Java

27 February 2024

11:54

W3School

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

public class Main{

public static void main(String[] strArg){

System.out.println("Hello world");

}

}

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

* Javac Main.java //compile
* Java Main //run
* Syntax
  + All lines should be part of class
  + Class name starts with upper case
  + Function/ varible name starts with small case
  + Use camelcase
  + Class name should match the filename
* Print() is similar to println(), it does not insert newline at the end
* Comments , similar to c++
* System.out.println("Hello "+name);
* Primitive data types: byte(1), short(2), int(4), long(8), float(4), double(8), boolean(1 bit), char(2)
* Float and double init end with f and d, floatf1 =35e3f;doubled1 =12E4d;
* Non premitive data types : String, Array, Class, Interface ,Starts with upper case and have null ad default
* Casting can be done as old c++ style (int)
* Operators : same as c++
* String: .length(), toUpperCase(), toLowerCase(), indexOf()
* Escape characters same as c++
* Math class : max(), min(), sqrt(), abs(), random()
* Boolean class
* If, else, else if : same as c++
* Stringresult =(time <18)?"Good day.":"Good evening.";
* Switch : same as c++
* Loop : for, while , do while: same as c++
* Break, continue: same as c++
* Array: String[]cars ={"Volvo","BMW","Ford","Mazda"}; int[][]myNumbers ={{1,2,3,4},{5,6,7}}; access similar to c++
* Methods : same as c++
* ---------------------------------

Public class Main{

int x = 5;

public static void main(String[] args){

Main oMain = new Main();

System.out.println(oMain.x);

}

}

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

* Final and constant modifier makes variable immutable
* Constructor same as c++
* Access specifier for class: public, default(only available in same package)
* Access specifiers for attr and methods: public, private, protected, default(in same package)
* Non access specifiers
  + For class
    - final: can't be inherited
    - abstract : objects can't be created as few methods don’t have definition
  + For attr and methods
    - final: immutable
    - static : one copy per class, accessed by class name
    - abstract: methods without body , func()=0 in c++
    - transient : skipped during serialization
    - synchronized : accessed by one thread at a time
    - volatile : always read from main memory. Not cached
* Package
  + import package.name.class // to use
  + package mypack; // to create package
  + javac -d . MyPackageClass.java // to compile package
  + Package name starts with lower case

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

import java.util.Scanner

public class Main{

public static void main(String[] args){

Scanner oScanner = new Scanner();

String strInput = oScanner.nextLine();

}

}

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

* Inheritance : public class Car extends Vehicle{}
* Runtime polymorphism : same as c++
* Inner/nested class : the constuctor call will be decided based on modifier/ specifier. For static, inner class constr can be called by outerClassName.new innerClassName()
* Interface: class with no method body, pure abstract
  + interface Animal{// no need for abstract specifier}
  + class Dog implements Animal{// need to define all abstract methods in interface}
* Only one class can be extended, but multiple interfaces can be iplemented
* enum Days{SUNDAY, MONDAY…}; Days eDays = Days.SUNDAY;
* enums inside a class can be used without names in switch and if statements
* Scanner class :nextLine(), nextBoolean(), nextByte() for all primitive type
* java.time package: calsses LocalDate, LocalTime, LocalDateTime, DateTimeFormatter
* Java.util.Arraylist;
  + Arraylist<String> arrString = new ArrayList<String>();
    - arrString.add("hello");
    - arrString.set(0, "hi");
    - arrString.clear();
    - arrString.get(0);
    - arrString.size();
    - arrString.remove(0);
* Collection.sort(arrString); // java.util.Collection
* LinkedList similar to ArrayList (same methods, inner implementation varries) // java.util
  + linkedList methods: addFirst, addLast, removeFirst, removeLast, getFirst, getLast
* HashMap<String, String> oHashMap = new ….
  + Put("a","b"), get("A"), remove("A"), clear(), size(), keySet() // all key, values(),
* HashSet : unique element
  + Add, contains, remove , clear, size
* To use primitive types in containers , always use wrapper classes like Integer for int etc.
* Iterator<String>it =cars.iterator(); it.next() // to iterate, hasNext() //to check
* Wrapper classes : for all primitive type, starts with caps, have helper funtion like tostring etc.
* Try, catch, throw, finally
* Use Regex class for regular expression
* Thread
  + Extend Thread and define run()
  + Implement runnable and define run() // for multiple inheritance use this as no multiple extends
* ---------------------------------------

public class Main extends Thread{

public void run(){

//

}

public static void main(){

Main oThread = new Main();

oThread.start();

}

}

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

public class Main implements Runnable{

public void run(){

//

}

public static void main(){

Main oMain = new Main();

Thread oThread = new Thread(oMain);

oThread.start();

}

}

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

* Interface with multiple extends : public interface OmniVore extends Animal, Mammal{}
* Lambda expression: similar to lambda method , without name (parameter1, parameter2) -> {code}
* [Java Lambda Expressions (w3schools.com)](https://www.w3schools.com/java/java_lambda.asp) // details
* Java.io.File class: canRead, canWrite, createNewFile, delete, exists, getName, getAbsolutePath, length, list, mkdir
  + File oFile = new File("a.txt");
  + FileWriter oFileWriter = new FileWriter(path); oFileWriter.write(""); close();
  + Use Scanner(new File(path)) as reader
* [Java Keywords (w3schools.com)](https://www.w3schools.com/java/java_ref_keywords.asp)