28-FEB-2020

-----------

METHOD OVERLOADING

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

When we have more than one method with same name within the class is called "Method Overloading"

--> It can differ based on:

-- Number of arguments in a method

-- Data type of arguments

-- sequence of data type

Ex: Create class with Method Overloading concept

public class MethodOL {

public void demoOne( int a, int b){

int c=a\*b;

System.out.println("multiplication of given value is: "+c);

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

Ex-2

public class MethodOverloadingEx {

public void melOv(int a,int b, int c) { // Methodname should start with small letter and Methodnames are same in both the methods, only differed by args

System.out.println("This is method-1");

}

public void melOv(int a) {

System.out.println("This is method-2");

}

public static void main(String[] args) {

MethodOverloadingEx mel=new MethodOverloadingEx();

mel.melOv(20, 30, 40);

mel.melOv(10);

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

to read substring

String str="LiveTech"

str.substring(4); //tech

str.substring(4,6); // te

WebDriver driver=new ChromeDriver();

ChromeDriver(){

------

------

}

There are some internal Constructors { Methods developed by Developers}

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

CONSTRUCTOR

===========

Constructor will automatically execute whenever we create instance object for Class

Rules for Constructor:

- Constructor name is same as Class name

- should not use "void" in Constructor

\*\* In general Constructors will use to initialize values into variables

Note: Constructor will not return value

Ex: Create class with constructor

public class ConstructorEx {

public ConstructorEx() { // We should not use "void" - remove 'void'. // Also Constructor name is same as Classname starting with Caps

System.out.println("My first constructor");

}

public static void main(String[] args) {

ConstructorEx cons= new ConstructorEx();

// No need to for the next step for return a value (i.e. cons.ConstructorEx) as in whenever an object is created Constructor will automaticall execute

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

CONSTRUCTOR OVERLOADING

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

Sometimes a class can have more than one Constructor is called Constructor overloading

Constructor can be differ based on argument list, like

- no of Arguments

- data type of arguments

- sequence of Data type of arguments

Ex: Create class with Constructor overloading

// Create a class for Constructor Overloading

public class ConstructorEx {

public ConstructorEx() {

System.out.println("My first constructor");

}

public ConstructorEx(int a) {

System.out.println("My second constructor");

}

public static void main(String[] args) {

ConstructorEx cons = new ConstructorEx(10);

ConstructorEx cons1 = new ConstructorEx();

}

}

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

INHERITANCE

===========

Acquiring properties from one class to another class is called Inheritance

Class-1 : Parent Class Class-2 : Child Class

method01(){ method01(){

} }

method02(){ call method01

call method02

}

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

Parent Class:

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

Parent Class also called as base class/ super class

From which class methods/ properties we acquire into another class that class is called Parent Class

In general parent class doesn't consist main method (i.e. not executable class)

Child Class

===========

Which class acquires methods from another class is called Child Class/ derived class / extend class

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

Extends Keyword:

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

"extends" is the keyword used to inherit the properties of parent class into child class

Syntax:

public class ParentClass extends ChildClass{

}

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

Ex: Create 2 classes with inheritance concept

Parent Class

Note: No main method in Parent Class

package oops.concept;

public class ParentClassEx {

public void parentEx() {

System.out.println("This is parent class");

}

public void sumOf() {

System.out.println("This is addition of two numbers");

}

}

Child Class:

package oops.concept;

public class ChildClassEx extends ParentClassEx{

public void parentEx() {

System.out.println(" This is child class");

}

public static void main(String[] args) {

ChildClassEx chd=new ChildClassEx();

chd.sumOf();

chd.parentEx();

}

}

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

METHOD OVERRIDING

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

When same method having in Parent class as well as in Child class it is called 'method overriding'

Using 'super' keyword we can call parent class method into child class

Syntax:

super.methodname(); // in childclass method we need to call

Ex. Create 2 classes with method overriding concept

Parent Class

Note: No main method in Parent Class

package oops.concept;

public class ParentClassEx {

public void parentEx() {

System.out.println("This is parent class");

}

public void sumOf() {

System.out.println("This is addition of two numbers");

}

}

Child Class:

package oops.concept;

public class ChildClassEx extends ParentClassEx{

public void parentEx() {

System.out.println(" This is child class");

super.parentEx();

}

public static void main(String[] args) {

ChildClassEx chd=new ChildClassEx();

chd.sumOf();

chd.parentEx();

}

}

Note: methodname is same in both the Parent Class & Child Class

By default it will return the value of Child Class if both the same methodname()

Using super keyword we can call the parent class method in Child class

============================================================END OF CLASS==========================================================================================