for data types:

byte, short, int, long, float, double, char, boolean

* These are primitive data types, table shows below.

一張含有 文字, 窗戶, 組, 螢幕擷取畫面 的圖片

自動產生的描述

flow control:

if, else -

Use “if” to specify a block of code to be executed, if a specified condition is true.

Use “else” to specify a block of code to be executed, if the same condition is false.

Use “else if” to specify a new condition to test, if the first condition is false.

if (condition) {

// block

}else if (condition) {

// block

} …

else{

// block

}

switch, case -

Use the switch statement to select one of many code blocks to be executed.

switch(expression) {

case x:

// code block

case y:

// code block

…

}

default -

Specify the default value in a switch case statement.

switch (month) {

case 2:

return 28;

case 4:

case 6:

case 9:

case 11:

return 30;

default:

return 31;

}

Declare default values in a Java annotation.

public @interface Editable {

boolean value() default false;

String name() default "Untitled";

}

Declare default method in an interface.

public interface Animal {

public void eat();

public void move();

public default void sleep() {

// implementation goes here...

}

}

for -

Using for knowing exactly how many times you want to loop through a block of code.

for (statement 1; statement 2; statement 3) {

// code block to be executed

}

do, while -

The while loop loops through a block of code as long as a specified condition is true.

while (condition) {

// code block to be executed

}

---------

do {

// code block to be executed

}

while (condition);

break, continue -

The break statement can be used to jump out of a loop.

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

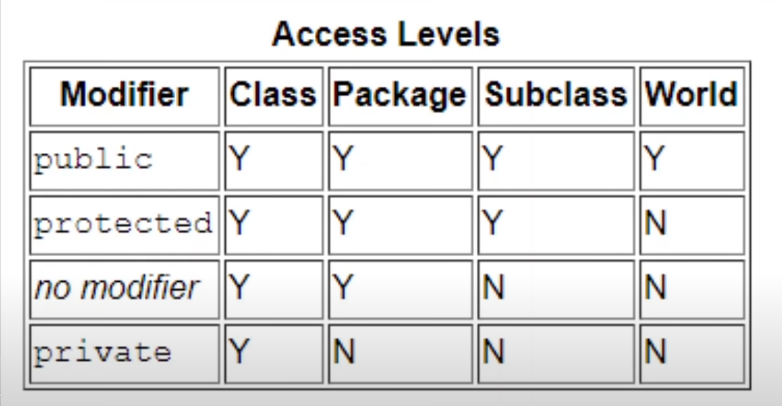
return -

Methods returning a value. e.q. - return val;

Methods not returning a value. e.q. - return;

modifiers

public, private, protected -



static, final - review training note.

abstract -

The abstract keyword is a non-access modifier, used for classes and methods.

Class: An abstract class is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

Method: An abstract method can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

synchronized -

As Java is a multi-threaded language, it supports a very important concept of Synchronization.

The process of allowing only a single thread to access the shared data or resource at a particular point of time is known as Synchronization.

1.Synchronized Instance methods:

public class MyCounter {

private int count = 0;

public synchronized void increment(int value) {

this.count += value;

}

public synchronized void decrement(int value) {

this.count -= value;

}

}

2. Synchronized Static Methods

public static MyStaticCounter {

private static int count = 0;

public static synchronized void increment(int value) {

count += value;

}

}

native -

The native keyword in Java is applied to a method to indicate that the method is implemented in native code using JNI (Java Native Interface). The native keyword is a modifier that is applicable only for methods, and we can’t apply it anywhere else.

public static void main(String[] args){

int var;

NameOfDLLFile obj = new NameOfDLLFile();

obj.var = null;

System.out.println("Before native method: var = " + var);

obj.test();

System.out.println("After native method: var = " + var);

}

// Native method

public native void test(){

static{

System.loadLibrary("NameOfDLLFile");

// Above C code in loaded in the JVM

}

}

strictfp -

strictfp is a modifier that stands for strict floating-point which was not introduced in the base version of java. It is used in java for restricting floating-point calculations and ensuring the same result on every platform while performing operations in the floating-point variable.

strictfp class Test {

// All concrete methods here are implicitly strictfp.

}

strictfp interface Test {

// All methods here becomes implicitly

// strictfp when used during inheritance.

}

class Car {

// strictfp applied on a concrete method

strictfp void calculateSpeed(){}

}

transient -

The transient keyword in Java is used to avoid serialization. If any object of a data structure is defined as a transient, then it will not be serialized.

transient int id; // This will not serialized.

volatile -

volatile keyword is used to mark a Java variable as "being stored in main memory" and not just to the CPU cache.

public volatile int counter = 0;

exception handling

try, catch -

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

try {

// Block of code to try

}

Catch(Exception e) {

// Block of code to handle errors

}

finally -

The finally statement lets you execute code, after try...catch, regardless of the result.

try {

// Block of code to try

}

Catch(Exception e) {

// Block of code to handle errors

}

finally {

// Block of code

}

throw, throws -

The throw keyword in Java is used to explicitly throw an exception from a method or any block of code.

The throws is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions.

try {

throw new NullPointerException("demo");

}

catch(NullPointerException e) {

System.out.println("Caught inside fun().");

throw e; // rethrowing the exception

}

type method\_name(parameters) throws exception\_list {

// block

}

assert -

assert is a Java keyword used to define an assert statement. An assert statement is used to declare an expected boolean condition in a program. If the program is running with assertions enabled, then the condition is checked at runtime. If the condition is false, the Java runtime system throws an AssertionError

assert expression1 [: expression2];

assert list != null && list.size() > 0 : "list variable is null or empty";

Object value = list.get(0);

class related

class - declare a class

package - declare a package

import - declares a Java class to use in the code

extends – inheriting a class

implements – inheriting interface(s)

interface – declare an interface

Object related

new - create a new object in heap memory

instanceof – compare two objects is the same datatype or not, return boolean

super - it is a reference variable that is used to refer parent class objects.

this - refers to the current object in a method or constructor.