable

int marks=values;

conditions

i) if marks is below 25 then display the grade as F

ii) if marks is 25 to 45 then display the grade as E

iii) if marks is 46 to 55 then display the grade as D

iv) if marks is 56 to 65 then display the grade as C

v) if marks is 66 to 75 then display the grade as B

vi) if marks is above 75 then display the grade as A

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

package Tutorial4;

public class Demo6 {

/\*

\* declare the marks int variable int marks=values;

\*

\* conditions

\* i) if marks is below 25 then display the grade as F

\* ii) if marks is 25 to 45 then display the grade as E

\* iii) if marks is 46 to 55 then display the grade as D

\* iv) if marks is 56 to 65 then display the grade as C

\* v) if marks is 66 to 75 then display the grade as B

\* vi) if marks is above 75 then display the grade as A

\*/

public static void main(String[] args) {

int marks=75;

if(marks<25)

{

System.out.println("Grade as F");

}

else if((marks>=25)&&(marks<=45))

{

System.out.println("Grade as E");

}

else if((marks>=46)&&(marks<=55))

{

System.out.println("Grade as D");

}

else if((marks>=56)&&(marks<=65))

{

System.out.println("Grade as C");

}

else if((marks>=66)&&(marks<=75))

{

System.out.println("Grade as B");

}

else

{

System.out.println("Grade as A");

}

}

}

========================================================================

Loop statement in java

there are 4 types of loops in java

1) for loop

2) while loop

3) do while loop

4) enhance for loop

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

1) for loop

if u want to repeat a block of statement for specific amount of time then we use loop.

syntax:

for(datatype variablename=startValue;condition/endvalue;increment/decrement)

{

//java statement

}

e.g

print 1 to 5 values using for loop

for(int i=1;i<=5;i++)

{

System.out.println(i);//1 2 3 4 5

}

package Tutorial4;

public class Demo7 {

public static void main(String[] args) {

//print 1 to 5 value using for loop.

for(int i=1;i<=5;i++)

{

System.out.println(i);//1 2 3 4 5

}

}

}

package Tutorial4;

public class Demo8 {

public static void main(String[] args) {

// print 10 to 20 values

for(int i=10;i<=20;i++)

{

System.out.println(i);

}

}

}

package Tutorial4;

public class Demo9 {

public static void main(String[] args) {

// print 5 to 1 values

for(int i=5;i>=1;i--)

{

System.out.println(i);//5 4 3 2 1

}

}

}

package Tutorial4;

public class Demo10 {

public static void main(String[] args) {

// print 40 to 60 values

for(int i=40;i<=60;i++)

{

System.out.println(i);

}

}

}

package Tutorial4;

public class Demo10 {

public static void main(String[] args) {

// print 10 to 15 values except 13 value

for(int i=10;i<=15;i++)

{

//14!=13

if(i!=13)

{

System.out.println(i);//10 11 12 14 15

}

}

}

}

package Tutorial4;

public class Demo11 {

public static void main(String[] args) {

// print 50 to 40 values except 45 numbers

for(int i=50;i>=40;i--)

{

if(i!=45)

{

System.out.println(i);

}

}

}

}

========================================

1) print even values between 1 to 20 values

package Tutorial4;

public class Demo11 {

public static void main(String[] args) {

// print even values between 1 to 20 values

for (int i = 1; i <=20; i++) {

if (i%2==0)

{

System.out.println(i);

}

}

}

}

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

2) print odd values between 11 to 30 values

package Tutorial4;

public class Demo11 {

public static void main(String[] args) {

// print odd values between 11 to 30 values

for (int i = 11; i <= 30; i++) {

if (i % 2 == 1) {

System.out.println(i);

}

}

}

}

=================================================================

Ass:

1) print even values between 40 to 60 except 44, 48 and 56 numbers

1) print odd values between 91 to 71 except 81, 87 and 77 numbers

=================================================================

Scenario Based Questions on for loop vvvvvimmmppppp

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

Scenario1:

for(;;)

{

System.out.println("Hello");

}

Result:

it will print Hello keyword infinite time

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

Scenario2:

for(int i=1;;)

{

System.out.println(i);

}

Result:

it will print 1 infinite time

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

Scenario3:

for(int i=1;i<=10;)

{

System.out.println(i);

}

Result:

it will print 1 infinite time

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

Scenario4:

for(int i=1;;i++)

{

System.out.println(i);

}

Result:

it will print infinite numbers

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

package Tutorial5;

public class Demo1 {

public static void main(String[] args) {

////it will print Hello keyword infinite times

for(;;)

{

System.out.println("Hello");

}

}

}

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

package Tutorial5;

public class Demo1 {

public static void main(String[] args) {

//it will print 1 infinite times

for(int i=1;;)

{

System.out.println(i);

}

}

}

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

package Tutorial5;

public class Demo1 {

public static void main(String[] args) {

//it will print 1 infinite times

for(int i=1;i<=10;)

{

System.out.println(i);

}

}

}

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

package Tutorial5;

public class Demo1 {

public static void main(String[] args) {

//it will print infinite numbers

for(int i=1;;i++)

{

System.out.println(i);

}

}

}

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

package Tutorial5;

public class Demo1 {

public static void main(String[] args) {

//it will print infinite odd numbers

for(int i=1;;i=i+2)

{

System.out.println(i);

}

}

}

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

package Tutorial5;

public class Demo1 {

public static void main(String[] args) {

//it will print infinite even numbers

for(int i=0;;i=i+2)

{

System.out.println(i);

}

}

}

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

2) while loop

it repeats a block of statement while condition is true.

syntax:

declare the variables/start value;

while(condition/endvalues)

{

//java statement

increment/decrement

}

e.g.

print 1 to 5 values using while loop

int i=1;

while(i<=5)

{

System.out.println(i); //1 2 3 4 5

i++;

}

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

package Tutorial5;

public class Demo1 {

public static void main(String[] args) {

/// print 1 to 5 values using while loop

int i = 1;

// 6<=5

while (i <= 5) {

System.out.println(i);// 1 2 3 4 5

i++;// 5+1=6

}

}

}

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

package Tutorial5;

public class Demo2 {

public static void main(String[] args) {

// print 40 to 50 values using while loop

int a=40;

while(a<=50)

{

System.out.println(a);

a++;

}

}

}

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

package Tutorial5;

public class Demo3 {

public static void main(String[] args) {

// print 90 to 80 values using while loop

int i=90;

while(i>=80)

{

System.out.println(i);

i--;

}

}

}

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

package Tutorial5;

public class Demo4 {

public static void main(String[] args) {

// print 40 to 50 even values using while loop

int i = 40;

while (i <= 50)

{

if (i % 2 == 0)

{

System.out.println(i);//40

}

i++;

}

}

}

========================================================

package Tutorial5;

public class Demo5 {

public static void main(String[] args) {

// print 40 to 60 odd values using while loop and

//except the 45, 53 values

int i=40;

while(i<=60)

{

if((i%2==1)&&(i!=45)&&(i!=53))

{

System.out.println(i);

}

i++;

}

}

}

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

package Tutorial5;

public class Demo6 {

public static void main(String[] args) {

// it will print hello keyword infinite times

while (true) {

System.out.println("hello");

}

}

}

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

package Tutorial5;

public class Demo6 {

public static void main(String[] args) {

int i=1;

// it will print 1 infinite times

while (true)

{

System.out.println(i);

}

}

}

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

package Tutorial5;

public class Demo6 {

public static void main(String[] args) {

int i=1;

// it will print infinite numbers

while (true)

{

System.out.println(i);

i++;

}

}

}

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

package Tutorial5;

public class Demo6 {

public static void main(String[] args) {

int i=1;

// it will print 1 infinite times

while (i<=10)

{

System.out.println(i);

}

}

}

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

3) do while loop

- it repeats the block of while condition is true but it still it will execute block at least one time if condition is false also.

syntax:

declare the variables/start values;

do

{

//java statement

increment/decrement;

}

while(condition/endvalues);

e.g.

print 1 to 5 values using do while loop

int i=1;

do

{

System.out.println(i);//1 2 3 4 5

i++;//6

}

while(i<=5);

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

why we don't use do while loop?

because it run will do block of code without checking the conditions in first iteration.

after second iteration it will check conditions.

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

package Tutorial5;

public class Demo7 {

public static void main(String[] args) {

// print 1 to 5 values using do while loop

int i=1;

do {

System.out.println(i);//1 2 3 4 5

i++;//6

}

while(i<=5);

}

}

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

package Tutorial5;

public class Demo8 {

public static void main(String[] args) {

// print 21 to 11 values using do while loop

int i=21;

do

{

System.out.println(i);

i--;

}

while(i>=11);

}

}

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

package Tutorial5;

public class Demo9 {

public static void main(String[] args) {

// // print 40 to 50 even values using do while loop

int i=40;

do

{

if(i%2==0)

{

System.out.println(i);

}

i++;

}

while(i<=50);

}

}

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

package Tutorial5;

public class Demo10 {

public static void main(String[] args) {

// print 80 to 90 odd value except the 85

int i = 80;

do {

if ((i % 2 == 1) && (i != 85)) {

System.out.println(i);

}

i++;

} while (i <= 90);

}

}

=========================================================

Non Primitive Datatypes

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

1) String

2) Arrays

3) class

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

2) Arrays

- Array is an object in java.

- in array we store single/same datatype values.

- at the time of creating array we mention array length.

- once we declare the length of array we can't modify it means array length is fixed.

- array index it starts from zero.

syntax:

datatype [] arrayVariableName= new datatype[length];

or

datatype arrayVariableName [] = new datatype[length];

datatype: byte, short, int, long, float, double, char, boolean, string

arrayVariableName any name

new it is predefine keyword in java which is used to create object.

length integer length

e.g.

int [] abc = new int[5];

float [] a1 =new float[10];

String [] a2 =new String[7];

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

how to create array?

datatype [] arrayVariableName= new datatype[length];

how to assign the values for array?

arrayVariableName[indexPosition]=value;

how to capture the array values

datatype variableName = arrayVariableName[indexPosition];

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

declare the int type of array

int [] abc = new int[5];

abc[0]=10;

abc[1]=20;

abc[2]=30;

abc[3]=40;

abc[4]=50;

System.out.println(abc[2]);

System.out.println(abc.length);

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

length property VVVVIMMMPPPP

it is used to capture the length of array.

return type is int.

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

package Tutorial6;

import java.util.Arrays;

public class Demo1 {

public static void main(String[] args) {

int[] abc = new int[5];

abc[0] = 10;

abc[1] = 20;

abc[2] = 30;

abc[3] = 40;

abc[4] = 50;

System.out.println(abc[0]);

System.out.println(abc[1]);

System.out.println(abc[2]);

System.out.println(abc[3]);

System.out.println(abc[4]);

}

}

package Tutorial6;

import java.util.Arrays;

public class Demo1 {

public static void main(String[] args) {

int[] abc = new int[5];

abc[0] = 10;

abc[1] = 20;

abc[2] = 30;

abc[3] = 40;

abc[4] = 50;

for(int i=0;i<=4;i++)

{

System.out.println(abc[i]);//10 20 30 40 50

}

}

}

============================================================

Find max values in array? VVVVVVVIMMMPPPPPPPPP

int [] abc = new int [5];

abc[0]=100;

abc[1]=200;

abc[2]=300;

abc [3]=500;

abc[4]=400;

step 1: assume first index position value is max value.

int max = abc[0];//500

step 2: iterate all the array values using for loop.

for(int i=0;i<=4;i++)

{

//400>500

if(abc[i]>max)

{

max=abc[i];//500

}

}

System.out.println(max);

====================================================

package Tutorial6;

public class Demo2 {

public static void main(String[] args) {

int[] abc = new int[5];

abc[0] = 1000;

abc[1] = 1200;

abc[2] = 800;

abc[3] = 1400;

abc[4] = 500;

int max= abc[0];//1200

for(int i=0;i<=4;i++)

{

//500 > 1400

if(abc[i]>max)

{

max=abc[i];//1400

}

}

System.out.println(max);

}

}

================================================================

find minimum values in array? VVVVIMMMPPPPP

int [] abc = new int [5];

abc[0]=100;

abc[1]=200;

abc[2]=30;

abc [3]=5;

abc[4]=40;

step 1: assume first index position values as minimum values

int min = abc[0];//30

step 2: iterate the values from array using for loop

for(int i=0;i<=4;i++)

{

//40 <5

if(abc[i]<min)

{

min=abc[i];//5

}

}

System.out.println(min);

===============================================

package Tutorial6;

public class Demo3 {

public static void main(String[] args) {

int [] abc = new int[5];

abc[0]=100;

abc[1]=20;

abc[2]=23;

abc[3]=244;

abc[4]=34;

int min = abc[0];//20

for(int i=0;i<=4;i++)

{

//34<20

if(abc[i]<min)

{

min=abc[i];//20

}

}

System.out.println(min);

}

}

==================================================

package Tutorial6;

public class Demo4 {

public static void main(String[] args) {

String [] abc = new String[5];

abc[0]="selenium";

abc[1]="RestAssured";

abc[2]="Page Object Model (POM) Design pattern";

abc[3]="Cucumber-BDD Framework";

abc[4]="Collection Framework";

System.out.println(abc[0]);

System.out.println(abc[1]);

System.out.println(abc[2]);

System.out.println(abc[3]);

System.out.println(abc[4]);

}

}

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

package Tutorial6;

public class Demo4 {

public static void main(String[] args) {

String[] abc = new String[5];

abc[0] = "selenium";

abc[1] = "Rest Assured";

abc[2] = "Page Object Model (POM) Design pattern";

abc[3] = "Cucumber-BDD Framework";

abc[4] = "Collection Framework";

for (int i = 0; i <= 4; i++) {

System.out.println(abc[i]);

}

}

}

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

enhance for loop

syntax:

for(datatype variableName : arrayName)

{

System.out.println(variableName);

}

package Tutorial6;

public class Demo4 {

public static void main(String[] args) {

String[] abc = new String[5];

abc[0] = "selenium";

abc[1] = "Rest Assured";

abc[2] = "Page Object Model (POM) Design pattern";

abc[3] = "Cucumber-BDD Framework";

abc[4] = "Collection Framework";

for(String xyz:abc)

{

System.out.println(xyz);

}

}

}

package Tutorial6;

public class Demo5 {

public static void main(String[] args) {

float abc[] = new float[5];

abc[0] = 10.345f;

abc[1] = 34.315f;

abc[2] = 26.05f;

abc[3] = 73.45f;

abc[4] = 12.5f;

for (float a : abc) {

System.out.println(a);

}

}

}

=========================================

package Tutorial6;

public class Demo6 {

public static void main(String[] args) {

//create a new array and assign the values.

int [] abc = {100,200,300,400};

System.out.println(abc.length);

for(int xyz:abc)

{

System.out.println(xyz);

}

}

}

========================================

package Tutorial6;

import java.util.Arrays;

public class Demo6 {

public static void main(String[] args) {

//create a new array and assign the values.

int [] abc = {100,200,300,400};

String xyz = Arrays.toString(abc);

System.out.println(xyz);

}

}

====================================================

package Tutorial6;

import java.util.Arrays;

public class Demo7 {

public static void main(String[] args) {

int[] abc = { 100, 20, 30, 400, 50 };

// sort() method is used to sort the value as per the ascending order

Arrays.sort(abc);

int a = abc[abc.length - 1];// max values

System.out.println(a);

System.out.println(abc[0]);

int b = abc[abc.length - 2];// max values

System.out.println(b);

}

}

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

package Tutorial6;

import java.util.Arrays;

public class Demo8 {

public static void main(String[] args) {

int[] abc = new int[5];

abc[0] = 100;

abc[1] = 20;

abc[2] = 23;

abc[3] = 244;

abc[4] = 34;

// sort arrays values as per the ascending order

Arrays.sort(abc);

int a = abc.length;// 5

// find max value

int max = abc[a - 1];

System.out.println(max);

//find 2nd max value

int secondmax = abc[a-2];

System.out.println(secondmax);

// find min values

int min = abc[0];

System.out.println(min);

}

}

==============================================================

copy all array values from one array to another array.

package Tutorial6;

public class Demo9 {

public static void main(String[] args) {

// Copy all values from abc array to xyz array.

int abc[] = { 100, 200, 400, 500 };

int xyz[] = abc;

// Copy all values from abc array to pqr array.

int[] pqr = abc.clone();

System.out.println(abc[1]);

System.out.println(xyz[1]);

System.out.println(pqr[1]);

}

}

==========================================================

package Tutorial6;

public class Demo10 {

public static void main(String[] args) {

//copy the some array values

int abc[] = { 100, 200, 400, 500,800,900 };

//400, 200

int [] xyz = {abc[2],abc[1]};

System.out.println(xyz[0]);//400

}

}

============================================================

package Tutorial6;

import java.util.Arrays;

public class Demo11 {

public static void main(String[] args) {

int abc[] = { 100, 200, 800, 900, 400, 500 };

// Arrays.toString() it used to print all arrays values in same line

// return type is String

String a = Arrays.toString(abc);

System.out.println(a);

// sort the all array values in ascending order

// return type is void

Arrays.sort(abc);

String b = Arrays.toString(abc);

System.out.println(b);

}

}

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

package Tutorial6;

public class Demo12 {

public static void main(String[] args) {

String abc[] = new String[5];

abc[0] = "java";

abc[1] = "selenium";

abc[2] = "testng";

abc[3] = "cucumber";

abc[4] = "restassured";

// reverse all the values from arrays.

//find max values in int array

//find min values in int array

}

}

=======================================================================

Create sub group and share link in main group

total student = 7 + M =8

female: 3/4

male:3/4

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

Methods in Java

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

what is methods?

- a java method is set of statement that are grouped together to perform some operations.

- method is also called as function.

- in java language or Object oriented programming languages there are 2 types of methods

1) Pre defined method

2) user defined method.

When we choose methods

- whenever we want to perform any operation multiple time then we choose methods.

- all programming concept we can use inside the methods

Adv of methods

- code reusability

- using method we can reduce the project code size.

- code maintenance in easy.

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

1) Pre defined method

there are different types of predefined method in java

1) String class method

2) Arrays class methods

3) Character class methods

4) Integer class method

5) Float class method

6) Boolean class method

7) Double class method

8) Short class method

9) Byte class methods

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

2) user defined method.

there are 2 main types in User defined method

i) static method

ii) non static method

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

ii) non static method

where we declare the non static method in java?

we declare non static method before the main method or after the main methods but inside the class.

how to non static method inside the main method?

if u want to call non static method inside the main method then we have to create object of class.

then we call non static method in main method by using object name.

how to declare the non static method?

syntax;

accessModifier nonReturnTypeKeyword methodName()

{

//statement

}

e.g.

public void abc()

{

//statement

}

what are the different types of access modifier

there are 4 types

i) public

ii) private

iii) default

iv) protected

what is non Return type keyword/

void keyword

what is method we provide?

we can any method name

where we declare non static method in java class?

- we declare method, before the main method or after the main method but inside the class.

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

public class Demo1

{

//non static method

public void abc()

{

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

}

public static void main(String [] args)

{

//we have to create object of class

Demo1 obj =new Demo1();

//call non method by using object name

obj.abc();

}

}

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

package Tutorial7;

public class Demo1 {

//Non static method

public void abc()

{

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

}

public static void main(String[] args)

{

//create object of class

Demo1 obj =new Demo1();

//call non static method by using object name

obj.abc();

}

}

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

package Tutorial7;

public class Demo2 {

//Non static method

public void abc()

{

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

}

public static void main(String[] args)

{

System.out.println("main method is started");

Demo2 obj =new Demo2();

obj.abc();

System.out.println("main method is ended");

}

}

===============================================================================

package Tutorial7;

public class Demo3 {

// Req: create non static method with reverseString method name and inside the

// method write logic for reverse String

public void reverseString()

{

String a="selenium";

int abc = a.length()-1;

for(int i=abc;i>=0;i--)

{

char p = a.charAt(i);

System.out.print(p);

}

}

public static void main(String[] args)

{

Demo3 obj =new Demo3();

obj.reverseString();

}

}

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

package Tutorial7;

public class Demo4 {

public static void main(String[] args) {

//create object of Demo3 class

Demo3 obj= new Demo3();

//call reverseString() method by using Demo3 object name

obj.reverseString();

}

}

package Tutorial7;

public class Demo6 {

//create 3 non static method

public void abc()

{

System.out.println("abc method in Demo6 class");

}

public void xyz()

{

System.out.println("xyz method in demo6 class");

}

public void pqr()

{

System.out.println("pqr method in demo6 class");

}

public static void main(String[] args)

{

System.out.println("main method is started");

//create object of Demo6 class

Demo6 obj =new Demo6();

obj.xyz();

obj.abc();

obj.pqr();

System.out.println("main method is ended");

}

}

package Tutorial7;

public class Demo7 {

//create non static method reverseString() method

public void reverseString()

{

String a="selenium";

int b = a.length()-1;

for(int i=b;i>=0;i--)

{

char c = a.charAt(i);

System.out.print(c);

}

System.out.println();

}

//create non static method as findMaxValue() method

public void findMaxValue()

{

int abc [] = {100,203,404,45,4546};

int max= abc[0];

for(int i=0;i<abc.length;i++)

{

if(abc[i]>max)

{

max=abc[i];

}

}

System.out.println(max);

}

//create non static method as findMiniValue() method

public void findMiniValue()

{

int abc [] = {12,3324,454,345,5456,43,4};

int min= abc[0];

for(int i=0;i<abc.length;i++)

{

if(abc[i]<min)

{

min=abc[i];

}

}

System.out.println(min);

}

public static void main(String[] args) {

Demo7 obj =new Demo7();

obj.reverseString();

obj.findMaxValue();

obj.findMiniValue();

}

}

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

package Tutorial7;

public class Demo8 {

public static void main(String[] args) {

//create object of Demo7 class

Demo7 obj =new Demo7();

obj.reverseString();

obj.findMiniValue();

obj.findMaxValue();

}

}

=========================================================================

ii) Non static method without return type and with multiple arguments or parameters

syntax:

AccessModifier NonReturnTypeKeyword methodName(Datatype variableName,Datatype variableName,Datatype variableName,)

{

//statement

}

there are 4 types of Access Modifier

i) public

ii) private

iii) default

iv) protected

what is Non Return Type Keyword

void keyword

What is method name we can provide?

we can give any method Name

Datatype

- byte

- short

- int

- long

- float

- double

- char

- boolean

- String

- arrays

variableName any name

where we declare the non static method

before the main method or after the main method but inside the class.

how to call non static method in main method?

- by creating object of the class.

then object name dot method name

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

public class Demo10

{

public void abc(String a)

{

System.out.println(a);

}

public static void main(String [] args)

{

Demo10 obj =new Demo10();

obj.abc("selenium");

obj.abc("selenium");

obj.abc("cucumber");

}

}

package Tutorial7;

public class Demo10 {

//Non static method without return type and 1 String arguments

public void abc(String a)

{

System.out.println(a);

}

public static void main(String[] args)

{

System.out.println("Main method is started");

Demo10 obj =new Demo10();

obj.abc("selenium");

obj.abc("selenium");

obj.abc("cucumber");

System.out.println("main method is ended");

}

}

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

package Tutorial7;

public class Demo11 {

//create non static method with 3 int arguments and write logic to find max value

public void maxValue(int a,int b, int c)

{

if((a>b)&&(a>c))

{

System.out.println("A is Max value");

}

else if(b>c)

{

System.out.println("B is max value");

}

else

{

System.out.println("C is max Value");

}

}

public static void main(String[] args) {

Demo11 obj = new Demo11();

obj.maxValue(101, 324, 333);

obj.maxValue(10, 20, -1000);

}

}

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

package Tutorial7;

public class Demo13 {

//create non static method with 1 String arguments and write logic to reverse string

public void reverseString(String abc)

{

int a = abc.length()-1;

for(int i=a;i>=0;i--)

{

char b = abc.charAt(i);

System.out.print(b);

}

System.out.println();

}

public static void main(String[] args) {

Demo13 obj =new Demo13();

obj.reverseString("selenium");

obj.reverseString("cucumber");

obj.reverseString("jenkins");

}

}

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

package Tutorial7;

public class Demo14 {

//create non static method without return and 0 arguments

public void abc()

{

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

}

//create non static method without return and 2 int arguments

public void xyz(int a,int b)

{

System.out.println(a+b);

}

public static void main(String[] args) {

Demo14 obj =new Demo14();

obj.abc();

obj.xyz(10, 1000);

}

}

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

package Tutorial19;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class Demo1 {

// create non static method without return type and 2 string arguments

public void loginFunctionality(String abc, String xyz) throws InterruptedException {

// connect browser

// up casting

WebDriver driver = new ChromeDriver();

// open a url

driver.get("https://www.saucedemo.com/");

// find username text box

WebElement wb1 = driver.findElement(By.name("user-name"));

wb1.sendKeys(abc);

// find password text box

WebElement wb2 = driver.findElement(By.id("password"));

wb2.sendKeys(xyz);

Thread.sleep(5000);

// click on login button

WebElement wb3 = driver.findElement(By.id("login-button"));

wb3.click();

}

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

Demo1 obj =new Demo1();

obj.loginFunctionality("problem\_user", "secret\_sauce");

}

}

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

package Tutorial19;

import java.io.File;

import java.io.IOException;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Demo2 {

public void abc(String screenshotName) throws IOException {

WebDriver driver = new ChromeDriver();

// open a url

driver.get("https://www.facebook.com");

TakesScreenshot ts = (TakesScreenshot) driver;

File src = ts.getScreenshotAs(OutputType.FILE);

File dest =new File(System.getProperty("user.dir")+"\\MayScreenshot\\"+screenshotName);

FileUtils.copyFile(src, dest);

}

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

Demo2 obj =new Demo2();

obj.abc("a2.png");

}

}

================================================================

Ass

create non static method without return type and 8 string arguments

up casting

open a fb register page

find first name

find lastname

find username

find password

find date drop down,

find month drop down,

find year drop down,

take screenshot

call non static method in main method

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

Ass:

create non static method without return type and 13 string arguments

find First Name:

find Last Name:

find Phone:

find Email:

find Address:

find City:

find State/Province:

find Postal Code:

find Country drop down

find and enter User Name:

find and enter Password:

find and enter Confirm Password:

click on submit button

take screenshot

call non static method in main method and at the time of calling method pass all arguments

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

Non Static Methods

i) Non static method without return type and 0 arguments

ii) Non static method without return type and multiple arguments

iii) Non static method with return type and 0 arguments

iv) non static method with return type and multiple arguments

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

iii) Non static method with return type and 0 arguments

-

syntax:

AccessModifier ReturnDatatypeName methodName()

{

///statement

return variableName/value;

}

what are the different access Modifier?

i) public ---> if we define any method as public then we can access this method anywhere in project

ii) private ---> if we define any method as private then we can access this method only inside the same class,

iii) default ---> if we define any method as default then we can access this method only within the package.

iv) protected ---> if we define any method as protected then we can access this method within the package as well as outside package

by using inheritance concept.

what is return data type name?

- byte

- short

- int

- long

- float

- double

- char

- boolean

- String

- Arrays

- Own class

- interface

we can give any method Name.

where we write Non static method with return type and 0 arguments?

we write method before the main method or after the main method but inside the class.

how to call non static method in main method?

-we have to create object of the class,

- and then by using object name we can access the method

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

// non Static method with int return type and 0 arguments

- write logic to find max value and return the max value

public class Demo1

{

public int findMaxValue()

{

int [] abc = {100,200,300,40,500,7000};

int max= abc[0];

for(int i=0;i<abc.length;i++)

{

if(abc[i]>max)

{

max=abc[i];

}

}

return max;

}

public static void main(String [] args)

{

Demo1 obj =new Demo1();

int a = obj.findMaxValue();

System.out.println(a);

}

}

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

//create Non static method with String return type and 0 args

// write logic to return reverse the String

public class Demo2

{

public String reverseString()

{

String a="madam";

String abc="";//

int b = a.length()-1;

for(int i=b;i>=0;i--)

{

char c = a.chatAt(i);//j

abc=abc+c; //

}

return abc;

}

public static void main(String [] args)

{

Demo2 obj = new Demo2();

//madam

String xyz = obj.reverseString();

if(xyz.equals("madam"))

{

System.out.println("String is Palindrome");

}

else

{

System.out.println("String is not Palindrome");

}

}

}

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

package Tutorial8;

public class Demo1 {

//create non static method with int return and 0 arguments

///find max value out from array and return the max value

public int findMaxValue()

{

int [] abc = {100,2032,404,45,54545};

int max = abc[0];

for(int i=0;i<abc.length;i++)

{

if(abc[i]>max)

{

max= abc[i];

}

}

return max;

}

public static void main(String[] args) {

System.out.println("Main method is started");

Demo1 obj = new Demo1();

int b = obj.findMaxValue();

System.out.println(b);

System.out.println("Main method is ended");

}

}

package Tutorial8;

public class Demo2 {

// create non static method with String return type

// write logic to reverse the String

public String reverseString()

{

String a="java";

String rev = "";//avaj

int b = a.length()-1;

for(int i=b;i>=0;i--)

{

char c = a.charAt(i);//j

rev = rev+c; // "ava" + 'j' ="avaj"

}

return rev;//avaj

}

public static void main(String[] args) {

//in main method compare actual string and reverse string is same or not,if

// same then print String is palindrome or if not print String is not Palindrome

Demo2 obj =new Demo2();

//avaj

String xyz = obj.reverseString();

if(xyz.equals("java"))

{

System.out.println("String is Palindrome");

}

else

{

System.out.println("String is not Palindrome");

}

}

}

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

package Tutorial8;

public class Demo2 {

// create non static method with String return type

// write logic to reverse the String

public String reverseString()

{

String a="radar";

String rev = "";//radar

int b = a.length()-1;

for(int i=b;i>=0;i--)

{

char c = a.charAt(i);//j

rev = rev+c;

}

return rev;//radar

}

public static void main(String[] args) {

//in main method compare actual string and reverse string is same or not,if

// same then print String is palindrome or if not print String is not Palindrome

Demo2 obj =new Demo2();

//radar

String xyz = obj.reverseString();

if(xyz.equals("radar"))

{

System.out.println("String is Palindrome");

}

else

{

System.out.println("String is not Palindrome");

}

}

}

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

iv) non static method with return type and multiple arguments

syntax"

AccessModifier ReturnDatatypeName methodName(datatype variableName, datatype variableName, datatype variableName)

{

//statement

return variableName/value;

}

what are the different Access modifier in java

1) public ---> if we define any method is public then we can access this method anywhere in project.

2) private ---> if we define any method as private then we can access this method only within the same class.

3) default ----> if we define any method as default then we can access this method only within the same package

4) protected ---> if we define any method as protected then we can access this method within the package as well as outside the package by using inheritance concept.

what is return datatype

byte

short

int

long

float

double

char

boolean

String

Arrays

own class

interface.

we can give any method name

where we declare the method

before the main method or after the main method but inside the class

how to call non static method in main method

- we create object of class

- then by using object name we call non static methods

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

//create non static method with String return and String arguments

//write logic to reverse the String and return the reverse String

public class Demo4

{

public String reverseString(String a)

{

String abc = "";

int b = a.length()-1;

for(int i=b; i>=0;i--)

{

char c = a.chatAt(i);

abc= abc + c;

}

return abc;

}

public static void main(String [] args)

{

Demo4 obj =new Demo4();

String xyz = obj.reverseString("java");

if(xyz.equals("java"))

{

System.out.println("String is Palindrome");

}

else

{

System.out.println("String is not Palindrome");

}

}

}

package Tutorial8;

public class Demo4 {

//create non static method with String return and String arguments

// write logic to reverse the String and return the reverse String

public String reverseString(String a)

{

String rev="";

int b = a.length()-1;

for(int i=b;i>=0;i--)

{

char c = a.charAt(i);

rev= rev+c;

}

return rev;

}

public static void main(String[] args) {

Demo4 obj =new Demo4();

String xyz = obj.reverseString("madam");

if(xyz.equals("madam"))

{

System.out.println("String is Palindrome");

}

else

{

System.out.println("String is not Palindrome");

}

}

}

package Tutorial8;

public class Demo7 {

public String checkStringIsPalindrome(String a)

{

String rev="";//radar

int b = a.length()-1;

for(int i=b;i>=0;i--)

{

char c = a.charAt(i);

rev = rev +c;

}

if(rev.equals(a))

{

return "String is Palindrome";

}

else

{

return "String is not Palindrome";

}

}

public static void main(String[] args) {

Demo7 obj =new Demo7();

String xyz = obj.checkStringIsPalindrome("radar");

System.out.println(xyz);

}

}

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

Ass:

Declare the age variable

int age=values

conditions

if age is between 0 to 11 then print child keyword

if age is between 12 to 17 then print teen keyword

if age is between 18 to 64 print adult keyword

if age is above 64 the print senior keyword

package Tutorial8;

public class Demo10 {

// Declare the age variable

// int age=values

//

//conditions

//if age is between 0 to 11 then print child keyword

//if age is between 12 to 17 then print teen keyword

//if age is between 18 to 64 print adult keyword

//if age is above 64 the print senior keyword

//create non static method with String return type and int type of arguments

public String checkAgeGroup(int age)

{

if((age>0)&&(age<=11))

{

return "child";

}

else if((age>=12)&&(age<=17))

{

return "Teen";

}

else if((age>=18)&&(age<=64))

{

return "Adult";

}

else

{

return "Senior";

}

}

public static void main(String[] args) {

Demo10 obj =new Demo10();

String xyz = obj.checkAgeGroup(1);

System.out.println(xyz);

}

}

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

Ass:

declare the int variable

int a=1;

conditions

i) if a number is between 1 to 100 then print a small number

ii) if a number is between 101 to 1000 then print a number is medium

iii) if a numbers is between 1001 to 10000 the print a number is large

iv) if a number is above 10001 then a number is biggest number

v) if a number is below 1 then a number is zero or negative number.

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

declare the marks int variable

int marks=values;

conditions

i) if marks is below 25 then display the grade as F

ii) if marks is 25 to 45 then display the grade as E

iii) if marks is 46 to 55 then display the grade as D

iv) if marks is 56 to 65 then display the grade as C

v) if marks is 66 to 75 then display the grade as B

vi) if marks is above 75 then display the grade as A

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

https://demoqa.com/buttons

//create non static method with String return type and String argument

- open application

-

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

https://prafpawar11.github.io/emptable.html

- create non static method without return type and with 1 String arguments

- if we pass tr keyword it should print all table rows

- if we pass th keyword it should print all table columns

-if we pass td keyword it should print all table data

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

https://demo.guru99.com/test/newtours/register.php

- create non static method with 1 String return type and 12 String arguments

- enter all information and click on submit button

- capture first name and last name return type same.

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

https://prafpawar11.github.io/multiplewindows.html

create a non static method with String return type and 1 String arguments

- open a application enter first name, last name and address

conditions

- if click on orange link name

- switch to it and enter credentials

take screenshot

- return url

- of click on facebook reg link

- switch to it and enter firstname, last name, username, password and DOJ

- take screenshot

- return type url

- if click on sauce demo link

- switch to it and enter credentials and run end to end flow

- return url

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

user defined Methods in java

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

there are 2 types of User defined methods

1) static method

2) Non static method

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

1) Static method in java

there are 4 types of methods

I) static method without return type and 0 arguments

ii) static method without return type and multiple arguments

iii) static method with return type and 0 arguments

iv) static method with return type and multiple arguments

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

2) Non Static method in java

there are 4 types of methods

I) Non static method without return type and 0 arguments

ii) Non static method without return type and multiple arguments

iii) Non static method with return type and 0 arguments

iv) Non static method with return type and multiple arguments

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

1) Static method in java

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

I) static method without return type and 0 arguments

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

syntax:

AccessModifier NonAccessModifier NonReturnType methodName()

{

//statement

}

e.g.

public static void abc()

{

//statement

}

what are the different Access Modifier in java?

there are 4 types of Access Modifier

1) public : if we declare any method as public then we can access this method anywhere within the project

2) private : If we declare any method as private then we can access this method within the class.

3) default : if we declare any method as default then we can access this method within the package.

4) protected : if we declare any method as protected then we can access this method within the package as well as outside the package by using inheritance concept.

What is non Access Modifier keyword in java

- static is non access modifier keyword.

- once we declare any method as static method, it will allocate memory location in class level.

- always we access the static method by class name dot method name.

what is non Return type keyword in java

- void is non return type

what is methodName?

- any method name,

where we declare the static method?

- we declare static method before the main method or after the main method but inside the class.

how to call static method in main method?

there are 3 different ways we can access the static method in main method

1) directly method name

2) by using class name dot method name \*\*\*\*\*\*\*\*\*

3) by using object name.

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

Scenario:

- create static method without return type and 0 arguments and write logic to find the max value from arrays.

public class Demo1

{

public static void findMaxValue()

{

int [] xyz = {100,200,300,4500,34,456};

int max= xyz[0];

for(int i=0;i<xyz.length;i++)

{

if(xyz[i]>max)

{

max=xyz[i];

}

}

System.out.println(max);

}

public static void main(String [] args)

{

//1) findMaxValue();

//2) Demo1.findMaxValue();

//3

Demo1 obj =new Demo1();

obj.findMaxValue();

}

}

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

package Tutorial10;

public class Demo1 {

// Static method without return type and 0 arguments/ parameter

public static void findMaxValue() {

int[] abc = { 100, 200,4457, 34 };

// imagine 0 index position value as max value

int max = abc[0];//4457

// iterate all values from array and compare with max variable value.

// and if max variable value is smaller than array value then re-assign the value

// for max variable

for(int i=0;i<abc.length;i++)

{

//34 > 4457

if(abc[i]>max)

{

max=abc[i];//4457

}

}

System.out.println(max);

}

public static void main(String[] args)

{

System.out.println("Main method is started");

//findMaxValue();

//Demo1.findMaxValue();

Demo1 obj =new Demo1();

obj.findMaxValue();

System.out.println("Main method is Ended");

}

}

=============================================================================================

Scenario 2:

- create 3 static method.

- access 1 static method directly

- access 2nd method by using class name

- access 3rd method by using object name

can we access all 3 static method by using class name

yes

can we access all 3 static method directly

yes

can we access all 3 static method by using object name

yes

can we access all 3 static method in another class

yes

can we access all 3 static method in another class by using class name

yes , but we have to take reference of method defined class.

can we access all 3 static method in another class directly

we can't access directly but, if we create relationship between 2 classes using extends keyword then easily we access static directly in another class.

can we access all 3 static method in another class by using object name?

yes

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

package Tutorial10;

public class Demo2 {

public static void abc() {

System.out.println("abc method in Demo2 class");

}

public static void pqr() {

System.out.println("PQR method in Demo2 class");

}

public static void xyz() {

System.out.println("XYZ method in Demo2 class");

}

public static void main(String[] args)

{

System.out.println("Pune");

abc();

System.out.println("Mumbai");

Demo2.pqr();

System.out.println("Delhi");

Demo2 obj =new Demo2();

obj.xyz();

System.out.println("Dubai");

}

}

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

package Tutorial10;

public class Sample2 {

public static void main(String[] args) {

Demo2.abc();

Demo2.pqr();

Demo2.xyz();

}

}

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

package Tutorial10;

public class Sample2 extends Demo2{

public static void main(String[] args) {

abc();

pqr();

xyz();

}

}

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

package Tutorial10;

public class Sample2 {

public static void main(String[] args) {

Demo2 obj =new Demo2();

obj.abc();

obj.pqr();

obj.xyz();

}

}

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

package Tutorial10;

public class Sample2 extends Demo2{

public static void main(String[] args) {

Sample2 obj =new Sample2();

obj.abc();

obj.pqr();

obj.xyz();

}

}

======================================================================================

package Tutorial10;

public class Demo3 {

public static void d1() {

System.out.println("D1 method in Demo3 class");

}

public static void d2() {

System.out.println("D2 method in Demo3 class");

}

public static void d3() {

System.out.println("D3 method in Demo3 class");

}

public static void main(String[] args) {

Demo3 obj = new Demo3();

obj.d1();

obj.d2();

obj.d3();

// Demo3.d1();

// Demo3.d2();

// Demo3.d3();

// d1();

// d2();

// d3();

}

}

package Tutorial10;

public class Sample3 extends Demo3

{

public static void main(String[] args) {

d1();

d2();

d3();

// Demo3 obj =new Demo3();

// obj.d1();

// obj.d2();

// obj.d3();

// Demo3.d1();

// Demo3.d2();

// Demo3.d3();

//

}

}

=============================================================================================

ii) static method without return type and multiple arguments or parameters

syntax;

AccessModifier NonAccessModifier NonReturnType MethodName(datatype variableName1,datatype variableName2,datatype variableName3)

{

//statement

}

what are the Access Modifier ?

I) public

2) private

3) default

4) protected

what is non Access modifier

static keyword

Non Return Type : void keyword

Method Name : any method name

datatype : byte, short, int, long, double, float, char, boolean, String, array , own class name

variableName1 :any name

where we create static method?

- before the main method or after the main method but inside the class

how to access static method in main method?

there are 3 ways

1) directly

2) by using className dot method name

3) by using object name

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

Scenario:

create static method with 2 String arguments and compare their values

public class Demo4

{

public static void compareString(String a, String b)

{

System.out.println(a.equals(b));

System.out.println(a.equalsIgnoreCase(b));

System.out.println(a==b);

System.out.println(a.subString(0).equals(b));

}

public static void main(String [] args)

{

Demo4.compareString("cucumber", "cucumber");

}

}

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

package Tutorial10;

public class Demo4 {

public static void compareString(String a,String b)

{

System.out.println(a.equals(b));//true

System.out.println(a.equalsIgnoreCase(a));//true

System.out.println(a==b);//true

System.out.println(a.substring(0));//RestApi

System.out.println(a.substring(0).equals(b));//true

System.out.println(a.equals(b)&&true);//true

System.out.println(a.equals(b)&&false);//false

System.out.println(!a.equals(b));//false

}

public static void main(String[] args) {

System.out.println("Main Method is Started");

Demo4.compareString("RestApi", "RestApi");

System.out.println("main method is ended");

}

}

package Tutorial10;

public class Demo5 {

/// create static method with 3 int args and find max value

public static void findMaxValue(int a, int b, int c)

{

if((a>b)&&(a>c))

{

System.out.println("A is max number");

}

else if(b>c)

{

System.out.println("B is Max Number");

}

else

{

System.out.println("C is max Number");

}

}

public static void main(String[] args)

{

Demo5.findMaxValue(100, 200, 3002);

findMaxValue(120, 32, 30);

}

}

=====================================================================================

Ass:

Create static method with 1 int arguments

conditions

i) if a number is between 1 to 100 then print a small number

ii) if a number is between 101 to 1000 then print a number is medium

iii) if a numbers is between 1001 to 10000 the print a number is large

iv) if a number is above 10001 then a number is biggest number

v) if a number is below 1 then a number is zero or negative number.

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

Create static method with 1 int arguments

conditions

i) if marks is below 25 then display the grade as F

ii) if marks is 25 to 45 then display the grade as E

iii) if marks is 46 to 55 then display the grade as D

iv) if marks is 56 to 65 then display the grade as C

v) if marks is 66 to 75 then display the grade as B

vi) if marks is above 75 then display the grade as A

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

// 1. static method without return type and int array args or parameters and find max value from array

// 2. static method without return type and int array or parameters and find min value from array

1. static method without return type and 2 boolean args or parameters

and perform all logical operator operation (AND operator and OR Operator)

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

create static method with 1 String arguments and write logic to find numbers, upper case character, lower case character and special character VVVVVVVVVVVVVVVVVVVVVIIIIIIIIMMMMMMMMMMMMPPPPPPPPPPPPPPPPPP

public class Demo9

{

public static void findValues(String abc)

{

//find numbers

String numbers = abc.replaceAll("[^0-9]","");

System.out.println(numbers);

//find upper case letter

String uppercase = abc.replaceAll("[^A-Z]","");

System.out.println(uppercase);

//find lower case letter

String lowercase = abc.replaceAll("[^a-z]","");

System.out.println(lowercase);

//find special character

String specialCharacter = abc.replaceAll("[a-zA-Z0-9]","");

System.out.println(specialCharacter);

}

public static void main(String[] args)

{

Demo9.findValues("90912abcABC2&%#$a234");

}

}

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

package Tutorial10;

public class Demo9 {

public static void findGroups(String abc)

{

//find numbers : don't replace 0-9 to "" (blank String)

String numbers =abc.replaceAll("[^0-9]", "");

System.out.println(numbers);

//find upper case letter : dont replace A-Z t "" blank String

String upperCase = abc.replaceAll("[^A-Z]", "");

System.out.println(upperCase);

//find lower case letter : dont replace a-z to "" (Blank String)

String lowerCase = abc.replaceAll("[^a-z]", "");

System.out.println(lowerCase);

//find special character : replace 0-9,A-Z and a-z to blank String

String specialCharacter = abc.replaceAll("[0-9A-Za-z]", "");

System.out.println(specialCharacter);

}

public static void main(String[] args)

{

Demo9.findGroups("9AR^55%@%ahayA123P");

}

}

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

create static method with 1 String arguments and find upper case and lower letter

combine the upper case letter and lower case letter and reverse the combined string

at the time of calling these method use below format

"%^&ABOLI01022022shinde\*&\*";

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

create static method with 1 String arguments lower letter

reverse the string

and check find string and reverse is Palindrome or not

at the time of calling these method use below format

"%^m&010a220d22a\*&m\*";

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

I) Static method without return type and 0 arguments

ii) Static method without return type and multiple arguments

iii) Static method with return type and 0 arguments

iv) Static method with return type and multiple arguments

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

iii) Static method with return type and 0 arguments

syntax:

AccessModifier nonAccessModifier returntype methodName()

{

//statement

return variableName/value;

}

e.g.

public static String abc()

{

//statement

return "selenium";

}

what are the different access modifier in java

there are 4 types

1) public : if declare any method as public then we can access this method anywhere in project

2) private : if we declare any method as private then we can access this method within the class

3) default : if we declare any method as default then we can access this method within the package

4) protected : if we declare any method as protected then we can access this method within the package as well as outside the package

by using inheritance concepts.

what is non Access modifier keyword

static keyword

what is return type keywords

i) byte

ii) short

iii) int

iv) long

v) float

vi) double

vii) char

viii) boolean

ix) String

x) array

xi) own class name

xii) interface

what is method name?

any method name

where we write the static method?

before the main method or after the main method but inside the class.

how to access static method in main method?

there are 3 ways we can access in main method

i) directly method name

ii) by using class name dot method name

iii) by using object name

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

Scenario:

create static method with String return type and

find upper case and lower case string and then concat the string and return the combined string.

public class Demo1

{

public static String concatString()

{

String abc="2553^%rohitPATIL&%%8090";

String uppercase= abc.replaceAll("[^A-Z]","");

String lowerCase = abc.replaceAll("[^a-z]","");

String xyz = uppercase.concat(lowerCase);

return xyz;

}

public static void main(String [] args)

{

String pqr = Demo1.concatString();

System.out.println(pqr);

}

}

package Tutorial11;

public class Demo1 {

// Scenario:create static method with String return type and

public static String concatString()

{

String a="172rohit@#$$PATIL9182";

//String xyz = a.replaceAll("[^A-Za-z]", "");

String uc =a.replaceAll("[^A-Z]", "");

String lc = a.replaceAll("[^a-z]", "");

String xyz = lc.concat(uc);

return xyz;

}

public static void main(String[] args) {

String pqr = Demo1.concatString();

System.out.println(pqr);

}

}

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

anagram String means all character should be same in String. VVVVVIMMMPPPPPPPP

scenario:

create static method with String return type and

and declare the 2 string and check string is anagram or not?

return the String is anagram or not

public class Demo2

{

public static String checkAnagramString()

{

String a="race";

String b="care";

//it will convert string into char [] array

char [] abc = a.toCharArray();

char [] pqr = b.toCharArray();

//it used to convert the arrays values as per the ascending order.

Arrays.sort(abc); //acer

Arrays.sort(pqr);//acer

if(Arrays.equals(abc, pqr)

{

return "String is anagram";

}

else

{

return "String is not anagram";;

}

}

public static void main(String[] args) {

String p = Demo2.checkAnagramString();

System.out.println(p);

}

}

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

package Tutorial11;

import java.util.Arrays;

public class Demo2 {

//check String is anagram or not

public static String checkAnagramString()

{

String a="madam";

String b="care";

//convert String into char [] array

char [] abc= a.toCharArray();

char [] pqr = b.toCharArray();

//sort the char [] values as per the ascending order

Arrays.sort(abc);//aadmm

Arrays.sort(pqr);//acer

//compare the both array values after sorting

if(Arrays.equals(abc, pqr))

{

return "String is anagram";

}

else

{

return "String is not anagram";

}

}

public static void main(String[] args) {

String p = Demo2.checkAnagramString();

System.out.println(p);

}

}

=====================================================================

Scenario:

find the max values from array and return the max value.

package Tutorial11;

public class Demo3 {

//find the max values from array and return the max value.

public static int findMaxValue()

{

int [] abc = {10,330,4344,2454,355,5555};

int max = abc[2];

for (int i = 0; i < abc.length; i++)

{

if(abc[i]>max)

{

max=abc[i];

}

}

return max;

}

public static void main(String[] args) {

int p = findMaxValue();

System.out.println(p);

}

}

===========================================================================

iv) Static method with return type and multiple arguments

syntax

AccessModifier nonAccessModifier returnType methodName(datatype variableName, datatype variableName,datatype variableName)

{

//statement

return variableName/value;

}

Access Modifier :public , private, default and protected

non Access Modifier :static

Return type : byte, short, int, long, float, double, char, boolean, String, array , own class name

method Name :any name

how to access static method in main method / static method

there are 3 ways we can access

I) directly method name

ii) by using class name dot method name

iii) by using object name

where we define the static method

before the main method or after the main method but inside the class.

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

Scenario:

create static method with String return type and with 2 String arguments

and check the string is anagram or not

public static String checkAnagramString(String abc, String pqr)

{

//convert the String into the char []

char [] a = abc.toCharArray();

char [] b = pqr.toCharArray();

//sort the char [] arrays values as per the ascending order

Arrays.sort(a);

Arrays.sort(b);

//compare the both arrays values

if(Arrays.equals(a, b))

{

return "String is anagram";

}

else

{

return "String is not anagram";

}

}

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

package Tutorial11;

import java.util.Arrays;

public class Demo4 {

public static boolean checkAnagramString(String abc,String pqr)

{

//convert the String into char [] array

char [] a= abc.toCharArray();

char [] b= pqr.toCharArray();

//sort the char [] arrays values as per the ascending order

Arrays.sort(a);

Arrays.sort(b);

//compare the both array values

if(Arrays.equals(a, b))

{

return true;

}

else

{

return false;

}

}

public static void main(String[] args)

{

boolean p = Demo4.checkAnagramString("race", "madam");

if(p)

{

System.out.println("String is anagram");

}

else

{

System.out.println("String is not anagram");

}

}

}

===================================================================================

Ass:

create static method with boolean return type and 0 args

and search Pusha the Rise keyword in google

- click on Wikipedia link and capture the release date

- click on IMDB link and capture the release date

- and compare Wikipedia date vs IMDB date if both are same return true keyword or false keyword.

===================================================================================

package Tutorial11;

public class Demo5 {

// find max value out of 3 values

// create static method with int return type and 3 int args

public static int findMaxValue(int a, int b, int c)

{

if((a>b)&&(a>c))

{

return a;

}

else if(b>c)

{

return b;

}

else

{

return c;

}

}

public static void main(String[] args) {

int p = Demo5.findMaxValue(100, 2000, 300);

System.out.println(p);

}

}

=========================================================================

package Tutorial11;

public class Demo7 {

public void abc()

{

System.out.println("abc method ");//2 3

}

public void xyz()

{

abc();

Demo7 obj =new Demo7();

obj.abc();

System.out.println("xyz method");//4

}

public static void main(String[] args) {

System.out.println("MMS");//1

Demo7 obj1 =new Demo7();

obj1.xyz();

System.out.println("MME");//5

}

}

package Tutorial11;

public class Demo8 {

public static void abc()

{

System.out.println("abc static method");//2 3 4

}

public void xyz()

{

abc();

Demo8.abc();

Demo8 obj =new Demo8();

obj.abc();

System.out.println("xyz non static method");//5

}

public static void main(String[] args)

{

System.out.println("MMS");//1

Demo8 obj1 =new Demo8();

obj1.xyz();

System.out.println("MME");//6

}

}

=====================================================================================

How to access static method in main method/static method?

there are 3 ways

- directly method name

- class name dot method name

- by using object name

how to access static method in non static method?

there are 3 ways

- directly method name

- class name dot method name

- by using object name

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

how to access non static method in non static method

there are 2 ways

- directly method name

- by using object name

how to access non static method in static method or main method

- by using object name

===================================================================================

i) Non static method without return type and 0 args / parameter

ii) Non static method without return type and multiple args / parameter

iii) Non static method with return type and 0 args / parameter

iv) Non static method with return type and multiple args / parameter

v) static method without return type and 0 args / parameter

vi) static method without return type and multiple args / parameter

vii) static method with return type and 0 args / parameter

viii) static method with return type and multiple args / parameter

=======================================================================================

Ass:

create static method with boolean return type and 0 args

and search Pusha the Rise keyword in google

- click on Wikipedia link and capture the release date

- click on IMDB link and capture the release date

- and compare Wikipedia date vs IMDB date if both are same return true keyword or false keyword.

=======================================================================================

open amazon.in

create static method with int return type and 1 string arguments

return max price

from main method search

at the time of calling pass below keyword

Samsung mobile phone

apple mobile phone

compare both value and print Samsung or apple keyword.

public class Demo1

{

public static int searchMobileAndFindMaxValue(String mobileName)

{

//connect browser

//incognito

//up casting

//maximize

//implicit wait

//page load time

//open a url

//find search button and use arguments name

..find all price

iterate all price

find max price

return max price

}

public static void main(String [] args)

{

int samsungprice = Demo1.searchMobileAndFindMaxValue("Samsung mobile Phone")

int appleprice = Demo1.searchMobileAndFindMaxValue("apple mobile Phone")

if(samsungprice>appleprice)

{

System.out.println("Samsung price is bigger ");

}

else

{

System.out.println("apple price is bigger ");

}

}

}

Non Primitive datatypes

1) String

2) arrays

3) class

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

What is String?

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

- String is sequence of character which is written in double quote.

- String may have numbers, upper case character, lower case character and special characters.

syntax:

Datatype variablename=values;

e.g.

String abc = "Welcome to automation testing class";

String a= "12345";

String b = "^%$$#";

String class methods

1) length() method

- it is used to capture the length of string.

- return type of length() method is int.

syntax:

String a="mumbai";

int b = a.length();

System.out.println(b);

package StringTutorial1;

public class Demo1 {

public static void main(String[] args) {

String a="Welcome to automation testing class";

int b = a.length();

System.out.println(b);

}

}

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

2) toUpperCase() method

- it is used to convert the string into the upper case.

- return type is String

String a="mumbai";

String b = a.toUpperCase();

System.out.println(b);

package StringTutorial1;

public class Demo2 {

public static void main(String[] args) {

String a="mumbai";

String b = a.toUpperCase();

System.out.println(b);

}

}

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

3) toLowerCase() method

- it will convert String into lower case.

- return type is String.

String a ="PUNE";

String b = a.toLowerCase();

System.out.println(b);

package StringTutorial1;

public class Demo3 {

public static void main(String[] args) {

String a="PUNE";

String b = a.toLowerCase();

System.out.println(b);

}

}

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

4) startsWith() method

- it is used to check specific string is starts with specific character or not?

- return type is boolean (true/false).

String abc="ui automation using selenium webdriver";

boolean b = abc.startsWith("ui automation");

System.out.println(b);

package StringTutorial1;

public class Demo4 {

public static void main(String[] args) {

String abc="ui Automation testing using selenium webdriver";

boolean b = abc.startsWith("ui Automation");

System.out.println(b);

}

}

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

5) endsWith() method

- it will check specific string is ends with specific character or not

- return type is boolean

syntax:

String abc="ui automation testing using selenium webdriver";

boolean b = abc.endsWith("webdriver");

System.out.println(b);

package StringTutorial1;

public class Demo5 {

public static void main(String[] args) {

String abc = "ui Automation testing using selenium webdriver";

boolean b = abc.endsWith("ium webdriver");

System.out.println(b);

}

}

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

6) contains() method

- it will check specific string contains specific keyword is present or not in any locations of the String.

- return type is boolean

syntax:

String abc = "ui Automation testing using selenium webdriver";

boolean b = abc.contains("Automation testing using");

System.out.println(b);

package StringTutorial1;

public class Demo8 {

public static void main(String[] args) {

String abc = "ui Automation testing using selenium webdriver";

boolean b = abc.contains("ium webdriver");

System.out.println(b);

}

}

package StringTutorial1;

public class Demo6 {

public static void main(String[] args) {

String a="SELEnium";

int b = a.length();

System.out.println(b);

String c= a.toUpperCase();

System.out.println(c);

String d = a.toLowerCase();

System.out.println(d);

boolean e = a.startsWith("SE");

System.out.println(e);//

boolean f = a.endsWith("ium");

System.out.println(f);

boolean g = a.contains("En");

System.out.println(g);

}

}

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

String methods in java

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

1) length() method

2) toUpperCase() method

3) toLowerCase() method

4) startsWith()method

5) endsWith() method

6) contains() method

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

7) equals() method

- it used to compare the 2 different Object.

- return type is boolean.

- it used to compare the ""actual content of the String/object".

syntax:

String a="selenium";

String b= "Selenium";

boolean c = a.equals(b);

System.out.println(c);

8) equalsIgnoreCase() method

- it is compare the 2 different String and it will ignore upper case and lower case.

- return type is boolean.

- it used to compare the actual content of string and it will ignore upper case and lower case.

syntax:

String a="selenium";

String b= "SELENIUM";

boolean c = a.equalsIgnoreCase(b);

System.out.println(c);

9) == operator

- it used to compare the different objects.

- return type is boolean

- this method is used to compare the """memory location of String/object.""".

syntax:

String a="selenium";

String b= "selenium";

System.out.println(a==b);

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

Diff between equals() and == operator VVVVVIMMMPPPPPP

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

package StringTutorial2;

public class Demo1 {

public static void main(String[] args) {

String a="selenium";

String b="selenium";

boolean c = a.equals(b);

System.out.println(c);//true

boolean d = a.equalsIgnoreCase(b);

System.out.println(d);//true

System.out.println(a==b);//true

}

}

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

package StringTutorial2;

public class Demo2 {

public static void main(String[] args) {

String c="selenium";//SCP Memory ---> same Memory location---> same Memory location

String d="selenium";//SCP Memory---> same Memory location

//Heap Memory--- different Memory location

String a = new String("selenium");

String b = new String("selenium");

System.out.println(a.equals(b));//true

System.out.println(a==b);//false

System.out.println(c.equals(d));//true

System.out.println(c==d);//true

}

}

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

10) concat() method

- it used to combine the 2 different string

- return type is String.

syntax:

String a="selenium";

String b="testing";

String c = a.concat(b);

System.out.println(c);

11) combine the string using + operator

- we can combine any values using + operator.

syntax:

String a="selenium";

String b="testing";

String c = a+b;

System.out.println(c);

package StringTutorial2;

public class Demo3 {

public static void main(String[] args) {

String a="selenium";

String b="testing";

String c = a.concat(b);

System.out.println(c);

String d= a+b;

System.out.println(d);

}

}

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

package StringTutorial2;

public class Demo4 {

public static void main(String[] args) {

System.out.println(10+10);//20

System.out.println(10+10+"selenium");//20selenium

System.out.println(10+"selenium"+10);//10selenium10

//String + int = String

System.out.println("selenium"+10);//selenium10

System.out.println("selenium"+10+10);//selenium1010

}

}

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

package StringTutorial2;

public class Demo5 {

public static void main(String[] args) {

System.out.println(10+"java"+10);//10java10

System.out.println("java"+(10));//java10

System.out.println("java"+ (10+10));//java20

System.out.println("java"+10+10);//java1010

}

}

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

12) charAt() method

- it used to capture the character as per the index position.

- return type is char datatype.

syntax:

String a="Pune";

char b = a.charAt(2);

System.out.println(b);

package StringTutorial2;

public class Demo6 {

public static void main(String[] args) {

String a="Pune";

System.out.println(a.charAt(3));//e

System.out.println(a.charAt(2));//n

System.out.println(a.charAt(1));//u

System.out.println(a.charAt(0));//P

}

}

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

Reverse String using charAt() method VVVVVVVVIMPPPPPPPPPPPPP

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

package StringTutorial2;

public class Demo6 {

public static void main(String[] args) {

//enup

String a="selenium";

int p = a.length()-1;//7

for(int i=p;i>=0;i--)

{

char b = a.charAt(i);

System.out.print(b);//muineles

}

}

}

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

package StringTutorial2;

public class Demo7 {

public static void main(String[] args) {

String a="cucumber";

int b = a.length()-1;

for(int i=b;i>=0;i--)

{

char c = a.charAt(i);

System.out.print(c);

}

}

}

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

13) trim() method

- it used to remove space from the start and end of string.

- return type is string.

syntax:

String a=" selenium testing ";

String b = a.trim();

System.out.print(b);

package StringTutorial2;

public class Demo8 {

public static void main(String[] args) {

String a=" selenium testing ";

System.out.println(a);

String b = a.trim();

System.out.println(b);

}

}

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

14) replace() method

- it used to replace the String.

- return type is string.

syntax:

String a="selenium testing";

String b = a.replace("selenium","cucumber");

System.out.println(b);//cucumber testing

package StringTutorial2;

public class Demo9 {

public static void main(String[] args) {

String a="selenium testing selenium";

String b = a.replace("selenium", "cucumber");

System.out.println(b);

}

}

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

15) replaceAll() method

it used to replace the string

return type is string

syntax:

String a="selenium testing";

String b = a.replaceAll("selenium","cucumber");

System.out.println(b);//cucumber testing

package StringTutorial2;

public class Demo9 {

public static void main(String[] args) {

String a="selenium testing selenium";

String b = a.replaceAll("selenium", "cucumber");

System.out.println(b);

}

}

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

package StringTutorial2;

public class Demo10 {

public static void main(String[] args) {

//remove the 2 or more space in between words

String a="selenium testing";

String b= a.replaceAll("\\s+", " ");

System.out.println(b);

}

}

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

package StringTutorial2;

public class Demo11 {

public static void main(String[] args) {

String a=" ui cucumber testing ";

String b = a.trim();

System.out.println(b);

String c = b.replaceAll("\\s+", " ");

System.out.println(c);

}

}

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

Remove the Starts, mid and ending space and reverse the string VVVVVVVVVIMMMMPPPPPP

package StringTutorial2;

public class Demo11 {

public static void main(String[] args) {

String a=" ui cucumber testing ";

String b = a.trim().replaceAll("\\s+", " ");

System.out.println(b);

int c = b.length()-1;

for(int i=c;i>=0;i--)

{

System.out.print(b.charAt(i));

}

}

}

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

16) subString() method

- it return the string based on index position.

- return type is string.

syntax:

String a="selenium";

String b = a.subString(6); //it will print string from index position 6 to end of string.

System.out.println(b);//um

String c = a.subString(0,3);

System.out.println(b);//sel //it will print string from index position 0 to 3-1 (2) index position.

package StringTutorial2;

public class Demo12 {

public static void main(String[] args) {

String a="selenium";

System.out.println(a.substring(4));//nium

System.out.println(a.substring(0,5));//selen

System.out.println(a.substring(0));//selenium

System.out.println(a.substring(3));//enium

System.out.println(a.substring(4,7));//niu

}

}

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

// Swap the String using third variables VVVVVIMMMPPPPPPPPPPPP

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

package StringTutorial2;

public class Demo13 {

public static void main(String[] args) {

// Swap the String using third variables

String a="java";

String b="selenium";

String c=a;

a=b;

System.out.println(a);

b=c;

System.out.println(b);

}

}

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

17) indexOf() method

- this method is used to find the first index of specific character.

- return type int

syntax:

String a="selenium";

int b = a.indexOf('e');

System.out.println(b);

18) lastIndexOf() method

- this method is used to find the last index of specific character.

- return type int

syntax:

String a="selenium";

int b = a.lastIndexOf('e');

System.out.println(b);

package StringTutorial2;

public class Demo14 {

public static void main(String[] args) {

String a="java";

System.out.println(a.indexOf('a'));//

System.out.println(a.lastIndexOf('a'));

}

}

==============================================

19) valueOf() method

- it convert the any datatype values into the string.

- return type is string.

syntax:

int a=100;

String b = String.valueOf(a);

System.out.println(b);

package StringTutorial2;

public class Demo15 {

public static void main(String[] args) {

int a=100;

String b = String.valueOf(a);

System.out.println(b);

}

}

package StringTutorial2;

public class Demo15 {

public static void main(String[] args) {

boolean a=true;

String b = String.valueOf(a);

System.out.println(b);

}

}

package StringTutorial2;

public class Demo15 {

public static void main(String[] args) {

char a='p';

String b = String.valueOf(a);

System.out.println(b);

}

}

================================================================================

package StringTutorial2;

public class Demo16 {

public static void main(String[] args) {

String a="selenium";

String b="selenium";

System.out.println(a.length());//8

System.out.println(a.toUpperCase());//SELENIUM

System.out.println(a.toLowerCase());//selenium

System.out.println(a.startsWith("sel"));//true

System.out.println(a.endsWith("um"));//true

System.out.println(a.contains("ni"));//true

System.out.println(a.equals(b));//true

System.out.println(a.equalsIgnoreCase(b));//true

System.out.println(a==b);//true

System.out.println(a.concat(b));//seleniumselenium

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

System.out.println(a.charAt(5));//i

System.out.println(a.trim());//selenium

System.out.println(a.replace("sel", "cucumber"));//cucumberenium

System.out.println(a.replaceAll("cucumber", "sel"));//selenium

System.out.println(a.substring(5));//ium

System.out.println(a.substring(0,5));//selen

System.out.println(a.replace("\\s+", " "));//selenium

System.out.println(a.indexOf('e'));//1

System.out.println(a.lastIndexOf('e'));//3

}

}

======================================================================

package StringTutorial2;

public class Demo16 {

public static void main(String[] args) {

String a="selenium";

String b="selenium";

System.out.println(a.length());//8

System.out.println(a.toUpperCase());//SELENIUM

System.out.println(a.toLowerCase());//selenium

System.out.println(a.startsWith("sel"));//true

System.out.println(a.endsWith("um"));//true

System.out.println(a.contains("ni"));//true

System.out.println(a.equals(b));//true

System.out.println(a.equalsIgnoreCase(b));//true

System.out.println(a==b);//true

System.out.println(a.concat(b));//seleniumselenium

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

System.out.println(a.charAt(5));//i

System.out.println(a.trim());//selenium

System.out.println(a.replace("sel", "cucumber"));//cucumberenium

System.out.println(a.replaceAll("cucumber", "sel"));//selenium

System.out.println(a.substring(5));//ium

System.out.println(a.substring(0,5));//selen

System.out.println(a.replace("\\s+", " "));//selenium

System.out.println(a.indexOf('e'));//1

System.out.println(a.lastIndexOf('e'));//3

System.out.println(String.valueOf(100.34f));//100.34

}

}

========================================================================

Revision

arrays:

- declare int type array

- and print all values using normal for loop

using enhance for loop

using Arrays.toString() method.

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

Array:

- declare String type array

- and print all values using normal for loop

using enhance for loop

using Arrays.toString() method.

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

Array:

- declare String type array

- and verify the specific value is present or not in array using if else condition.

- normal for loop

using enhance for loop

using Arrays.toString() method.

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

Array:

- declare int type array

- and find max values from array using normal for loop

using enhance for loop

using sort() method

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

Array:

- declare int type array

- and find min values from array using normal for loop

using enhance for loop

using sort() method

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

Array:

- declare String type array

- and reverse all array values using normal for loop

using enhance for loop

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

write down all String method in notepad file with detailed descriptions.

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

Open google application capture the title and capture the url

package demo;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Demo1 {

public static void main(String[] args) {

//connect to actual browser

System.setProperty("webdriver.chrome.driver", "E:\\softech class\\chromedriver-win64\\chromedriver-win64\\chromedriver.exe");

//Up Casting

WebDriver driver = new ChromeDriver();

//open url

driver.get("https://www.facebook.com");

String title = driver.getTitle();

System.out.println(title);

String url = driver.getCurrentUrl();

System.out.println(url);

driver.navigate().to("https://www.redbus.com");

driver.navigate().back();

driver.navigate().forward();

}

}

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

method name description of method return type of method

1) length() it will capture the length of string int

2) toUpperCase() it is used to convert string into uppercase String

3) toLowerCase() it is used to convert String into lowercase String

4) startsWith("prefix value") it is used to check specific string is startswith specific

character or not boolean

5) endsWith("suffix value") it is used to check specific string is endswith specific

character or not. boolean

6)contains(String str) it will check specific keyword present or not in any

location of string. boolean

7)equals(Object O) it is used to check compare the actual content of string boolean

8)equalsIgnoreCase(String str) it is used to ignore the uppercase and lowercase characters

of string and compare the actual content. boolean

9)== operator it is used to compare the memory location of string/object boolean

10)concat(String str) it is used to combine the two different strings. String

11)+ Operator it is used to combine the two different objects. String

12)charAt(int index) it is used to capture the character as per index position. char

13)trim() it will remove starting and ending space. String

14)replace(String oldstr,String newStr) it will replace the string from old string to new string String

15)replaceAll(String oldStr,String newStr) it will replace the string from old string to new string String

16)subString(int index) it will return string as per the index position String

17)subString(int startIndex,int endIndex) it will return string from start index to end index String

18)indexOf(char ch) it will return index position of specific character int

19)lastIndexOf(char ch) it will return last index position of specific character int

20)valueOf(Object o) it will convert any data type value into string String

class Demo

{

public static void main(String args[])

{

//connect to actual browser

System.setProperty("webdriver.chrome.driver","chromedriver path");

//Up Casting

WebDriver driver = new ChromeDriver();

//open url

driver.get("https://www.facebook.com");

//capture title

String title = driver.getTitle();

System.out.println(title);

//capture url

String url = driver.getCurrentUrl();

System.out.println(url);

}

}

java programs:

1)

- declare int type array

- and print all values using normal for loop

-using enhance for loop

-using Arrays.toString() method.

package java\_tutorial1;

import java.util.Arrays;

public class Practice {

public static void main(String[] args) {

// and print all values using normal for loop

int a[]= {100,900,80,700,78};

for(int i=0;i<=a.length-1;i++)

{

System.out.println(a[i]);

}

//using enhance for loop

for(int ab:a)

{

System.out.println("using enhance for loop " +ab);

}

//using Arrays.toString()

System.out.println(Arrays.toString(a));

}

}

============================================================================================================================

2)Array:

- declare String type array

- and print all values using normal for loop

using enhance for loop

using Arrays.toString() method.

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

package java\_tutorial1;

import java.util.Arrays;

public class Practice1 {

public static void main(String[] args) {

String a[]= {"java","BDD","cucumber","selenium"};

// and print all values using normal for loop

for(int i=0;i<a.length;i++)

{

System.out.println(a[i]);

}

//using enhance for loop

for(String ab:a)

{

System.out.println(ab);

}

//using Arrays.toString() method.

System.out.println(Arrays.toString(a));

}

}

============================================================================================================================

3)Array:

- declare String type array

- and verify the specific value is present or not in array using if else condition.

- normal for loop

using enhance for loop

using Arrays.toString() method.

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

package java\_tutorial1;

import java.util.Arrays;

public class Practice2 {

public static void main(String[] args) {

String str[]= {"Pune","Mumbai","Sangli","kolhapur"};

//using for loop

for(int i=0;i<=str.length-1;i++)

{

if(str[i].equals("Pune"))

{

System.out.println("test case is pass");

}

}

//using enhance for loop

for(String a:str)

{

if(a.equals("Pune"))

{

System.out.println("test case is pass");

}

}

//using Arrays.toString() method.

boolean ab =Arrays.toString(str).contains("Pune");

if(ab)

{

System.out.println("test case is pass");

}

}

}

============================================================================================================================

4)Array:

- declare int type array

- and find max values from array using normal for loop

-using enhance for loop

-using sort() method

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

package java\_tutorial1;

import java.util.Arrays;

public class Practice3 {

public static void main(String[] args) {

//using for loop

int a[]= {100,200,300,400,500};

int max=a[0];

for(int i=0;i<=a.length-1;i++)

{

if(a[i]>max)

{

max=a[i];

}

}

System.out.println(max);

//using enhance for loop

int max1 = a[0];

for(int ab:a)

{

if(ab>max1)

{

max1=ab;

}

}

System.out.println(max1);

//using sort() method

Arrays.sort(a);

int len = a.length;

System.out.println(a[len-1]);

}

}

============================================================================================================================

Array:

- declare int type array

- and find min values from array using normal for loop

using enhance for loop

using sort() method

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

package java\_tutorial1;

import java.util.Arrays;

public class Practice4 {

public static void main(String[] args) {

//using for loop

int a[]= {120,20,400,60,5};

int min=a[0];

for(int i=0;i<=a.length-1;i++)

{

if(a[i]<min)

{

min=a[i];

}

}

System.out.println(min);

//using enhance for loop

int min1=a[0];

for(int ab:a)

{

if(ab<min1)

{

min1=ab;

}

}

System.out.println(min1);

//using sort() method

Arrays.sort(a);

System.out.println(a[0]);

}

}

============================================================================================================================

Array:

- declare String type array

- and reverse all array values using normal for loop

-using enhance for loop

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

package java\_tutorial1;

public class Practic5 {

public static void main(String[] args) {

String str[]= {"selenium","TestNG","Cucumber","POM","Github","Maven"};

//using for loop

for(int i=0;i<=str.length-1;i++)

{

String b =str[i];

for(int j=b.length()-1;j>=0;j--)

{

System.out.print(b.charAt(j));

}

System.out.println();

}

//using enhance for loop

for(String ab:str)

{

for(int i=ab.length()-1;i>=0;i--)

{

System.out.print(ab.charAt(i));

}

System.out.println();

}

}

}

**Java OOPs (Object Oriented Programming Language) concept VVVVVVVVVIIIIIIIMMMMMMMPPPPPPPPPPPP**

there are 4 main pillar in OOPs

I) inheritance VVVVVVVVVIIIIIIIMMMMMMMPPPPPPPPPPPP

2) Polymorphism VVVVVVVVVIIIIIIIMMMMMMMPPPPPPPPPPPP

3) Abstraction VVVVVVVVVIIIIIIIMMMMMMMPPPPPPPPPPPP

4) Encapsulation VVVVVVVVVIIIIIIIMMMMMMMPPPPPPPPPPPP

**I) inheritance**  VVVVVVVVVIIIIIIIMMMMMMMPPPPPPPPPPPP

- inheritance is process of reusing class members from one class to another class.

- Class members means variables and methods

- we achieve inheritance by using """""extends keyword."""".

- the class where class members are getting inherited it called as parent class or super class or base class.

- the class, in which class members are getting inherited is called as child class or derived class or sub class.

there are 5 types of inheritance

I) single level inheritance

2) multilevel inheritance

3) Hierarchical inheritance

4) multiple inheritance

5) Hybrid inheritance

**I) single level inheritance**

- in single level inheritance we extends one class in another class.

syntax:

public class Sample1

{

}

public class Demo1 extends Sample1

{

}

Sample1: Sample1 class is parent of Demo1 class

Demo1 : Demo1 class is child class of Sample1 class.

- in Sample1 class we defined the 5 methods and Demo1 class we defined the 4 Methods.

**Scenario 1:**

- if we create object of Sample1 and reference of Sample1 class then we can access all the methods from Sample1 class only.

Sample1 obj1 =new Sample1();

**Scenario 2:**

- if we create object of Demo1 class and reference of Demo1 class then we can access all the methods from Demo1 class as well Sample1 class.

Demo1 obj =new Demo1();

**Scenario 3: up casting**

- if we create object of Demo1 class and reference of Sample1 class then we can access all the methods from Sample1 class only.

Sample1 obj =new Demo1();

: if we mention reference of Parent Class and object of Child Class this allow to access all the methods from Parent Class.

ParentClass obj =new ChildClass();

Note:

1) if have same non static method present inside the Parent Class and Child Class, """"then priority goes to Object Creation class."""

2) if have same static method present inside the Parent class and child class then """priority goes to reference class"""".

package Tutorial3;

public class ClassB extends ClassA {

public void a1() {

System.out.println("A1 method in ClassB");

}

public void b1() {

System.out.println("B1 method in ClassB");

}

public void b2() {

System.out.println("B2 method in ClassB");

}

public static void main(String[] args) {

// if we mention reference of ParentClass and object of ChildClass

// this allow to access all the from ParentClass only.

// but if """""same non static method parent inside the ParentClass and ChildClass""""

//then it will call same non static method from object creation class.

ClassA obj = new ClassB();

obj.a1();//ClassB

obj.a2();//ClassA

obj.a3();//ClassA

}

}

package Tutorial4;

public class ClassA {

public static void a1() {

System.out.println("A1 static method in ClassA");

}

public static void a2() {

System.out.println("A2 static method in ClassA");

}

public static void a3() {

System.out.println("A3 static method in ClassA");

}

public static void main(String[] args) {

// if we mention refernce of ParentClass and Object of ChildClass

// this allow to access all the methods from PARENT CLASS ONLY

// But IF SAME STATIC METHOD PRESENT INSIDE THE PARENT CLASS AND CHILD CLASS

// then priority goes to reference class for same static method

ClassA obj = new ClassB();

obj.a1();

obj.a2();

obj.a3();

}

}

**Revise:**

Scenario 1: if mention reference of Parent class and Object of Parent Class this allow to access all the methods from Parent class only

ParentClass obj =new ParentClass();

Scenario 2:if mention reference of Child class and Object of Child Class this allow to access all the methods from Parent class and Child Class

ChildClass obj =new ChildClass();

Scenario 3: if mention reference of Parent class and Object of Child Class this allow to access all the methods from Parent class only

ParentClass obj =new ChildClass();

- if we have same static method present inside the Parent class and child class then priority goes to reference

- if we have same non static method present inside the Parent class and child class then priority goes to Object Creation.

**2) Multilevel inheritance:**

- in Multilevel inheritance, we have to extends one class in another class, and another class in another class.

- just imagine we have 3 classes , ClassA , ClassB and ClassC.

- we have extends ClassA in ClassB and

we have extends ClassB in ClassC.

ClassA: ClassA is parent of ClassB,

ClassA grand parent class of ClassC

ClassB: ClassB is parent of ClassC

ClassB is child class of ClassA

ClassC: ClassC is child class of ClassB

grand child of ClassA.

just take example in ClassA we have defined 10 methods , in ClassB we have defined 5 Methods and in ClassC we have defined 3 Methods

Scenario1 :

ClassA obj1 =new ClassA(); //we can access 10 methods from ClassA

Scenario 2:

ClassB obj2 =new ClassB(); //we can access 15 methods from ClassA and ClassB

Scenario 3:

ClassC obj3 =new ClassC(); //we can access 18 methods from ClassA , ClassB and ClassC

scenario 4:

ClassA obj4 =new ClassB(); //we can access 10 methods from ClassA

Scenario 5:

ClassA obj6 =new ClassC(); //we can access 10 methods from ClassA

Scenario 6:

ClassB obj5 =new ClassC(); //we can access 15 methods from ClassA and ClassB

public class ClassA {

public void a1() {

System.out.println("A1 method");

}

public void a2() {

System.out.println("A2 method");

}

public void a3() {

System.out.println("A3 method");

}

public void a4() {

System.out.println("A4 method");

}

public static void main(String[] args) {

// ClassB obj = new ClassC();

//

// obj.a1();

// obj.a2();

// obj.a3();

// obj.a4();

// obj.b1();

// obj.b2();

// obj.c1();

}

}

package Tutorial6;

public class ClassB extends ClassA{

public void b1()

{

System.out.println("B1 method");

}

public void b2()

{

System.out.println("B2 method");

}

public static void main(String[] args) {

}

}

package Tutorial6;

public class ClassC extends ClassB {

public void c1() {

System.out.println("C1 method");

}

public static void main(String[] args) {

ClassB obj6 = new ClassC();// we can access ClassA and ClassB

obj6.a1();

obj6.a2();

obj6.a3();

obj6.a4();

obj6.b1();

obj6.b2();

// ClassA obj5 =new ClassC(); //we can access ClassA

//

// obj5.a1();

// obj5.a2();

// obj5.a3();

// obj5.a4();

// ClassA obj4 =new ClassB();//we can access ClassA

// obj4.a1();

// obj4.a2();

// obj4.a3();

// obj4.a4();

// ClassC obj3 =new ClassC();//we can access ClassA, ClassB and ClassC

// obj3.a1();

// obj3.a2();

// obj3.a3();

// obj3.a4();

// obj3.b1();

// obj3.b2();

// obj3.c1();

// ClassB obj2 =new ClassB();//we can access ClassA and ClassB

// obj2.a1();

// obj2.a2();

// obj2.a3();

// obj2.a4();

// obj2.b1();

// obj2.b2();

// ClassA obj1 =new ClassA();//we can access 4 Methods from ClassA

// obj1.a1();

// obj1.a2();

// obj1.a3();

// obj1.a4();

}

}

**3) Hierarchical inheritance**

one parent class have multiple child class.

just imagine we have 3 classes, ClassA , ClassB and ClassC

ClassA extends in ClassB

and ClassA extends in ClassC.

ClassA

ClassB extends ClassA

ClassC extends ClassA

- just example in ClassA we have defines 10 methods, in ClassB we have defined 5 Methods and in ClassC we have defined 3 Methods

Scenario 1

ClassA obj =new ClassA(); //we can access all methods from ClassA

Scenario 2:

ClassB obj =new ClassB(); //we can access all methods from ClassA and ClassB

Scenario 3:

ClassC obj =new ClassC(); ////we can access all methods from ClassA and ClassC

scenario 4:

ClassA obj =new ClassB(); //we can access all methods from ClassA

Scenario 5;

ClassA obj =new ClassC(); //we can access all methods from ClassA

Scenario 6:

ClassB obj =new ClassC(); //invalid scenario

public class ClassA {

public void a1() {

System.out.println("A1 method");

}

public void a2() {

System.out.println("A2 method");

}

public void a3() {

System.out.println("A3 method");

}

public void a4() {

System.out.println("A4 method");

}

public static void main(String[] args) {

}

}

package Tutorial9;

public class ClassB extends ClassA {

public void b1() {

System.out.println("B1 method");

}

public void b2() {

System.out.println("B2 method");

}

}

package Tutorial9;

public class ClassC extends ClassA {

public void c1() {

System.out.println("C1 method");

}

public static void main(String[] args) {

ClassA obj5 = new ClassC();

obj5.a1();

obj5.a2();

obj5.a3();

obj5.a4();

// ClassA obj4 =new ClassB();

// obj4.a1();

// obj4.a2();

// obj4.a3();

// obj4.a4();

// ClassC obj3 = new ClassC();

// obj3.a1();

// obj3.a2();

// obj3.a3();

// obj3.a4();

// obj3.c1();

// ClassB obj2 =new ClassB();

// obj2.a1();

// obj2.a2();

// obj2.a3();

// obj2.a4();

// obj2.b1();

// obj2.b2();

// ClassA obj1 =new ClassA();

// obj1.a1();

// obj1.a2();

// obj1.a3();

// obj1.a4();

}

}

**What is inheritance in java? VVVVVIMMMMMMPPPPPPP**

- Inheritance is process of reusing the class/data members from one class to another class.

- Data members is nothing but variables and methods.

- we achieve the inheritance between multiple classes by using extends keywords.

- once we use extends keyword in child classes, then by using child class object we can access all the data members/properties from parent class as child class.

- the class where class members are getting inherited is called as parent class or super class or base class.,

- the class to which class members are getting inherited is called as child class or derived class or sub class.

there are 5 types of inheritance in java.

I) single level inheritance

2) multilevel inheritance

3) Hierarchical inheritance

4) multiple inheritance

5) hybrid inheritance

- in single level inheritance we extends one class in another class.

- in multilevel inheritance we extends one class in another class, and another class in another class and so on.

- in hierarchical inheritance we extends parent class in multiple child classes.

- in multiple inheritance we have extends 2 classes in one class, which is does not supports,

- in hybrid inheritance is combination of multilevel inheritance and multiple inheritance, and java does not support for hybrid inheritance

- in current framework we are using Hierarchical inheritance, in my framework BaseClass is super parent class of all the classes in frameworks.

- we extends BaseClass in PageLayer package container classes, as well as Utility Layer container classes, and also TestLayer package container classes.

- so here we have extends BaseClass in multiple classes.

--------

why java does not support for multiple inheritance or Hybrid inheritance? VVVVVIMMMMMMPPPPPPP

- The reason behind multiple inheritance to prevent ambiguity issue.

- consider the scenario, we have 3 classes ClassA, ClassB and ClassC.

- we have defined display() method in ClassA as well as in ClassB.

- and we have extends ClassA in ClassC as well as we extends ClassB in ClassC, means ClassC have 2 parent Class as ClassA and ClassB.

- if we create object of child class C then we can access all methods from parent class as well as from child class.

- but currently display() method present inside the ClassA and ClassB, so java compiler does not decided which display() method we have call.

-to prevent such condition multiple inheritance is not supported by java.

- still if u want achieve the multiple inheritance then we use interface concept.

**where you have used inheritance concept in framework? VVVVVIMMMMMMPPPPPPP**

-there are 5 types of inheritance in java.

I) single level inheritance

2) multilevel inheritance

3) Hierarchical inheritance

4) multiple inheritance

5) hybrid inheritance

-in my current[Hybrid Framework/ BBD Framework] framework we use Hierarchical inheritance, Hierarchical inheritance we have extends the Parent Class in multiple child classes.

- in my current framework we have created BaseClass and this BaseClass is super parent of all the classes in frameworks.

- we have extends the BaseClass in Page Layer package container classes, as well as we have extends BaseClass in Utility Layer containers classes, and also

we have extends the BaseClass in Test Layer (Step Definition) Package containers classes.

- here we have BaseClass is parent class of multiple child classes, and we use Hierarchical inheritance concept in framework.===================================================================================

Multiple Inheritance

package Tutorial11;

public class ClassA {

public void display() {

System.out.println("Display method in ClassA");

}

}

package Tutorial11;

public class ClassB {

public void display() {

System.out.println("Display method in ClassB");

}

}

package Tutorial11;

//Multiple Inheritance : it show compile time error/Syntax Error

public class ClassC extends ClassA extends ClassB

{

}

==========================================================================

**what are the different types of variables in java? VVVVVIMMMMMMPPPPPPP**

- there are 3 types of variables in java

I) local variable

2) static variable

3) instance variable

I) local variable

- if we define any variable within the block (Methods, constructor , else if block, loops) then access scope of these only within the block.

2) static variable

- static variable is also called as global static variables.

- we declare static variable outside the methods but inside the class by using static keywords/(Non Access Modifier keyword).

- we can access static variable within the class as well as outside the class by using the access modifier concept.

- if declare static variable,

we can access static variable by using 3 ways in blocks.

I) directly variable name

2) class name dot variable name

3) by using object name dot variable name.

3) instance variable

- instance variable is also called as global variable / global instance variable/non static variable.

- we declare the instance variable outside the methods but inside the class.

- We can access instance variable within the class as well as outside the class as per the access modifiers.

- we can access instance variable in non static method by using 2 ways.

I) directly variable name

2) by object name dot variable name.

- we can access instance variable in static method/main method by using object name dot variable name.

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

package Tutorial11;

public class Demo1 {

public void abc() {

int a = 100;

System.out.println(a);

}

public void xyz() {

int c = 120;

if (true) {

int b = 100;

System.out.println(b);

System.out.println(c);

} else {

System.out.println(c);

// System.out.println(b);

}

// System.out.println(b);

System.out.println(c);

}

public static void main(String[] args) {

int p = 100;

System.out.println(p);

for (int i = 0; i <= 10; i++) {

int s = 12;

System.out.println(s);

}

// System.out.println(s);

int r = 190;

for (int i = 1; i <= 5; i++) {

System.out.println(r);

}

System.out.println(r);

}

}

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

package Tutorial11;

public class Demo2 {

//static variable /static global variable

static int a=100;

public void abc()

{

System.out.println(a);

System.out.println(Demo2.a);

Demo2 obj =new Demo2();

System.out.println(obj.a);

}

public static void xyz()

{

System.out.println(a);

System.out.println(Demo2.a);

Demo2 obj =new Demo2();

System.out.println(obj.a);

}

public static void main(String[] args)

{

System.out.println(a);

System.out.println(Demo2.a);

Demo2 obj =new Demo2();

System.out.println(obj.a);

}

}

package Tutorial11;

public class Demo3 {

//instance variable /global variable / non static variable

int a=190;

public void abc()

{

System.out.println(a);

Demo3 obj =new Demo3();

System.out.println(obj.a);

}

public static void xyz()

{

Demo3 obj=new Demo3();

System.out.println(obj.a);

}

public static void main(String[] args)

{

Demo3 obj =new Demo3();

System.out.println(obj.a);

}

}

**what are the different access modifier in java? VVVVVIMMMMMMPPPPPPP**

there are 4 types of Access Modifier in java

1) private

2) default

3) protected

4) public

1) private

- if we declare any data members of the class as private then access scope of that data members within the class.

- we can't access outside the class.

2) default

- if we declare any data members of the class as default then access scope of that data members within the Package

- we can't access outside the package

- there is no keyword for default data members.

3) protected

- if we declare any data members of the class as protected then access scope of that data members within the Package as well as outside the package but

outside the package by using inheritance concept.s

4) public

- if we declare any data members of the class as public then access scope of that data members anywhere in project.

package Tutorial12;

public class Sample101 {

// private instance variable

private String a1 = "Private variable";

// default instance variable

String b1 = "Default Variable";

// protected instance variable

protected String c1 = "Protected variable";

// public instance variable

public String d1 = "Public Variable";

// private non static method

private void p1() {

System.out.println("Private Method");

}

// default non static method

void q1() {

System.out.println("Default Method");

}

// protected Non static method

protected void r1() {

System.out.println("Protected Method");

}

// public non static method

public void s1() {

System.out.println("Public Method");

}

public static void main(String[] args) {

Sample101 obj = new Sample101();

System.out.println(obj.a1);

System.out.println(obj.b1);

System.out.println(obj.c1);

System.out.println(obj.d1);

obj.p1();

obj.q1();

obj.r1();

obj.s1();

}

}

package Tutorial12;

public class Pune101 extends Sample101 {

public static void main(String[] args) {

Pune101 obj = new Pune101();

System.out.println(obj.b1);

System.out.println(obj.c1);

System.out.println(obj.d1);

obj.q1();

obj.r1();

obj.s1();

}

}

package Tutorial123;

import Tutorial12.Sample101;

public class Mumbai101 extends Sample101 {

public static void main(String[] args) {

Mumbai101 obj = new Mumbai101();

System.out.println(obj.c1);

System.out.println(obj.d1);

obj.r1();

obj.s1();

}

}

package Tutorial120;

public class Sample121 {

private static String a = "private";

static String b = "default";

protected static String c = "protected";

public static String d = "public";

public static void main(String[] args) {

System.out.println(a);

System.out.println(b);

System.out.println(c);

System.out.println(d);

}

}

package Tutorial120;

public class Pune121 extends Sample121 {

public static void main(String[] args) {

System.out.println(b);

System.out.println(c);

System.out.println(d);

}

}

package Tutorial130;

import Tutorial120.Sample121;

public class Mumbai121 extends Sample121{

public static void main(String[] args)

{

System.out.println(c);

System.out.println(d);

}

}

Binding

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

- connecting method call to method body is called as binding.

there are 2 types of binding VVVVVIMMMPPPPPPPPPPP

1) static binding

2) dynamic binding

1) static binding:

- Static binding is also called as early binding.

- Static binding will happens before actually program runs.

- static binding which can be resolved at the compile time by compiler.

- Method overloading is best example of the static binding.

- binding of all static method, private method and final method done at compile time by compiler.

2) dynamic binding:

- Dynamic binding is also called as late binding.

- Dynamic binding can be resolved at run time by JVM.

- Dynamic binding is happens after running the programs.

- Method overriding best example of dynamic binding.

=======================================================================================

package Tutorial1;

public class Demo1 {

// class level memory location: before running the program

public static void abc() {

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

}

// it store in heap memory location:

// heap memory will create after running the program means at run time

public void xyz() {

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

}

public static void main(String[] args)

{

Demo1.abc();

Demo1 obj1 =new Demo1();

obj1.xyz();

}

}

=======================================================================================

2) Polymorphism VVVVVVVVIMMMPPPPPPPPP

there are 2 main types of polymorphism

1) Method overloading

2) Method overriding

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

1) Method overloading VVVVVVVVIMMMPPPPPPPPP

- Method overloading is also called as compile time polymorphism, or early binding or static binding.

- if we have 2 or more method with same method name in same class but with different signature.

- signature means

I) same method name but number of arguments is different

2) same method name and also same number arguments but type of arguments is different.

3) same method name and also same number or arguments and also same type of arguments but sequence/order of arguments is different.

or

- signature means

I) number of arguments is different

2) type of arguments is different.

3) sequence/order of arguments is different.

- can we overload the main method?

yes

- can we overload the static method?

yes

- can we overload the non static method?

yes

- can we overload the private method?

yes

- can we overload the final method?

yes

- can we overload the default method?

yes

- can we overload the protected method?

yes

- can we overload the abstract method?

yes

- can we overload the constructor?

yes

==============================================================================

- can we overload the static?

yes

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

package Tutorial1;

public class Demo2 {

public static void abc() {

System.out.println("abc static Method with 0 args");

}

public static void abc(int a)

{

System.out.println("abc static Method with 1 int args "+a);

}

public static void abc(String a)

{

System.out.println("abc static Method with 1 String args "+a);

}

public static void abc(int a, String b)

{

System.out.println("abc static Method with 1 int and 1 String args "+a +""+b);

}

public static void abc(String a, int b) {

System.out.println("abc static Method with 1 String and 1 int args "+a +""+b);

}

public static void main(String[] args) {

Demo2.abc(10, "cucumber");

Demo2.abc("Selenium");

Demo2.abc();

Demo2.abc(1000);

Demo2.abc("TestNG", 9090);

}

}

=================================================================================

Can we achieve the method overloading for non static method?

yes

package Tutorial1;

public class Demo3 {

// can we overload the non static method

public void display()

{

System.out.println("Non Static Method with 0 args");

}

public void display(int a)

{

System.out.println("Non Static Method with 1 int args");

}

public void display(char a)

{

System.out.println("Non Static Method with 1 char args "+a);

}

public void display(int a, char b)

{

System.out.println("Non Static Method with 1 int and 1 char args "+(a+b));

}

public void display(char a, int b)

{

System.out.println("Non Static Method with 1 char and 1 int args "+(a+b));

}

public static void main(String[] args) {

Demo3 obj =new Demo3();

obj.display('a', 10);

obj.display();

obj.display(1000);

obj.display(100, 'P');

obj.display('S');

}

}

===============================================================================

can we achieve the method overloading for private static method

yes

package Tutorial1;

public class Demo4 {

// can we achieve the method overloading for private static method

private static void xyz() {

System.out.println("Private Static Method with 0 args");

}

private static void xyz(String a) {

System.out.println("Private Static Method with 1 String args " + a);

}

private static void xyz(char a) {

System.out.println("Private Static Method with 1 char args " + a);

}

private static void xyz(String a, char b) {

System.out.println("Private Static Method with 1 String and 1 char args " + a + " " + b);

}

private static void xyz(char a, String b) {

System.out.println("Private Static Method with 1 char and 1 String args " + a + " " + b);

}

public static void main(String[] args)

{

Demo4.xyz();

Demo4.xyz('A');

Demo4.xyz("Jenkins");

Demo4.xyz('A', "Rest Assured");

Demo4.xyz("Hamcrest Framework", 'A');

}

}

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

can we achieve the method overloading for private NON STATIC method

yes

package Tutorial1;

public class Demo5 {

// can we achieve the method overloading for private NON STATIC method

private void xyz() {

System.out.println("Private NON Static Method with 0 args");

}

private void xyz(String a) {

System.out.println("Private NON Static Method with 1 String args " + a);

}

private void xyz(char a) {

System.out.println("Private NON Static Method with 1 char args " + a);

}

private void xyz(String a, char b) {

System.out.println("Private NON Static Method with 1 String and 1 char args " + a + " " + b);

}

private void xyz(char a, String b) {

System.out.println("Private NON Static Method with 1 char and 1 String args " + a + " " + b);

}

public static void main(String[] args) {

Demo5 obj = new Demo5();

obj.xyz();

obj.xyz('P');

obj.xyz("Hamcrest Framework");

obj.xyz('M', "JsonPath");

obj.xyz("JDBC", 'P');

}

}

===========================================================================

can we achieve the method overloading for final STATIC method

Yes

can we achieve the method overloading for final Non STATIC method

Yes

package Tutorial1;

public class Demo6 {

// can we achieve the method overloading for final STATIC method

final public static void display() {

System.out.println("Final Static Method with 0 args");

}

final public static void display(int a) {

System.out.println("Final Static Method with 1 int args " + a);

}

final public static void display(String a) {

System.out.println("Final Static Method with 1 String args " + a);

}

final public static void display(int a, String b) {

System.out.println("Final Static Method with 1 int and 1 String args " + a + b);

}

final public static void display(String a, int b) {

System.out.println("Final Static Method with 1 String and 1 int args " + a + b);

}

public static void main(String[] args) {

Demo6.display();

Demo6.display(100);

Demo6.display("GitHub");

Demo6.display(100, "Maven Build tool");

Demo6.display("POM-Design Pattern to design the OR", 9090);

}

}

package Tutorial1;

public class Demo7 {

// can we achieve the method overloading for final NON STATIC method

final public void display() {

System.out.println("Final NON Static Method with 0 args");

}

final public void display(int a) {

System.out.println("Final NON Static Method with 1 int args " + a);

}

final public void display(String a) {

System.out.println("Final NON Static Method with 1 String args " + a);

}

final public void display(int a, String b) {

System.out.println("Final NON Static Method with 1 int and 1 String args " + a + b);

}

final public void display(String a, int b) {

System.out.println("Final NON Static Method with 1 String and 1 int args " + a + b);

}

public static void main(String[] args) {

Demo7 obj = new Demo7();

obj.display();

obj.display(100);

obj.display("POM");

obj.display(100, "Version Control System tool");

obj.display("GitHub", 90900);

}

}

===========================================================================

Can we achieve method overloading for protected static method

yes

Can we achieve method overloading for protected non static method

yes

package Tutorial1;

public class Demo8 {

// can we achieve the method overloading for protected STATIC method

protected static void display() {

System.out.println("protected Static Method with 0 args");

}

protected static void display(int a) {

System.out.println("protected Static Method with 1 int args " + a);

}

protected static void display(String a) {

System.out.println("protected Static Method with 1 String args " + a);

}

protected static void display(int a, String b) {

System.out.println("protected Static Method with 1 int and 1 String args " + a + b);

}

protected static void display(String a, int b) {

System.out.println("protected Static Method with 1 String and 1 int args " + a + b);

}

public static void main(String[] args) {

Demo8.display();

Demo8.display(100);

Demo8.display("JDBC");

Demo8.display(100, "BDD Framework");

Demo8.display("Extent Reports", 100);

}

}

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

package Tutorial1;

public class Demo9 {

// can we achieve the method overloading for protected Non STATIC method

protected void display() {

System.out.println("protected Non Static Method with 0 args");

}

protected void display(int a) {

System.out.println("protected Non Static Method with 1 int args " + a);

}

protected void display(String a) {

System.out.println("protected Non Static Method with 1 String args " + a);

}

protected void display(int a, String b) {

System.out.println("protected Non Static Method with 1 int and 1 String args " + a + b);

}

protected void display(String a, int b) {

System.out.println("protected Non Static Method with 1 String and 1 int args " + a + b);

}

public static void main(String[] args) {

Demo9 obj = new Demo9();

obj.display();

obj.display(100);

obj.display("Version Control Tool");

obj.display(100, "Build Tool");

obj.display("REST API", 90909);

}

}

===========================================================================

can we achieve the method overloading for static main method

yes

can we achieve the method overloading for Non static main method

yes

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

package Tutorial1;

public class Demo10 {

// can we achieve the method overloading for main static method

public static void main() {

System.out.println("main method with 0 args");

}

public static void main(int a) {

System.out.println("main method with 1 int args " + a);

}

public static void main(String a) {

System.out.println("main method with 1 String args " + a);

}

public static void main(String a, int b) {

System.out.println("main method with 1 String and 1 int args " + a + b);

}

public static void main(int a, String b) {

System.out.println("main method with 1 int and 1 String args " + a + b);

}

public static void main(String[] args)

{

Demo10.main();

Demo10.main(100);

Demo10.main("GitHub");

Demo10.main(100, "Maven");

Demo10.main("Rest",100);

}

}

package Tutorial1;

public class Demo11 {

// can we achieve the method overloading for main Non static method

public void main() {

System.out.println("main Non Static method with 0 args");

}

public void main(int a) {

System.out.println("main Non Static method with 1 int args " + a);

}

public void main(String a) {

System.out.println("main Non Static method with 1 String args " + a);

}

public void main(String a, int b) {

System.out.println("main Non Static method with 1 String and 1 int args " + a + b);

}

public void main(int a, String b) {

System.out.println("main Non Static method with 1 int and 1 String args " + a + b);

}

public static void main(String[] args) {

Demo11 obj = new Demo11();

obj.main();

obj.main(100);

obj.main("Maven");

obj.main(100, "Rest");

obj.main("Assert", 1000);

}

}

===========================================================================

Can we achieve the method overloading for constructor

yes

package Tutorial1;

public class Demo12 {

// can we achieve the method overloading for constructor

public Demo12() {

System.out.println("Constructor 0 args");

}

public Demo12(int a) {

System.out.println("Constructor 1 int args " + a);

}

public Demo12(String a) {

System.out.println("Constructor 1 String args " + a);

}

public Demo12(int a, String b) {

System.out.println("Constructor 1 int and 1 String args " + a + b);

}

public Demo12(String a, int b) {

System.out.println("Constructor 1 String and 1 int args " + a + b);

}

public static void main(String[] args) {

Demo12 obj1 = new Demo12();

Demo12 obj2 =new Demo12(100);

Demo12 obj3= new Demo12("Rest API");

new Demo12(100,"Maven");

new Demo12("build tool",9010);

}

}

=====================================================================

can we achieve the method for abstract method

yes

package Tutorial1;

public abstract class Demo13 {

//can we achieve the method overloading for abstract method

public abstract void abc();

public abstract void abc(int a);

public abstract void abc(String a);

public abstract void abc(int a, String b);

public abstract void abc(String a, int b);

}

package Tutorial1;

public class Sample13 extends Demo13 {

@Override

public void abc() {

System.out.println("abc method with 0 args");

}

@Override

public void abc(int a) {

System.out.println("abc method with 1 int args " + a);

}

@Override

public void abc(String a) {

System.out.println("abc method with 1 String args " + a);

}

@Override

public void abc(int a, String b) {

System.out.println("abc method with 1 int and 1 String args " + a + b);

}

@Override

public void abc(String a, int b) {

System.out.println("abc method with 1 String and 1 int args " + a + b);

}

public static void main(String[] args) {

Demo13 obj = new Sample13();

obj.abc();

obj.abc(100);

obj.abc("TestNG");

obj.abc(100, "Cucumber");

obj.abc("BDD Framework", 1000);

}

}

===============================================================

can we override the main method

No

can we override the static method

No

can we override the private method

No

can we override the final method

No

can we override the constructor

No

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

can we override the abstract method

yes

can we override the non static method

yes

Polymorphism

there are 2 types of Polymorphism

1) Method overloading

2) Method Overriding

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

2) Method Overriding VVVVIMMMPPPPPPPPPPP

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

- Method Overriding is also called as run time polymorphism or dynamic binding or late binding.

- if parent class and child class have same method with same signature and with same return type is called method overriding.

- same signature means

1) numbers of arguments is same for method in child and parent class.

2) type of arguments is same for method in child class and parent clas

3) order or sequence of arguments is same for method in child class and parent class.

- we achieve the method overriding concept by using inheritance concept.

- to achieve the method overriding 2 or more classes.

- we can achieve the method overriding for

1) public non static method

2) protected non static method

3) default non static method,

4) abstract method

- we can not achieve the method overriding for

1) private method

2) static method

3) main method

4) final method

5) constructor

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

1) public non static method

package Tutorial2;

public class Demo1 {

public void abc() {

System.out.println("abc method with 0 args");

}

public void xyz(int a, int b) {

System.out.println("xyz method with 2 int args " + (a + b));

}

}

package Tutorial2;

public class Sample1 extends Demo1 {

public void abc() {

System.out.println("abc method in Sample1 class");

}

public void xyz(int p, int s) {

System.out.println("xyz method in sample1 classs");

}

public static void main(String[] args) {

Sample1 obj =new Sample1();

obj.abc();

obj.xyz(10, 100);

}

}

=========================================================================

package Tutorial2;

public class Demo2 {

public void d1() {

System.out.println("D1 method in Demo2 class");

}

public int d2() {

System.out.println("D2 method in Demo2 class");

return 100;

}

public void d3(String a, String b) {

System.out.println("D3 method in Demo2 class");

}

public String d4(int a, int b) {

System.out.println("D3 method in Demo2 class");

return "selenium";

}

}

package Tutorial2;

public class Sample2 extends Demo2{

public void d1()

{

System.out.println("D1 method in Sample2 class");

}

public int d2()

{

System.out.println("D2 method in Sample2 class");

return 1000;

}

public void d3(int a, int b, String c)

{

System.out.println("D3 method in Sample2 class");

}

public void d4()

{

System.out.println("D4 method in Sample2 class");

}

public static void main(String[] args) {

Sample2 obj =new Sample2();

obj.d1();

obj.d2();

obj.d3(10, 10, "abc");

obj.d4(100, 100);

}

}

=============================================================================

2) protected non static method

package Tutorial2;

public class Demo3 {

protected void abc(int a)

{

System.out.println("abc method in demo3 class");

}

protected int pqr()

{

System.out.println("PQR method in demo3 class");

return 1000;

}

protected void xyz(int a,String b)

{

System.out.println("XYZ method in demo3 class");

}

}

package Tutorial2;

public class Sample3 extends Demo3 {

public void abc(int a) {

System.out.println("abc method in Sample3 class");

}

public int pqr() {

System.out.println("PQR method in Sample3 class");

return 19090;

}

public void xyz(int b,String p) {

System.out.println("XYZ method in Sample3 class");

}

public static void main(String[] args) {

Sample3 obj =new Sample3();

obj.abc(100);

obj.pqr();

obj.xyz(100, "selenium");

}

}

=================================================================

3) default non static method

package Tutorial2;

public class Demo4 {

void abc(String a) {

System.out.println("abc method in Demo4 class");

}

String pqr(int a, int b)

{

System.out.println("PQR method in Demo4 class");

return "Selenium";

}

void xyz(float a)

{

System.out.println("XYZ method in Demo4 class");

}

}

package Tutorial2;

public class Sample4 extends Demo4 {

protected void abc(String p)

{

System.out.println("Abc method in Sample4 class");

}

String pqr(int a, int b)

{

System.out.println("PQR method in Sample4 class");

return "selenium";

}

void xyz(float a)

{

System.out.println("XYZ method in Sample4 class");

}

public static void main(String[] args) {

Sample4 obj =new Sample4();

obj.abc("selenium");

obj.pqr(10, 100);

obj.xyz(1900.34f);

}

}

================================================================================

4) abstract method

package Tutorial2;

public abstract class Demo5 {

public abstract void abc();

public abstract void pqr(String a);

public abstract int xyz();

}

package Tutorial2;

public class Sample5 extends Demo5 {

@Override

public void abc() {

System.out.println("abc method in Sample5");

}

@Override

public void pqr(String a) {

System.out.println("pqr method in Sample5");

}

@Override

public int xyz() {

System.out.println("XYZ method in Sample5");

return 100;

}

public static void main(String[] args) {

Sample5 obj =new Sample5();

obj.abc();

obj.pqr("selenium");

obj.xyz();

}

}

===================================================================================

can we override the static method in java? VVVVVVIMMMPPPPPPPPPPPPPPPPPPP

No,

why we can not override the static method in java? VVVVVVIMMMPPPPPPPPPPPPPPPPPPP

- we cannot override the static method in java because method overriding is based on dynamic binding at run time and

static method are bonded using static binding at compile time by compiler, that is reason we can't override the static method in java

- if we declare static method in parent class and child class with same signature then it is called method hiding concept.

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

can we override the main method in java?

- No, because method it self as static method

why we can not override the main method in java? VVVVVVIMMMPPPPPPPPPPPPPPPPPPP

- we cannot override the static method in java because method overriding is based on dynamic binding at run time and

static method are bonded using static binding at compile time by compiler, that is reason we can't override the static method in java

- if we declare static method in parent class and child class with same signature then it is called method hiding concept.

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

package Tutorial2;

public class Demo7 {

public static void abc()

{

System.out.println("abc method in Demo7 class");

}

public static void xyz(String a, int b)

{

System.out.println("xyz method in Demo7 class");

}

}

package Tutorial2;

public class Sample7 extends Demo7 {

public static void abc()

{

System.out.println("abc method in Sample7 class");

}

public static void xyz(String a, int b)

{

System.out.println("xyz method in Sample7 class");

}

public static void main(String[] args) {

Sample7 obj =new Sample7();

obj.abc();

obj.xyz("cucumber", 1000);

}

}

===========================================================================================

can we override the final method? VVVVVVIMMMPPPPP

No,

why we cannot override the final method VVVVVVIMMMPPPPP

final cannot method be override because final keyword is used for the in future we cannot change the method logic or we cannot the override the methods.

- once we declare any method as final, we cannot modify the method logic.

package Tutorial2;

public class Sample8 extends Demo8{

//compile time error or syntax error

final void abc()

{

System.out.println("abc method in Demo8 class");

}

}

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

can we override the private method? VVVVVVIMMMPPPPPPPPP

No,

why we can not override the private method? VVVVVVIMMMPPPPPPPPP

if we declare any method as private then access scope of this method is within the class, so we cannot override it.

because to achieve the overriding we need at least 2 or more classes

package Tutorial2;

public class Demo8 {

private void abc()

{

System.out.println("abc method in Demo8 class");

}

}

package Tutorial2;

public class Sample8 extends Demo8 {

private void abc() {

System.out.println("abc method in Sample8 class");

}

public static void main(String[] args) {

Sample8 obj =new Sample8();

obj.abc();

}

}

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

can we override the constructor in java? VVVVVIMMMPPPPPPPPPPPPPPPp

No

why we cannot override the constructor? VVVVVIMMMPPPPPPPPPPPPPPPp

constructor cannot be override because constructor name should be same as class name.

and to achieve the method overriding we need 2 or classes. inside the java we cannot create 2 classes with same name.

package Tutorial2;

public class Demo9 {

public Demo9()

{

System.out.println("Demo9 0 args cons");

}

}

==========================================================================================

There we different method overriding

1) findElement() ---> this is present inside the WebDriver interface as well as WebElement interface

2) findElements() ---> this is present inside the WebDriver interface as well as WebElement interface

3) getScreenshotAs() ---> this is present inside the TakesScreenshot interface as well as WebElement interface

4) getText() ---> this is present inside the Alert interface as well as WebElement interface

5) sendKeys() ---> this is present inside the Alert interface as well as WebElement interface

==========================================================================================

diff between Method overriding and method hiding?

method hiding

- if we have same static method present inside the parent class as well as child class with same signature is called method hiding

- Method hiding is also called as static binding or compile time polymorphism or early binding

- Method resolution take care by compiler for static methods

Method overriding

-if we have same non static method present inside the parent class and child class with same signature is called method overrding

- Method overriding is also called as dynamic binding or run time polymorphism or late binding

- Method resolution take by the JVM at run time

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

Where we are using overloading in framework? VVVVVVVVIMMMPPPPPP

- we use overloading concept in Page Layer package in src main java.

- in page layer package we create object repository as well as we write associated method for object repository.

- just take an example, we have to create new employee by entering firstname, middle name and last name, but middle name is optional.

in this case we create addEmployee() method with 2 String arguments as well as addEmployee() method with 3 string arguments

- in Utility Layer package we create reusable script code with same method name with different signature.

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

Where we are using Overriding in framework? VVVVVVVVIMMMPPPPPP

- We use overriding concept in Page Layer package in src/main/java.

- in page layer package we create object repository as well as we write associated method for object repository.

- just example in parent class we have created associated method to achieve the functionality and we have inherited same method in child classes to remodify the logic for methods.

- in Utility Layer package we create reusable script code with same method name with same signature in child class and parent class.

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

VVVVIMMMPPPPPPPPPPPPPPPPPPPPPPPPPPPP

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

1) What is Polymorphism?

2) diff between method overloading and method overriding

3) diff between static binding and dynamic binding

4) can we override the private method

5) why we can't override the private method

6) can we override the static method

7) why we can't override the static method

8) can we override the main method

9) why we can't override the main method

10) can we override the final method

11) why we can't override the final method

12) can we override the constructor

13) why we can't override the constructor

14) diff method hiding and method overriding

15) where you have used method overloading concept in framework

16) where you have used method overriding concept in framework

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

1) what is inheritance

2) where you have used inheritance concept in framework

3) why we can't achieve the multiple inheritance

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

1) what are the different variables in java

2) what are the different types of access modifier in java and their access scope

1) diff between == operator and equals() method

OOPs concept

1) Inheritance

2) Polymorphism

3) Abstraction

4) Encapsulation

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

3) Abstraction VVVVVVVVIMMPPPPPPPPPP

- Abstraction is process of hiding the implementation details and show the functionality to the users.

- there are 2 ways we can achieve the Abstraction

1) by declaring the Abstract Class

2) by declaring the Interface

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

1) by declaring the Abstract Class VVVVVVVVIMMPPPPPPPPPP

- Abstraction is process of hiding the implementation details and showing only functionality to users.

- by declaring the abstract class we can achieve the 0 to 100% abstractions.

- If we know the method name but dont know how to implement it then we use abstract class.

- if java class contains if we have one or more abstract method then we mark class as abstract class.

- in abstract class, we can declare abstract method (incomplete method) as well as concrete method (complete method).

- we achieve the abstraction only for abstract non static method.

- in abstract class,

1) we can declare the private method in abstract class

2) we can declare the static method in abstract class

3) we can declare the main method in abstract class

4) we can declare the final method in abstract class

5) we can declare the constructor in abstract class

6) we can declare the non static method in abstract class.

- in abstract class,

I) we can't declare the abstract private method

2) we cant declare the abstract static method

3) we can't declare the abstract main method

4) we can't declare the abstract final method

5) we can't declare the abstract constructor.

- to achieve abstraction in abstract class, we use inheritance concept.

- we implements all methods in another class with the help of extends keyword.

- we can't create object of Abstract class, but we can mention reference of Abstract class.

- If java class contains if we have one more abstract method then that class is called as abstract class.

Syntax: declare the abstract method

1) Abstract method without return type and 0 arguments

AccessModifier AbstractKeyword nonReturntype methodName();

1) Abstract method without return type and multiple arguments

AccessModifier AbstractKeyword nonReturntype methodName(Datatype variableName, datatype variableName);

3) Abstract method with return type and 0 arguments

AccessModifier AbstractKeyword Returntype methodName();

4) Abstract method with return type and multiple arguments

AccessModifier AbstractKeyword Returntype methodName(Datatype variableName, datatype variableName);

what are the different access modifier in java

there are 4 types of access modifier in java

1) private

2) public

3) default

4) protected

what is non return type keyword?

void

what is return type keyword

byte, short, int, long, float, double, char, boolean, String, array, and own class.

what is method name?

any name

Datatype; byte, short, int, long, float, double, char, boolean, String, array, and own class.

variable name: any name

where we declare the abstract method?

before the main method or after the main method but inside the class.

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

package Tutorial3;

public abstract class Demo1 {

//abstract method without return type and 0 args

public abstract void abc();

//abstract method without return type and 2 int args

public abstract void pqr(int a,int b);

//abstract method with String return type and 0 args

public abstract String reverseString();

//abstract method with int return type and 2 int arguments

public abstract int addition(int a,int b);

//Non static method/ complete method/ concrete method

public void display()

{

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

}

}

package Tutorial3;

public class Sample1 extends Demo1 {

@Override

public void abc() {

System.out.println("ABC method in Sample1 class");

}

@Override

public void pqr(int a, int b) {

System.out.println("PQR method in Sample1 class " + (a + b));

}

@Override

public String reverseString() {

String a = "selenium";

String rev = "";

for (int i = a.length() - 1; i >= 0; i--) {

rev = rev + a.charAt(i);

}

return rev;

}

@Override

public int addition(int a, int b) {

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

return a + b;

}

public static void main(String[] args) {

Demo1 obj = new Sample1();

obj.abc();

obj.addition(10, 100);

obj.reverseString();

obj.pqr(100, 100);

obj.display();

// Sample1 obj =new Sample1();

// obj.abc();

// obj.pqr(10, 10);

// obj.reverseString();

// obj.addition(100,10);

// obj.display();

}

}

===========================================================================

package Tutorial3;

public abstract class Demo2 {

//Abstract method or incomplete complete

public abstract void abc();

//can we declare private in abstract class

private void a1()

{

System.out.println("A1 Private Method");

}

//can we declare static in abstract class

public static void a2()

{

System.out.println("A2 static method");

}

//can we declare final method in abstract class

final public void a3()

{

System.out.println("A3 final method");

}

//can we declare constructor in abstract class

public Demo2()

{

System.out.println("Demo2 constructor");

}

//can we declare main method in abstract clss

public static void main(String[] args) {

Demo2 obj =new Sample2();

obj.a1();

obj.a2();

obj.a3();

obj.abc();

}

}

package Tutorial3;

public class Sample2 extends Demo2{

@Override

public void abc() {

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

}

public static void main(String[] args) {

Sample2 obj =new Sample2();

obj.abc();

obj.a2();

obj.a3();

}

}

=======================================================================

package Tutorial3;

public abstract class Demo3 {

public abstract void d1();

//can we declare abstract private method in abstract class : NO

private abstract void d2();

//can we declare abstract static method in abstract class : No

public static abstract void d3();

//can we declare abstract final method in abstract clas : No

final abstract void d4();

//can we declare abstract main method in abstract class : No

public abstract static void main(String [] args);

//can we declare the abstract constructor in abstract class : No

public abstract Demo3();

}

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

can we declare abstract private method in abstract class VVVVVVIMMPPPPPPP

NO

why we can't declare abstract private method in abstract class. VVVVVVIMMPPPPPPP

to achieve the abstraction compulsory we have to override this method in child class,

but access scope of private method is within the class.

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

can we declare abstract static method in abstract class VVVVVVIMMPPPPPPP

No

why we can't declare abstract static method in abstract class. VVVVVVIMMPPPPPPP

to achieve the abstraction compulsory we have to override this method in child class,

but static method are bonded on static binding at compile time by compiler and we achieve overriding or abstraction we have to use run time

polymorphism concept.

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

can we declare abstract final method in abstract class VVVVVVIMMPPPPPPP

No

why we can't declare abstract final method in abstract class. VVVVVVIMMPPPPPPP

to achieve the abstraction compulsory we have to override this method in child class,

but once declare any method as final, in future we can't modify the method logic.

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

can we declare abstract main method in abstract class VVVVVVIMMPPPPPPP

No

why we can't declare abstract main method in abstract class. VVVVVVIMMPPPPPPP

because main method is static method

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

can we declare the abstract constructor in abstract class VVVVVVIMMPPPPPPP

No

why we can't declare abstract constructor in abstract class. VVVVVVIMMPPPPPPP

to achieve the abstraction compulsory we have to override constructor in child class,

but constructor name should be same as class name, in java we can not create 2 classes with same name.

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

can we declare the private method in abstract class? VVVVVVIMMPPPPPPP

yes

can we declare static method in abstract class? VVVVVVIMMPPPPPPP

yes

can we declare final method in abstract class? VVVVVVIMMPPPPPPP

yes

can we declare main method in abstract class? VVVVVVIMMPPPPPPP

yes

can we declare constructor in abstract class VVVVVVIMMPPPPPPP

yes

can we achieve overloading for abstract method VVVVVVIMMPPPPPPP

yes

can we achieve overriding for abstract method VVVVVVIMMPPPPPPP

yes

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

package Tutorial3;

public abstract class Demo2 {

//Abstract method or incomplete complete

public abstract void abc();

//can we declare private in abstract class

private void a1()

{

System.out.println("A1 Private Method");

}

//can we declare static in abstract class

public static void a2()

{

System.out.println("A2 static method");

}

//can we declare final method in abstract class

final public void a3()

{

System.out.println("A3 final method");

}

//can we declare constructor in abstract class

public Demo2()

{

System.out.println("Demo2 constructor");

}

//can we declare main method in abstract clss

public static void main(String[] args) {

Demo2 obj =new Sample2();

obj.a1();

obj.a2();

obj.a3();

obj.abc();

}

}

======================================================================================

package Tutorial3;

public abstract class Demo4 {

//can we create all complete method in abstract class : yes

public void a1() {

System.out.println("A1 method");

}

public void a2() {

System.out.println("A2 method");

}

public void a3() {

System.out.println("A3 method");

}

public static void main(String[] args) {

}

}

package Tutorial3;

public class Sample4 extends Demo4{

public static void main(String[] args) {

Demo4 obj =new Sample4();

obj.a1();

obj.a2();

obj.a3();

}

}

================================================================

package Tutorial3;

public abstract class Demo5 {

public abstract void d1();

public abstract void d2();

public abstract void d3();

public abstract void d4();

}

package Tutorial3;

public abstract class Sample5 extends Demo5{

public void d1()

{

System.out.println("D1 method ");

}

public void d2()

{

System.out.println("D2 method ");

}

}

package Tutorial3;

public class Pune5 extends Sample5{

public void d3()

{

System.out.println("D3 method");

}

public void d4()

{

System.out.println("D4 method");

}

public static void main(String[] args) {

Sample5 obj=new Pune5();

obj.d1();

obj.d2();

obj.d3();

obj.d4();

}

}

========================================================================

package Tutorial3;

public abstract class Demo7 {

public abstract void d1();

public abstract void d2();

public abstract void d3();

}

package Tutorial3;

public abstract class Sample7 extends Demo7 {

public void d1() {

System.out.println("D1 method");

}

public abstract void s1();

public abstract void s2();

}

package Tutorial3;

public class Pune7 extends Sample7 {

public void d2() {

System.out.println("D2 Method");

}

public void d3() {

System.out.println("D3 Method");

}

public void s1() {

System.out.println("s1 method");

}

public void s2() {

System.out.println("s2 method");

}

public static void main(String[] args) {

Pune7 obj =new Pune7();

obj.d1();

obj.d2();

obj.d3();

obj.s1();

obj.s2();

// Sample7 obj =new Pune7();

// obj.d1();

// obj.d2();

// obj.d3();

// obj.s1();

// obj.s2();

// Demo7 obj =new Pune7();

// obj.d1();

// obj.d2();

// obj.d3();

}

}

=========================================================================================

package Tutorial3;

public abstract class Demo7 {

public abstract void d1();

public abstract void d2();

public abstract void d3();

}

package Tutorial3;

public abstract class Sample7 extends Demo7 {

public void d1() {

System.out.println("D1 method");

}

public abstract void s1();

public abstract void s2();

}

package Tutorial3;

public abstract class Pune7 extends Sample7 {

public void d2() {

System.out.println("D2 Method");

}

public void s1() {

System.out.println("s1 method");

}

public abstract void p1();

}

package Tutorial3;

public abstract class Mumbai7 extends Pune7 {

public void d3() {

System.out.println("D3 method");

}

public void s2() {

System.out.println("S2 method");

}

public abstract void m1();

}

package Tutorial3;

public class Delhi7 extends Mumbai7 {

@Override

public void m1() {

System.out.println("M1 method");

}

@Override

public void p1() {

System.out.println("P1 method");

}

public static void main(String[] args) {

Delhi7 obj = new Delhi7();

obj.d1();

obj.d2();

obj.d3();

obj.s1();

obj.s2();

obj.p1();

obj.m1();

// Mumbai7 obj =new Delhi7();

// obj.d1();

// obj.d2();

// obj.d3();

// obj.s1();

// obj.s2();

// obj.p1();

// obj.m1();

// Pune7 obj =new Delhi7();

// obj.d1();

// obj.d2();

// obj.d3();

// obj.s1();

// obj.s2();

// obj.p1();

// Sample7 obj =new Delhi7();

// obj.d1();

// obj.d2();

// obj.d3();

// obj.s1();

// obj.s2();

// Demo7 obj =new Delhi7();

//

// obj.d1();

// obj.d2();

// obj.d3();

}

}

Abstraction in java

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

- Abstraction is process of hiding the implementation details and only show functionality to the user.

there are 2 ways we can achieve the abstraction in java

1) by declaring abstract class

2) by declaring the interface.

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

2) by declaring the interface. VVVVVVIMMMPPPPPPPP

- Abstraction is process of hiding the implementation details and only show functionality to the user.

- interface is type of definition block which contains 100% are abstract method as well as we can define the static method, private method and default non static method.

- We implements interface in another class by using "implements" keywords.

- all interface variables are final variable.

- static and final modifier are not allowed in interface for variables

-by default all interface variable are static and final.

- we can't create object of interface but we can mention reference of interface.

- to create the interface we have to use interface.

- by using interface we can achieve the multiple inheritance by extending the 2 interface in one class.

- In interface we create

i) abstract method (incomplete method).

ii) we can create static method(complete method)

iii) we can create private method (complete method)

iv) default non static method (complete method)

- in interface

I) we can't create abstract static method

ii) we can't create abstract private method

iii) we can't create abstract final method

iv) we can't create abstract constructor

v) we can't create abstract main method

vi) we can't constructor in interface

vii) we can't create final method

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

package Tutorial4;

public interface Demo1 {

public abstract void d1();

public void d2();

}

package Tutorial4;

public class Sample1 implements Demo1{

@Override

public void d1() {

System.out.println("D1 Method");

}

@Override

public void d2() {

System.out.println("D2 Method");

}

public static void main(String[] args) {

Demo1 obj =new Sample1();

obj.d1();

obj.d2();

}

}

===================================================================================

package Tutorial4;

public interface Demo2 {

// abstract method without return type and 0 args

public void d1();

// abstract method without return type and 2 int args

public void d2(int a, int b);

// abstract method with String return type and 0 args

public abstract String d3();

// abstract method with int return type and 2 int args

public int d4(int a, int b);

public static void main(String[] args) {

Sample2 obj = new Sample2();

obj.d1();

obj.d2(100, 100);

obj.d3();

obj.d4(100, 200);

}

}

package Tutorial4;

public class Sample2 implements Demo2 {

@Override

public void d1() {

System.out.println("D1 method");

}

@Override

public void d2(int a, int b) {

System.out.println("D2 method");

}

@Override

public String d3() {

System.out.println("D3 method");

return "selenium";

}

@Override

public int d4(int a, int b) {

System.out.println("D4 method");

return 10;

}

public static void main(String[] args) {

Sample2 obj =new Sample2();

obj.d1();

obj.d2(100, 100);

obj.d3();

obj.d4(100, 200);

}

}

===============================================================================

package Tutorial4;

public interface Demo3 {

public void d1();

// can we create private method in interface: yes

private void d2() {

System.out.println("D2 private method");

}

// can we create static method interface : yes

public static void d3() {

System.out.println("D3 Static method");

}

// can we create non static default method : yes

default void d5() {

}

}

==============================================================================

package Tutorial4;

public interface Demo3 {

public void d1();

// can we create private method in interface: yes

private void d2() {

System.out.println("D2 private method");

}

// can we create static method interface : yes

public static void d3() {

System.out.println("D3 Static method");

}

// can we create non static default method : yes

default void d4() {

System.out.println("Default Method ");

}

//can we create main method in interface: yes

public static void main(String[] args) {

Demo3 obj =new Sample3();

obj.d1();

obj.d2();

Demo3.d3();

obj.d4();

}

}

package Tutorial4;

public class Sample3 implements Demo3 {

@Override

public void d1() {

System.out.println("D1 method");

}

public static void main(String[] args) {

Sample3 obj = new Sample3();

obj.d1();

obj.d4();

// if u want to call static method from interface then always we have to use

// reference of Interface

Demo3.d3();

}

}

==========================================================================

package Tutorial4;

public interface Demo4 {

public abstract void d1();

// can we create abstract static method : No

public abstract static void d2();

// can we create abstract private method : No

private abstract void d3();

// can we create abstract final method : No

final abstract void d4();

// can we create abstract main method : No

public abstract static void main(String [] args);

// can we create abstract constructor : No

public abstract Demo5();

// can we create final method in interface : No

final void d6()

{

}

// can we create constructor in interface : No

public Demo4()

{

}

}

=====================================================================

can we declare the abstract method in interface without using abstract keyword VVVVVVIMMPPPPP

yes

can we declare private method in interface VVVVVVIMMPPPPP

yes

can we declare static method in interface VVVVVVIMMPPPPP

yes

can we declare the default non static in interface VVVVVVIMMPPPPP

yes

can we remodify the value for interface variables VVVVVVIMMPPPPP

No

why we can't modify the value for interface variables? VVVVVVIMMPPPPP

because once we declare any variable in interface is called final variables.

=====================================================================

can we create abstract static method in interface? VVVVVVIMMPPPPP

No

why we can't create abstract static method in interface? VVVVVVIMMPPPPP

- to achieve the abstraction compulsory we have to use overriding concept.

- and static are bonded by static binding at compile time by compiler and method overriding is support for run time binding.

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

can we create abstract private method in interface? VVVVVVIMMPPPPP

No

why we can't create abstract private method in interface? VVVVVVIMMPPPPP

- to achieve the abstraction compulsory we have to use overriding concept.

- because access scope of private method is within the class and to achieve the abstraction we need at least 2 or mor classes

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

can we create abstract final method in interface? VVVVVVIMMPPPPP

No

why we can't create abstract final method in interface? VVVVVVIMMPPPPP

- to achieve the abstraction compulsory we have to use overriding concept.

- because once we declare any method as final we can't modify the logic for final method in another classes.

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

can we create abstract main method in interface? VVVVVVIMMPPPPP

No

why we can't create abstract main method in interface? VVVVVVIMMPPPPP

- because main method is itself is static method

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

can we create abstract constructor in interface? VVVVVVIMMPPPPP

No

why we can't create abstract constructor in interface? VVVVVVIMMPPPPP

- to achieve the abstraction compulsory we have to use overriding concept.

because constructor name should be same like class name and to achieve the abstraction we need 2 or more classes. and in java

we can't declare 2 classes with same name.

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

can we declare final method in interface VVVVVVIMMPPPPP

No

why we can't declare final method in interface? VVVVVVIMMPPPPP

- final method can't modify or we can't change the logic.

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

can we declare constructor in interface VVVVVVIMMPPPPP

No.

why we can't declare constructor in interface? VVVVVVIMMPPPPP

- because we can't create object of interface.

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

can we implements abstract class in interface? VVVVVVIMMPPPPP

NO

why we can't implements abstract class in interface? VVVVVVIMMPPPPP

because in abstract class we can mention the constructor , final method or non static method but in interface constructor is not allowed, as well final method is not allowed as well as non static is not allowed

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

can we implements interface in abstract class? VVVVVVIMMPPPPP

yes by using implements keyword, we implements interface in abstract class using

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

can we define abstract protected method interface

no

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

package Tutorial4;

public interface Demo4 {

public abstract void d1();

// can we create abstract static method : No

public abstract static void d2();

// can we create abstract private method : No

private abstract void d3();

// can we create abstract final method : No

final abstract void d4();

// can we create abstract main method : No

public abstract static void main(String [] args);

// can we create abstract constructor : No

public abstract Demo5();

// can we create final method in interface : No

final void d6()

{

}

// can we create constructor in interface : No

public Demo4()

{

}

}

====================================================================

can we remodify the value for interface variables VVVVVVIMMPPPPP

No

why we can't modify the value for interface variables? VVVVVVIMMPPPPP

because once we declare any variable in interface is called final variables.

====================================================================

package Tutorial4;

public interface Demo5 {

// by default all interface variable are static and final

String a = "selenium";

public void d1();

public void d2();

public void d3();

}

package Tutorial4;

public abstract class Sample5 implements Demo5 {

public void d1() {

System.out.println("D1 method");

}

public void d2() {

System.out.println("D2 method");

}

}

package Tutorial4;

public class Pune5 extends Sample5 {

@Override

public void d3() {

System.out.println("D3 method");

}

public static void main(String[] args) {

// we can't create object of interface as well as abstract class but we can

// mention reference of interface as well as abstract class

Demo5 obj = new Pune5();

obj.d1();

obj.d2();

obj.d3();

System.out.println(Demo5.a);

}

}

====================================================================

package Tutorial4;

public interface Demo6 {

public void d1();

public void d2();

public void d3();

}

package Tutorial4;

public abstract class Sample6 implements Demo6 {

public void d1() {

System.out.println("D1 method");

}

public abstract void s1();

}

package Tutorial4;

public abstract class Pune6 extends Sample6{

public void d2()

{

System.out.println("D2 method");

}

public void s1()

{

System.out.println("S1 method");

}

public abstract void p1();

}

package Tutorial4;

public class Mumbai6 extends Pune6 {

@Override

public void d3() {

System.out.println("D3 method");

}

@Override

public void p1() {

System.out.println("P1 method");

}

public static void main(String[] args) {

Mumbai6 obj =new Mumbai6();

obj.d1();

obj.d2();

obj.d3();

obj.s1();

obj.p1();

// Pune6 obj =new Mumbai6();

// obj.d1();

// obj.d2();

// obj.d3();

// obj.s1();

// obj.p1();

// Sample6 obj =new Mumbai6();

// obj.d1();

// obj.d2();

// obj.d3();

// obj.s1();

// Demo6 obj =new Mumbai6();

// obj.d1();

// obj.d2();

// obj.d3();

}

}

==================================================================

package Tutorial4;

public interface Demo7 {

void d1();

void d2();

}

package Tutorial4;

public interface Sample7 extends Demo7 {

public void s1();

public void s2();

}

package Tutorial4;

public interface Pune7 extends Sample7{

public void p1();

public void p2();

}

package Tutorial4;

public class Mumbai7 implements Pune7 {

@Override

public void s1() {

System.out.println("S1 method");

}

@Override

public void s2() {

System.out.println("S2 method");

}

@Override

public void d1() {

System.out.println("D1 method");

}

@Override

public void d2() {

System.out.println("D2 method");

}

@Override

public void p1() {

System.out.println("P1 method");

}

@Override

public void p2() {

System.out.println("P2 method");

}

public static void main(String[] args) {

Mumbai7 obj =new Mumbai7();

obj.d1();

obj.d2();

obj.s1();

obj.s2();

obj.p1();

obj.p2();

// Pune7 obj =new Mumbai7();

// obj.d1();

// obj.d2();

// obj.s1();

// obj.s2();

// obj.p1();

// obj.p2();

// Sample7 obj =new Mumbai7();

// obj.d1();

// obj.d2();

// obj.s1();

// obj.s2();

// Demo7 obj =new Mumbai7();

// obj.d1();

// obj.d2();

}

}

====================================================================

package Tutorial4;

public interface Demo8 {

public void d1();

public void d2();

}

package Tutorial4;

public interface Sample8 {

public void s1();

public void s2();

}

package Tutorial4;

//Multiple inheritance using Interface

public class Pune8 implements Demo8, Sample8 {

@Override

public void s1() {

System.out.println("S1 method");

}

@Override

public void s2() {

System.out.println("S2 method");

}

@Override

public void d1() {

System.out.println("D1 method");

}

@Override

public void d2() {

System.out.println("D2 method");

}

public static void main(String[] args) {

Pune8 obj =new Pune8();

obj.d1();

obj.d2();

obj.s1();

obj.s2();

// Sample8 obj =new Pune8();

// obj.s1();

// obj.s2();

// Demo8 obj =new Pune8();

// obj.d1();

// obj.d2();

}

}

===================================================================

package Tutorial4;

public interface Demo9 {

public void d1();

}

package Tutorial4;

//we implements interface in abstract class using implements keyword

public abstract class Sample9 implements Demo9 {

public abstract void s1();

}

package Tutorial4;

public class Pune9 extends Sample9 {

@Override

public void d1() {

System.out.println("D1 method");

}

@Override

public void s1() {

System.out.println("S1 method");

}

public static void main(String[] args) {

Sample9 obj = new Pune9();

obj.d1();

obj.s1();

// Demo9 obj =new Pune9(); obj.d1();

}

}

=============================================================

package Tutorial4;

public abstract class Demo10 {

public abstract void d1();

}

package Tutorial4;

//can we implements abstract class in interface : NO

//why we can't implements abstract class in interface

//because in abstract class we can mention the constructor ,

//final method or non static method but in interface consturtor is not allowed, as well final method is not allowed as well as

//non static is not allowed

public interface Sample10 implements Demo10{

public void s1();

}

===========================================================================

diff between Abstract class and interface? VVVVIMMMPPPPPPPPPPPPPPP

-

abstract class

1) to create Abstract class we use abstract keyword

2) we implements abstract class methods in another class using extends keyword

3) we can't achieve the multiple interface using abstract class.

4) in abstract class contains static, non static and final variables.

5) we can implements interface in abstract class using implements keywords/

6) we can declare constructor in abstract class

7) we can declare final method in abstract class

8) we can declare protected abstract method in abstract class.

9) compulsory we have to use abstract keyword to create abstract method in abstract class.

interface class

1) to create interface we use interface keyword

2) we implements interface methods in another class using implements keyword

3) we can achieve the multiple interface using interface.

4) interface contains we have static and final variable by default.

5) we can't implements abstract class in interface.

6) we can't declare constructor in interface

7) we can't declare final method in interface.

8) we can't declare protected abstract method in interface.

9) abstract keyword is optional to create the abstract method in interface.

1) can overload the static method in java?

2) can override the static method in java and if not what is reason?

3) can we declare abstract static method in abstract class and if not what is reason?

4) can we declare abstract static method in interface and if not what is reason?

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

1) can overload the main method in java?

2) can override the main method in java and if not what is reason?

3) can we declare abstract main method in abstract class and if not what is reason?

4) can we declare abstract main method in interface and if not what is reason?

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

1) can overload the private method in java?

2) can override the private method in java and if not what is reason?

3) can we declare private abstract method in abstract class and if not what is reason?

4) can we declare private abstract method in interface and if not what is reason?

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

1) can overload the final method in java?

2) can override the final method in java and if not what is reason?

3) can we declare final abstract method in abstract class and if not what is reason?

4) can we declare final abstract method in interface and if not what is reason?

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

1) can overload the protected method in java?

2) can override the protected method in java

3) can we declare protected abstract method in abstract class

4) can we declare protected abstract method in interface and if not what is reason?

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

1) can overload the constructor in java?

2) can override the constructor in java and if not what is reason?

3) can we declare abstract constructor in abstract class and if not what is reason?

4) can we declare abstract constructor in interface and if not what is reason?

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

1) can we declare constructor in abstract class

2) can we declare constructor in interface

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

Encapsulation VVVVIMMMPPPPPPPPP

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

- it is process of wrapping the code and data in single unit.

- in Encapsulation process, we can access the private fields in another class by using getter() and setter() method

- by using Encapsulation process we control over on data.

- Encapsulation is technique to making the field as private and providing control over the data.

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

Where you have used Encapsulation concept in Framework? VVVVIMMMPPPPPPPPP

- we use Encapsulation concept in page Layer package in src/main/java as well as Test Layer package in src/test/java.

- in Page Layer package we declare all the object repository as private.

- and also in Test layer package we declare all object as private.

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

public class Demo1

{

private String variableName="value";

//getter method

public String getVariableName()

{

return variableName;

}

public void setVariableName(String newVariableName)

{

variableName= newVariableName;

}

}

package Tutorial5;

public class Demo1 {

private String fname = "Amruta";

String lname = "Patil";

// non static method with String return type: getter() method

public String getFname() {

return fname;

}

/// non static method without return type and 1 string arguments : setter()

/// method

public void setFname(String newFname) {

// re-assign value for fname variable

fname = newFname;

}

}

package Tutorial5;

public class Sample1 extends Demo1{

public static void main(String[] args) {

Sample1 obj =new Sample1();

System.out.println(obj.lname);

String a= obj.getFname();

System.out.println(a);//Amruta

obj.setFname("Pravin");//set new name

String b = obj.getFname();

System.out.println(b);//Pravin

}

}

=================================================================

package Tutorial5;

public class Demo2 {

private int numbers=9090;

//getter() method

public int getNumbers()

{

return numbers;

}

//setter() method

public void setNumbers(int newNumbers)

{

numbers=newNumbers;

}

}

package Tutorial5;

public class Sample2 {

public static void main(String[] args) {

Demo2 obj =new Demo2();

int a = obj.getNumbers();

System.out.println(a);

obj.setNumbers(8888);

int b = obj.getNumbers();

System.out.println(b);

}

}

============================================================================

package Tutorial5;

public class Demo3 {

private String firstName = "Rahul";

private String lastName = "Patil";

private int age = 25;

// getter() method

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

public int getAge() {

return age;

}

// setter() method

public void setFirstName(String newFirstName) {

firstName = newFirstName;

}

public void setLastName(String newLastName) {

lastName = newLastName;

}

public void setAge(int newAge) {

age = newAge;

}

}

package Tutorial5;

public class Sample3 {

public static void main(String[] args) {

Demo3 obj =new Demo3();

System.out.println(obj.getFirstName());

System.out.println(obj.getLastName());

System.out.println(obj.getAge());

obj.setFirstName("Amit");

obj.setLastName("Pandit");

obj.setAge(27);

System.out.println(obj.getFirstName());

System.out.println(obj.getLastName());

System.out.println(obj.getAge());

}

}

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

package Tutorial5;

public class Sample3 extends Demo3 {

public static void main(String[] args) {

Sample3 obj = new Sample3();

System.out.println(obj.getFirstName());

System.out.println(obj.getLastName());

System.out.println(obj.getAge());

obj.setFirstName("Amit");

obj.setLastName("Pandit");

obj.setAge(27);

System.out.println(obj.getFirstName());

System.out.println(obj.getLastName());

System.out.println(obj.getAge());

}

}

==================================================================

public class Demo5

{

private void abc()

{

System.out.println("abc private method");

}

public void xyz()

{

abc();

}

}

public class Sample5

{

public static void main(String [] args)

{

Demo5 obj =new Demo5();

obj.xyz();

}

}

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

package Tutorial5;

public class Demo5 {

private void abc() {

System.out.println("ABC Private method");

}

public void xyz()

{

abc();

}

}

package Tutorial5;

public class Sample5 {

public static void main(String[] args) {

Demo5 obj =new Demo5();

obj.xyz();

}

}

========================================================================

package Tutorial5;

public class Demo4 {

private void loginFunctionality() {

System.out.println("Private Login Functionality Username");

System.out.println("Private Login Functionality Password");

}

public void getLoginFunctionality() {

loginFunctionality();

}

}

package Tutorial5;

public class Sample4 {

public static void main(String[] args) {

Demo4 obj =new Demo4();

obj.getLoginFunctionality();

}

}

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

package Tutorial5;

public class Demo6 {

private void addEmployee(String firstName,String lastName)

{

System.out.println("Private Add Employee First Name is "+firstName);

System.out.println("Private Add Employee Last Name is "+lastName);

}

public void getAddEmployee(String newFirstName,String newLastName)

{

addEmployee(newFirstName, newLastName);

}

}

package Tutorial5;

public class Sample6 {

public static void main(String[] args) {

Demo6 obj = new Demo6();

obj.getAddEmployee("Dipali", "Kulkarni");

}

}

|  |  |
| --- | --- |
| **Method overloading** | **Method overriding** |
| If we have same method with different signature. | If we have same method in parent class and child class with same signature |
| Different signature means   1. Number of arguments is different 2. Type of argument is different 3. Sequence or order or arguments is different | Same signature means   1. Number of argument is same 2. Type of arguments is same 3. Sequence of arguments is same |
| It is also called static binding | It is also called dynamic binding |
| It Is also called early binding | It is also called late binding |
| We achieve the method overloading in one or more classes | We achieve the method overriding in two or more classes |
| It is called compile time Polymorphism | It is called as run time polymorphism |
| We can overload private method | We can not override the private method |
| We can overload static method | We can not override the static method |
| We can overload final method | We cannot override the final method |
| We can overload constructor | We can’t override the constructor |
| We can overload the main method | We can’t override the main method |
| There are different method is overloaded in selenium   1. implicitlyWait 2. pageLoadTimeout 3. frame() method 4. to() method 5. click() method in Actions class 6. sendKeys() method in Actions class 7. doubleClick() method in Actions class 8. contextClick() method in Actions class 9. moveToElement() method in Actions class 10. release() method in Actions class 11. clickAndHold() method in Actions class 12. dragAndDrop() method in Actions class | There we different method overriding   1. findElement() 2. findElements() 3. getScreenshotAs() 4. getText() 5. sendKeys() |

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| Can we create constructor in abstract class  Yes | Can we create constructor in interface  No |
| Can we create final method in abstract class  Yes | Can we create final method in interface  No |
| Can we create private method in abstract class  Yes | Can we create private method in interface  Yes |
| Can we create static method in abstract class  Yes | Can we create static method in interface  Yes |
| Can we create main method in abstract class  Yes | Can we create main method in interface  Yes |
| Can we create non static method in abstract class  Yes | Can we create non static method in interface  Yes but we have to use default or private access modifier |
| Can we can create public abstract non static method in abstract class  yes | Can we create public abstract non static method in interface.  Yes |
| Can we create default abstract non static method in abstract class  Yes | Can we create default abstract non static method in interface.  Yes |
| Can we create protected abstract non static method in abstract class.  Yes | Can we create protected abstract non static method in interface.  No |
| Can we create private abstract non static method in abstract class.  No | Can we create private abstract non static method in interface.  No |
| Can we create static abstract method in abstract class  No | Can we create static abstract method in interface  No |
| Can we create final abstract method in abstract class  No | Can we create final abstract method in interface  No |
| Can we create abstract constructor in abstract class  No | Can we create abstract constructor in interface  No |
| Can we create abstract main method in abstract class  No | Can we create abstract main method in abstract class  No |
| What are the different method we can declare in abstract class   1. Static method 2. Main method 3. Non static method 4. Final method 5. Private method 6. Constructor | What are the different method we can declare in interface   1. Static method 2. Main method 3. Non static method using default or private access modifier 4. Private method |
| We achieve the abstraction in abstract class for   1. Public abstract non static method 2. Default abstract non static method 3. Protected abstract non static method | We achieve abstraction in interface for   1. Public abstract non static method 2. Default abstract non static method |
| If we declare any variable in abstract class we can modify the values for it | If we declare any variable in interface , we can’t modify it, because interface variables are static and final. |
| If u want to create abstract method in abstract class, compulsory we have to use abstract keyword | If u want to create abstract method in interface, abstract keyword is optional. |