Ch-2 (notes)
JAVA - Programming Introduction

2022.10.08 08:34

Comments:- are the description about the features of a Program. They are being ignored by the compiler. There are 3 types of comments in JAVA.

- **Single line comments:-**

```
//This is my single line Comment in JAVA.
```

- **multiple line comments:-**

```
/*This is multiline
Comments in JAVA
used to refresh
many lines as Comment*/
```

- **JAVA documentation Comments:-** used to Provide descrip for every feature in a JAVA Program. helps in creating a .html file called API (Application Programming interface) document.

```
/** description about a class */
class code

/** description about a method */
method Code
```

① What is api document?

=> it is an .html file that Contains description of all the features of a Software, Product, or a technology.

(Starting a Java Program)

```
/* First Java Program
written by Ranjit Chhetri
Date:- 22nd Aug 2021 */

import java.lang.System;
import java.lang.String;        } either two or

import java.lang.*; //importing
class first
{
    Public static void main(String args[])
    {
        System.out.Print("welcome to Java");
    }
}
```

- import java.lang.*;
  - import → to bring the code after execution from Java library
  - java.lang.* → Package containing classes eg: System & String.
  - * → all classes and interface of that Package.

② #include vs import statement?

⇒ #include directive makes the compiler go to the C/C++ Standard library and copy the code from the header file into the program. that increases Program size and waste memory or Processor time.

eg:-

#include<stdio.h>
(10 lines of code) ⟶ will be (510 lines)
(500 (all files inside header files))
+
(10 lines of actual code)

import :- makes the JVM to go the Java Standard library, ex the code there and substitute the ~~code there~~ result into the Program. no code is copied and help us to save memory or Processor time.

eg:

```
import java.lang.*;
```

---

class :- as Java is Purely OOPS so we can't write any Program without any class or object.

eg:

```
class birds
{
   variable
   +
   methods
}
```

Object creation:

Syntax for creating object is

```
classname objectname = new classname();
```

objectname → new name

new → keyword to allocate/create.

```
First obj = new first();
```

eg:

```
class first
{
    Public static void main(String args[])
    {
        First obj = new first();
    }
}
```

```
Public static void main(String args[])
```

Public → Public calling available to JVM.

static → To call and execute without creating an object

args[] → array of String (to store value pass to main method).

```
System.out.Println("This is a coding");
```

System → class name

out → static variable (in system class) → field

Println → Print in next line

(P1) Program to find the sum of two numbers:

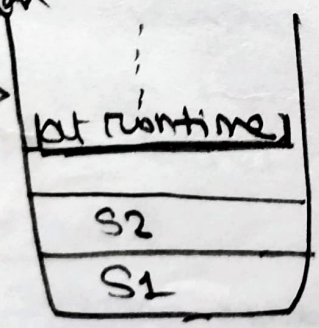
```
import java.lang.*;

class sum
{
    Public static void main(String args[])
    {
        int x=10;
        int y=20;

        int z=x+y;

        System.out.Println(z);
    }
}
```

---

* [Memory areas of JAVA Program and its uses?]

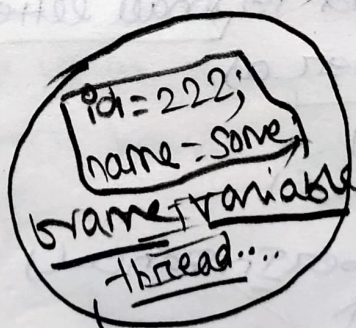
(i) (Heap memory)

garbage collection is used here →



→ used to hold the objects in memory.
· constant pool,

(ii) (Stack memory)



→ used to hold variable and

(iii) Program counter register?
→ address of the current execution of Program.

(iv) native method stack
→ contain all native method used by Program

(v) class area (method)
↳ contain class part