ASSIGNMENT 14

1)EXPLORE JUST-IN-TIME-COMPILER

Bytecode is one of the most important features of java that aids in cross-platform execution. The way of converting bytecode to native machine language for execution has a huge impact on its speed of it. These Bytecode have to be interpreted or compiled to proper machine instructions depending on the instruction set architecture. Moreover, these can be directly executed if the instruction architecture is bytecode based. Interpreting the bytecode affects the speed of execution. In order to improve performance, JIT compilers interact with the Java Virtual Machine (JVM) at run time and compile suitable bytecode sequences into native machine code. While using a JIT compiler, the hardware is able to execute the native code, as compared to having the JVM interpret the same sequence of bytecode repeatedly and incurring overhead for the translation process

2)EXPLORE THE .CLASS FILE OF ANY EXISTING CODE AND CHECK IT

with the help of javap(java print) we can explore .class file when we run command javap filename.java it will show the .class file with all the things which is added by complier at compile time like default constructer and in interface complier add final static before variable and add abstract keyword in method signature

3)DIFFERENCE BETWEEN ACCESS SPECIFIER AND ACCESS MODIFIER?

Java provides entities called “Access Modifiers or access specifiers” that help us to restrict the scope or visibility of a package, class, constructor, methods, variables, or other data members.

These access modifiers are also called “Visibility Specifiers”.

Access modifiers:

Default : When no access modifier is specified, it is treated as default modifier. Its scope is limited within the package.

Public: The word itself indicates that it has scope everywhere, i.e; it is visible everywhere even outside the package.

Private: It has scope only within the class

Protected : Its scope limits within the package and all sub classes.

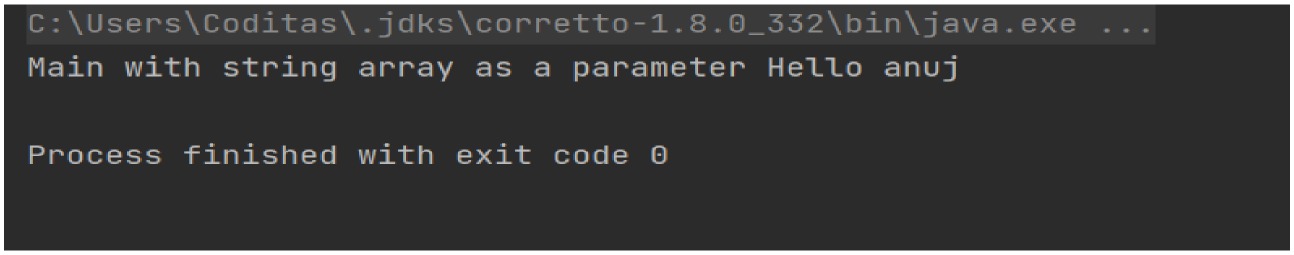
3. Non-access modifiers are those keywords which do not have anything related to the level of access but they provide a special functionality when specified.

4. Eg:- Final, Strictfp, Static, Abstract.

4)CAN WE HAVE MULTIPLE MAIN METHODS IN CLASS?

Yes we have multiple main method inside class but we have to pass different parameter same like overloading

```
package DAY_2_20_07_2022_JAVA;
public class CheckMultipleMain {
  public static void main(String args[])
  {
     System.out.println("Main with string array as a parameter Hello anuj");
  }
  public static void main(int arr[])
  {
     System.out.println("Main with int array as a parameter");
  }
}
```



```
C:\Users\Coditas\.jdks\corretto-1.8.0_332\bin\java.exe ...
Main with string array as a parameter Hello anuj

Process finished with exit code 0
```

5) Can we overload and override main method?

So the answer is yes we can overload main method but we cannot override it because we can't override static method (static method is called only with ClassName

6)Can i write main method as private protected and default?

Yes we can declare main method as private ,protected , default it complies successfully without error but at runtime,it says that the main method is not public

7)without main method can we execute our code?How

Yes we can execute java code without main method with the help of static block(because static block execute before main method and we have to stop the program after static block execution with system.exit(0) otherwise it will throw an error main method not found) but we can use this only with jdk 1.6 or previous

and we can use Application class of javafx

8)Can we change the return type of main method

No we can't if we try to change the return type complier will throw

exception

Main method must return a value of type void

9)Explore keyword strictfp

strictfp is a modifier that stands for strict floating-point which was not introduced in the base version of java as it was introduced in Java version 1.2. 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.

suppose we divide 10.3 in window so we will get 1.3333

in linux we get 1.33334

in mac we get 1.33335

so if we want same value in all operating system we have to use strictfp