Uttara InfoSolutions

www.uttarainfo.com

Compiler and JVM checks:

1) There can be any number of class definitions (class header + class body) in a .java file.

class header template:

```
<access sp> <opt modifiers> class <identifier> <opt extends/implements identifier>
{
  // class body containing members
}
```

However only one class definition can be marked with the 'public' access specifier. Others should not have any explicit access specifier mentioned (implies package scope). Also the name of the .java file should be the same as that of the public class defined in it.

Try to break this rule and see what happens by quickly creating a few class definitions per file and trying out permutations/combinations.

H.W: Can you have a file with no public class defined? If yes, then what can you name the file?

2) The compiler DOES NOT CHECK whether a class definition has a main() method or not. The compiler does not care about it. Remember in a 100 class program, maybe only one or two classes will have main(). After all main() is just a starting point of program execution. We will use the other classes to create objects in this main() which we will be learning soon in the next set of classes.

3) The compiler checks syntax and semantics of the .java file by going through every line of code. If everything is proper only then it will create the .class bytecode file. You should be able to play the role of the compiler by identifying if a program code will compile or not. I will send you some examples on this soon.

4) The compiler is given the .java file name to compile. As many classes are defined in the file, that many .class files will be created. The JVM is given one class file name to run. The JVM will execute only the main() method of that class only. So remember you might have

a .java file with 3 class definitions and all of them having main(); which one will be executed depends on which class name you pass to the JVM.

5) The JVM begins execution from the main() of the class you have given as input. Remember the JVM will search for the main() with the following header:

public static void main(String[] args)

If you change the method header, the code might compile but it definitely will not be executed by the JVM. The JVM is hardcoded to begin execution from this method header alone.

H.W: What is the only thing you change in this method header?

6) java TextX -> The lifecycle of the JVM begins when java.exe is executed by the OS. The OS gives memory as a temporary resource to the JVM process and the JVM creates the stack/heap mem areas in this and only then it will search for the main() method header to start executing the program.
Our program lifecycle begins with the main() and ends with the main().

<u>Datatypes & Operators:</u>

1) There are 8 primitive datatypes classified as integer, decimal and boolean datatypes. String is not a primitive in Java.

2) You should know the type of data, bitsize, range of values each datatype indicates. Based on the range of the values you want for a particular variable, you will choose one datatype over the other.

3) Integer datatypes are implemented using 2`s compliment numbering. Decimal datatypes are implemented using IEEE754 format representation. Hence the range of decimals are much larger than integer ones. IEEE754 format can represent infinity and NaN(NotANumber) whereas 2s compliment datatypes cannot. Hence integer division throws an exception when you divide by zero.

4) The datatypes working is fixed irrespective of the OS or hardware on which you run the Java program.

5) Decimals cannot represent all(decimal) values with precision, especially negative powers of 1. So do not use this datatype for currency based arithmetic. Use int/long or BigDecimal for precision based calculations.

6) There are unary, binary and ternary operators. Binary operators are arithmetic, conditional/logical, relational and bitwise. You should be able to recognize all of the

operators and importantly know what datatyped operands t hey take as input and what datatyped operand they return as output.

For ex: Try doing this in a main()...

System.out.println(3/4); // what do you think will be the value printed out? Now try it out. You will know the answer only when you know how the '/' operator works based on the operands it takes.

7) Arithmetic operators -> all of them take number datatypes (decimal or integer) except '+' which can also take String as its operands datatypes. Only '+' is overloaded in them.

What is the result and result`s datatype for the following operations?
a) 10 + 10
b) 10 + 10L
c) "10" + 10
d) 10.0 + 10
e) 10 / 0.0
f) 10f / 0
d) 10s / 10l

8) Bitwise operators -> &,|,<<,>>,>>>,^

What is the result of
a) 8 << 1
b) 8 << 2.5
c) 8 >> 3
d) 8 >> 32
e) 8 << 33
f) -8 >>> 1
g) 8 >>> 1

9) &,| can be used with boolean operands. Do not use it as they are not short-circuit enabled.

10) Arithmetic operators can overflow/underflow without showing any failures at compile time or at runtime -> Silent failures.

11) Casting is done to satisfy compiler. JVM does conversion at runtime anywhere casting is used. Casting means converting one datatype to another.

In general, datatypes from smaller to bigger ones are casted automatically -> up/implicit casting.

From bigger datatypes, if you want to move to a smaller datatyped variable ->
down/explicit casting is required.