1.A static block executes once in the life cycle of any program

|  |  |
| --- | --- |
| down vote | Static block can be used to show that a program can run without main function also.  //static block  //static block is used to initlize static data member of the clas at the time of clas loading  //static block is exeuted before the main  class B  {  static  {  System.out.println("Welcome to Java");  System.exit(0);  } |

https://ssl.gstatic.com/ui/v1/icons/mail/images/cleardot.gif

2.method calling/with no retun and parameters/with retun nad no parameters/with retun and parameters/with storing value.

public class

{

int x;

int method(int y)

{

x=y;

return x;

}

}

public satic void main (string args[])

{

classname t = new classname();

t.method();

int total=t.method(24);

system.out.println("total is:" +x)

}

https://ssl.gstatic.com/ui/v1/icons/mail/images/cleardot.gif

3.try/catch/finally

class TestExceptions {

static void myMethod(int testnum) throws Exception {

System.out.println ("start - myMethod");

if (testnum == 12)

throw new Exception();

System.out.println("end - myMethod");

return;

}

public static void main(String args[]) {

int testnum = 12;

try {

System.out.println("try - first statement");

myMethod(testnum);

System.out.println("try - last statement");

}

catch ( Exception ex) {

System.out.println("An Exception");

}

finally {

System. out. println( "finally") ;

}

System.out.println("Out of try/catch/finally - statement");

}

}

4.variable

 variable in Java has a specific type, which determines the size and layout of the variable's memo

int a, b, c; // Declares three ints, a, b, and c.

int a = 10, b = 10; // Example of initialization

byte B = 22; // initializes a byte type variable B.

double pi = 3.14159; // declares and assigns a value of PI.

char a = 'a'; // the char variable a iis initialized with value 'a'

5.method

A Java method is a collection of statements that are grouped together to perform an operation. When you call the System.out.**println()** method, for example, the system actually executes several statements in order to display a message on the console.

**Syntax**

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

// body

}

6.void

The void keyword allows us to create methods which do not return a value.

public class ExampleVoid {

public static void main(String[] args) {

methodRankPoints(255.7);

}

public static void methodRankPoints(double points) {

if (points >= 202.5) {

System.out.println("Rank:A1");

}else if (points >= 122.4) {

System.out.println("Rank:A2");

}else {

System.out.println("Rank:A3");

}

}

}

## Method Overloading:

When a class has two or more methods by the same name but different parameters, it is known as method overloading. It is different from overriding. In overriding, a method has the same method name, type, number of parameters, etc.

public class ExampleOverloading {

public static void main(String[] args) {

int a = 11;

int b = 6;

double c = 7.3;

double d = 9.4;

int result1 = minFunction(a, b);

// same function name with different parameters

double result2 = minFunction(c, d);

System.out.println("Minimum Value = " + result1);

System.out.println("Minimum Value = " + result2);

}

// for integer

public static int minFunction(int n1, int n2) {

int min;

if (n1 > n2)

min = n2;

else

min = n1;

return min;

}

// for double

public static double minFunction(double n1, double n2) {

double min;

if (n1 > n2)

min = n2;

else

min = n1;

return min;

}

}

Exceptions:

checked Exceptions:

**Checked exceptions** − A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions.

import java.io.File;

import java.io.FileReader;

public class FilenotFound\_Demo {

public static void main(String args[]) {

File file = new File("E://file.txt");

FileReader fr = new FileReader(file);

}

}

### Output

C:\>javac FilenotFound\_Demo.java

FilenotFound\_Demo.java:8: error: unreported exception FileNotFoundException; must be caught or declared to be thrown

FileReader fr = new FileReader(file);

^

1 error

**Unchecked exceptions** − An unchecked exception is an exception that occurs at the time of execution. These are also called as **Runtime Exceptions**.

public class Unchecked\_Demo {

public static void main(String args[]) {

int num[] = {1, 2, 3, 4};

System.out.println(num[5]);

}

}

### Output

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5

at Exceptions.Unchecked\_Demo.main(Unchecked\_Demo.java:8)

Inheritance:

## extends Keyword

**extends** is the keyword used to inherit the properties of a class. Following is the syntax of extends keyword.

class Calculation {

int z;

public void addition(int x, int y) {

z = x + y;

System.out.println("The sum of the given numbers:"+z);

}

public void Subtraction(int x, int y) {

z = x - y;

System.out.println("The difference between the given numbers:"+z);

}

}

public class My\_Calculation extends Calculation {

public void multiplication(int x, int y) {

z = x \* y;

System.out.println("The product of the given numbers:"+z);

}

public static void main(String args[]) {

int a = 20, b = 10;

My\_Calculation demo = new My\_Calculation();

demo.addition(a, b);

demo.Subtraction(a, b);

demo.multiplication(a, b);

}

}

**Output**

The sum of the given numbers:30

The difference between the given numbers:10

The product of the given numbers:200

Over ridding:

 If a class inherits a method from its superclass, then there is a chance to override the method provided that it is not marked final.

class Animal {

public void move() {

System.out.println("Animals can move");

}

}

class Dog extends Animal {

public void move() {

System.out.println("Dogs can walk and run");

}

}

public class TestDog {

public static void main(String args[]) {

Animal a = new Animal(); // Animal reference and object

Animal b = new Dog(); // Animal reference but Dog object

a.move(); // runs the method in Animal class

b.move(); // runs the method in Dog class

}

}

### Output

Animals can move

Dogs can walk and run

Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

public interface Vegetarian{}

public class Animal{}

public class Deer extends Animal implements Vegetarian{}

Deer d = new Deer();

Animal a = d;

Vegetarian v = d;

Object o = d;

## Implementing Interfaces

A class uses the **implements** keyword to implement an interface. The implements keyword appears in the class declaration following the extends portion of the declaration.

/\* File name : MammalInt.java \*/

public class MammalInt implements Animal {

public void eat() {

System.out.println("Mammal eats");

}

public void travel() {

System.out.println("Mammal travels");

}

public int noOfLegs() {

return 0;

}

public static void main(String args[]) {

MammalInt m = new MammalInt();

m.eat();

m.travel();

}

}

### Output

Mammal eats

Mammal travels