**Methods**

A **method** is a block of code which only runs when it is called.

You can pass data, known as parameters, into a method.

Methods are used to perform certain actions, and they are also known as functions.

**Why use methods? To reuse code: define the code once, and use it many times.**

If a method is returning any we should declare the return type.

If a method is not returning any we should declare the return type as void.

A class can have multiple methods

We can have a method inside a method/ we can call method from another method

## Java static method

If you apply static keyword with any method, it is known as static method.

* A static method belongs to the class rather than the object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* A static method can access static data member and can change the value of it.

**Types of Java methods**

• Standard Library Methods

• User-defined Methods(Static and Non Static Methods)

Note: Static methods can be called with class name(no need of object)

Non Static Methods can be called with object

Static methods can only access/call Static methods only

**Syntax**: AccessModifier returnType MethodName(ArgumentList/ parametersList){}

**Example** : programe for all types of methods

public class prim {

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

prim p =new prim();

String pstatus=p.method3(true);

System.out.println(pstatus);

System.out.println(p.method4(20, 30));

int c=prim.method1(3, 2);

System.out.println(c);

prim.method2();

}

public static int method1(int a, int b) {

int c=a+b;

return c;

}

public static void method2() {

int c=5-2;

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

}

public String method3(boolean x) {

String status=null;

if(x==true) {

status="PASS";

}

return status;

}

public int method4(int i, int j) {

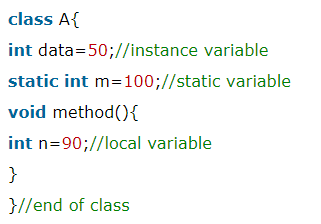
return i\*j;

}

}

|  |  |
| --- | --- |
| Difference between non-static and static methods | |
| non-static methods | static methods |
| Memory is allocated multiple times | Memory is allocated only once at the time of class loading.whenever method is calling time only. |
| It is specific to an object so that these known are also known as instance method. | These are common to every object so that it is also as member method or class method. |
| These methods always access with object reference | These property always access with class reference |

***Variables in class:- There are 2 types of variables in class***



**Static Variables:**

Example of static variable

**//Java Program to demonstrate the use of static variable**

class Student{

int rollno;//instance variable

String name;

static String college ="ITS";//static variable

//constructor

Student(int r, String n){

rollno = r;

name = n;

}

//method to display the values

void display (){System.out.println(rollno+" "+name+" "+college);}

}

//Test class to show the values of objects

public class TestStaticVariable1{

public static void main(String args[]){

Student s1 = new Student(111,"Karan");

Student s2 = new Student(222,"Aryan");

//we can change the college of all objects by the single line of code

//Student.college="BBDIT";

s1.display();

s2.display();

}

}

**Object Oriented Programming (Concepts/OOPS Concepts)**

**Class :** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

**Object:** An object in Java is the physical as well as logical entity whereas a class in Java is a logical entity only. An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical

An object has three characteristics:

* State:represents the data (value) of an object.
* Behavior: represents the behavior (functionality) of an object such as deposit, withdraw, etc.
* Identity: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

Creating an object or instance

Syntax: classname objectName=new classname();

Ex: Student s1=new Student();//creating an object of Student

***Method overloading:***

when we have more than one method with same name within the class is called Method overloading

it can be differ based on:

-number of arguments in a method

-Data type of arguments

-sequence of data type of arguments

Ex:

public class MethodOL {

public void demoOne(int a, int b){

int c= a\*b;

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

}

public void demoOne(String x, String y){

String z= x+y;

System.out.println(z);

}

public static void main(String[] args) {

MethodOL ml= new MethodOL();

ml.demoOne(5, 6);

ml.demoOne("InsightQ", "Technologies");

}

}

***Constructor:***

Constructor will automatically execute whenever we created instance object for Class

rules:

* constructor name is same as Class name
* should not use "void" in Constructor

in general Constructors will use to initialize values into variables

ex:

public class ConstEx {

public ConstEx(){

System.out.println("this is sample script");

}

public static void main(String[] args) {

ConstEx ce= new ConstEx();

}

}

***Constructor overloading:***

sometimes in a class we can have more than one Constructor is called Constructor overloading

constructors can be differ based on argument list

ex:

public class ConstEx {

public ConstEx(int a, int b){

int c= a\*b;

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

}

public ConstEx(String x, String y){

String z= x+y;

System.out.println(z);

}

public static void main(String[] args) {

ConstEx ce= new ConstEx(5,6);

ConstEx cex= new ConstEx("Mindq","Systems");

}

}

### **Program of the counter without static variable**

//Java Program to demonstrate the use of an instance variable

//which get memory each time when we create an object of the class.

class Counter{

int count=0;//will get memory each time when the instance is created

Counter(){

count++;//incrementing value

System.out.println(count);

}

public static void main(String args[]){

//Creating objects

Counter c1=new Counter();

Counter c2=new Counter();

Counter c3=new Counter();

}

}

### **Program of counter by static variable**

//Java Program to illustrate the use of static variable which

//is shared with all objects.

class Counter2{

static int count=0;//will get memory only once and retain its value

Counter2(){

count++;//incrementing the value of static variable

System.out.println(count);

}

public static void main(String args[]){

//creating objects

Counter2 c1=new Counter2();

Counter2 c2=new Counter2();

Counter2 c3=new Counter2();

}

}

**In Java, there 4 type of the access specifiers.**

* + public: accessible in all class in your application.
  + protected: accessible within the class in which it is defined and in its subclass(es)
  + private: accessible only within the class in which it is defined.
  + default (declared/defined without using any modifier) : accessible within same class and package within which its class is defined.

***Inheritance:***

* acquiring properties from one class to another class is called Inheritance parent class:
* parent class also called as base class/super class from which class methods we call into another class that class is called parent class

***child class:***

which class acquires methods from another class is called Child class/derived class/extened class

***extends Keyword:***

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

syntax:

public class ChildClass extends ParentClass{

}

ex:

***step 1: create parent class {i.e. ParentClass} with 2 submethods and without main method (i.e. non-executable class)***

script:

package oops.concepts;

public class ParentClass {

public void login(){

System.out.println("setup program");

}

public void tearDown(){

System.out.println("it is logout program");

}

}

Step 2: create child class with "extends" keyword extends Keyword:

extends is the keyword used to inherit the properties of other class

ex:

package oops.concepts;

public class ChildClass extends ParentClass{

public static void main(String[] args) {

ChildClass cc= new ChildClass();

cc.login();

cc.tearDown();

}

}

***method overriding:***

whereas same method having in Parent class as well as childclass

ex:

package oops.concepts;

public class ChildClass extends ParentClass{

public void login(){

System.out.println("setup program2");

}

public static void main(String[] args) {

ChildClass cc= new ChildClass();

cc.login();

cc.tearDown();

ParentClass pp= new ParentClass();

pp.login();

}

}

Super keyword in Overriding

super keyword is used for calling the parent class method/constructor in child class

syntax:

super.methodname()

ex:

public class MyChildClass extends MyBaseClass{

public void setUp(){

System.out.println("Login with Invalid data");

super.setUp();

}

public static void main(String[] args) {

MyChildClass mcc= new MyChildClass();

mcc.setUp();

mcc.tearDown();

MyBaseClass mbc= new MyBaseClass();

mbc.setUp();

}

}

NOTE:- Sometimes we may have variable with some name in class level & method level in that case to specify class level variable in method we use "this" keyword.

***SYNTAX:- this.variable\_name***

Ex:1:- Create class which have the same variable name in class level and method level

script:

public class CalculationsEx {

static int result;//Class variable

//to perform addition

public void addOne(int a, int b) {

int result; //method level variable

result= a+b;

System.out.println(result);

this.result=result;

}

public static void main(String[] args) {

CalculationsEx ce= new CalculationsEx();

ce.addOne(10, 20);

System.out.println("Sum of given values is: "+result);

}

***Exception Handling:- Exception handle is a set of code that handles an exception.***

***There are two types of Exceptions***

***(i) Compile time Exceptions.***

***(ii) Runtime Exceptions.***

***(i) Compile time Exceptions.***

***There are two types of 1) Syntax errors***

***2) Semantic errors***

***1) Syntax errors: Ex:- String a;***

***Stg a; -->Syntax Error***

***2) Semantic errors: Variables may declared twice***

***EX:- int a;***

***---------------------***

***--------------------***

***int a;***

***Note:- These can be highlighted in red, in Eclipse while writing the programme by default it will compile.***

***(ii) Runtime Exceptions.***

* ***Which are interrupts the normal flow of programme execution.***
* ***good programme should handle all the exceptions and continue with its normal flow of programme execution.***
* ***Exception can be handled in java using "try" & "catch" block.***

***Syntax:- { for try& catch }***

***try{***

***statement(s)***

***}***

***catch (Exception Exception\_name){***

***statement(s)***

***}***

***NOTE:- >>> The finally block is executed irrespective of an exception being raised in the try block.***

***>>> It is an optional to use with a tryt block.***

***Syntax:- { for try& catch with finally }***

***try{***

***statement(s)***

***}***

***catch (Exception Exception\_name){***

***statement(s)***

***}***

***finally{***

***statement(s)***

***}***

***EX-1:-*** ***try***

*{*

***int*** *a=10;*

***int*** *b=5;*

***int*** *c=a/b;*

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

*}*

***catch****(Exception e) {*

*System.****out****.println("b-value should be zero");*

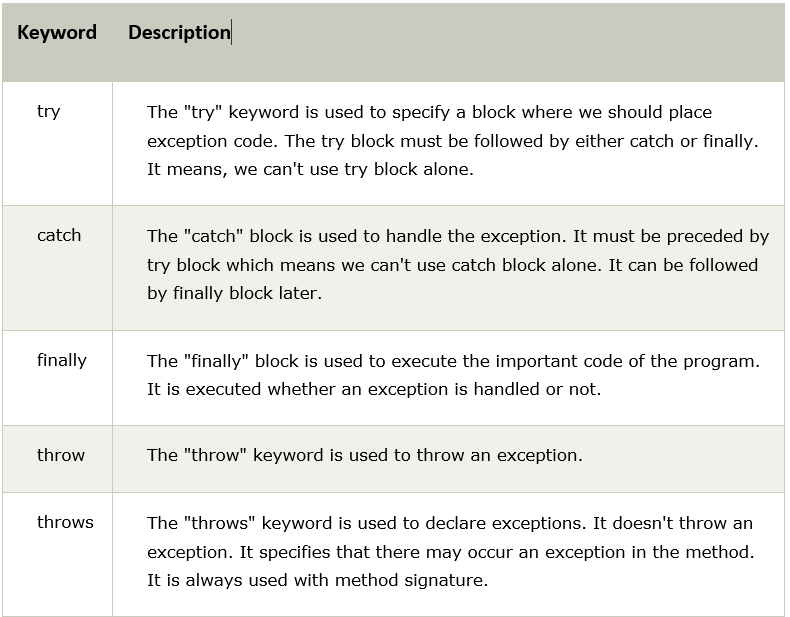
*}*

***finally*** *{*

*System.****out****.println("Demo for try catch block");*

*}*

***There are 5 keywords which are used in handling exceptions in Java.***



**Java Exceptions:**

1. **Arithmetic Exception**  
   It is thrown when an exceptional condition has occurred in an arithmetic operation.
2. **ArrayIndexOutOfBoundException**It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
3. **ClassNotFoundException**This Exception is raised when we try to access a class whose definition is not found
4. **FileNotFoundException**This Exception is raised when a file is not accessible or does not open.
5. **IOException**It is thrown when an input-output operation failed or interrupted
6. **InterruptedException**It is thrown when a thread is waiting , sleeping , or doing some processing , and it is interrupted.
7. **NoSuchFieldException**It is thrown when a class does not contain the field (or variable) specified
8. **NoSuchMethodException**It is thrown when accessing a method which is not found.
9. **NullPointerException**This exception is raised when referring to the members of a null object. Null represents nothing
10. **NumberFormatException**This exception is raised when a method could not convert a string into a numeric format.
11. **RuntimeException**This represents any exception which occurs during runtime.
12. **StringIndexOutOfBoundsException**It is thrown by String class methods to indicate that an index is either negative than the size of the string