DAY-54 [ACCESS-MODIFIES PART-2]

------

static access-modifier:

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

--> static access modfiers is applicable only on methods and variables and not applicable on top level classes.

static can be used for inner classes [static nested class]

legal combinations:

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

static variable

static block

static method

static nested class

native access\_modifier:

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

native access\_modiers are applicable only to methods.It is not applicable to variables,inner class,top-level class.

native keyword is used because:

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

1. to improve the performance of the program.

2. to achive the memory level communication

3. to use already existing legacy non-java code from c++

syntax for native keyword is:

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

1. load the native libraries

2. declare a native method

3. invoke a native method

EXAMPLE:

---------

class Demo

{

static

{

System.out.println("native libraries path:");

}

public native void fun1();

// implementation is already available

}

class Demo1

{

public static void main(String[] args)

{

Demo d = new Demo();

d.fun1();

}

}

OUTPUT:

-------

native libraries path:

Exception in thread "main" java.lang.UnsatisfiedLinkError: Demo.fun1()V

at Demo.fun1(Native Method)

at Demo1.main(Demo1.java:15)

illegal combinations with native:

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

native and abstract

native and strictfp

adv: performance of the program

dis\_adv: program may become platform dependent.

transient access\_modifier:

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

--> this access\_modifier is only applicable for variables.It is used only during the process of 'serialization mechanism'.

serialization : saving the data of an object in a file.

de-serialization : taking the object data from the file.

note: for sensitive data in a object we must use transient keyword.

volatile access-modifiers:

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

--> it is applicable only to variable.

int a = 25; --> p1/t1

int a = 25; --> p2/t2

int a = 25; --> p3/t3

but,

creation of seprate local copies evrytime leads to complexity of program execution.

and also memory usage will be ineffient.

the problem of 'data consistency ' can be solved.

note: it is almost deprecated modifier in java.

REFER DIA: 1 for conclusion.

volatile