Introduction to Programming

Types of Languages:

Procedural

1. Specifies a series of well-structured steps and procedures to compose a program
2. contains a systematic order of statements, functions and commands to complete a task

Functional

1. Writing a program only in pure functions i.e. Never modify variables, but only create new ones as an output.
2. used in situations where we have to perform lots of different operations on the same set of data.
3. They also first class functions, a function can be passed as an argument to other functions.

Object Oriented

1. The main aim of OOP is bind together the data and the functions that can operate on them, so that on part of code can the data except that function

Static

1. Performs type checking at compile time
2. Errors will be shown at compile time
3. Declare Datatypes before you use it
4. More Control

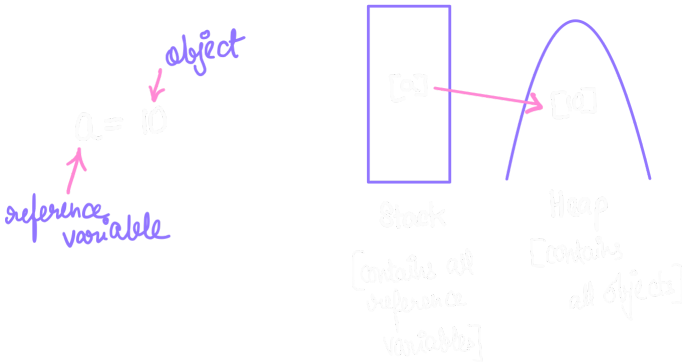
   E.g. int a = "name" { will give error }

Dynamic

1. Performs type checking at runtime.
2. Errors is not shown until the program is run
3. No need to declare datatypes of variables
4. Saves time in writing code but might give error at runtime

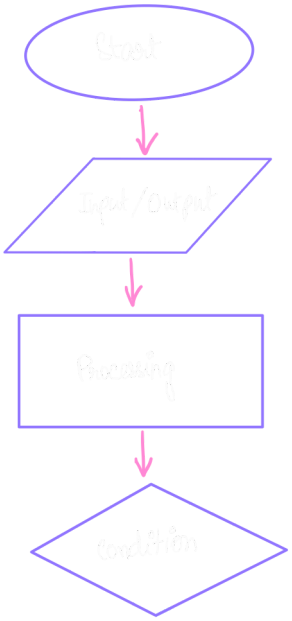
E.g. a = 10; a = " name " { will give no errors

Stack & Heap Memory



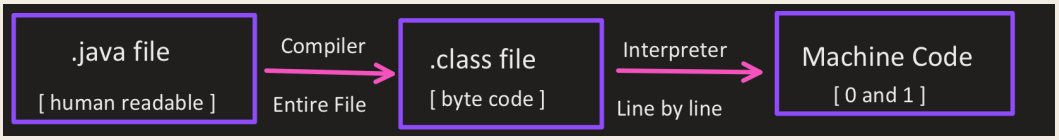
1. More than one reference variable can point to object
2. if one of the reference variables is changed then the object will be changed for all reference variables

Flow-Charts



02-first-java

Introduction to Java



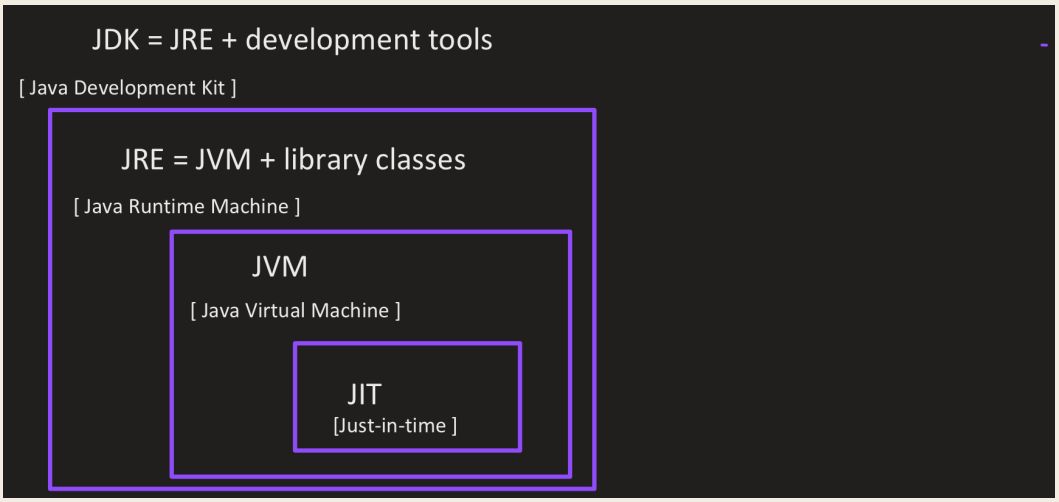
1. This code will not directly run on a system.
2. Only JVM (interpreter) can run the byte code
3. " Java is platform independent "

Platform Independence

1. In C & C++, the compiler converts source code to machine code (which is executable), which is .exe or .out file, making it platform dependent.

2. In Java, the compiler converts source code its byte-code (which will run on all platforms) . Different operating systems have different (Java Virtual Machine) JVM converts the byte code into machine code.

Java Architecture



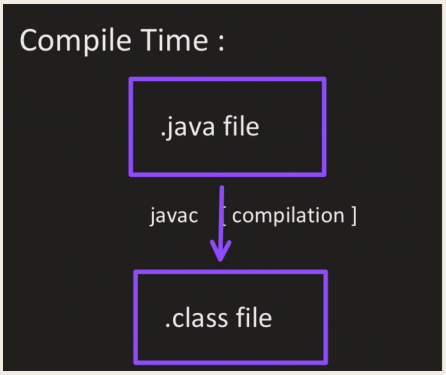
Java Development Kit (JDK) :

1. Development tools -> provides an environment to develop
2. JRE -> executes the program
3. Compiler (javac)
4. Archive (jar)
5. Interpreter / loader

Java Runtime Environment (JRE) :

1. It only provides an environment to only to run the program.
2. Class loader loads all classes (.class files) needed to execute the program.
3. It verifies the byte-code and check the format of code.
4. It consists of:
    a. Development Technologies
    b. User Interface Toolkits
    c. Integration Libraries
    d. Base Libraries
    e. JVM

How Java works ?



Class Loader :

1. Loading :

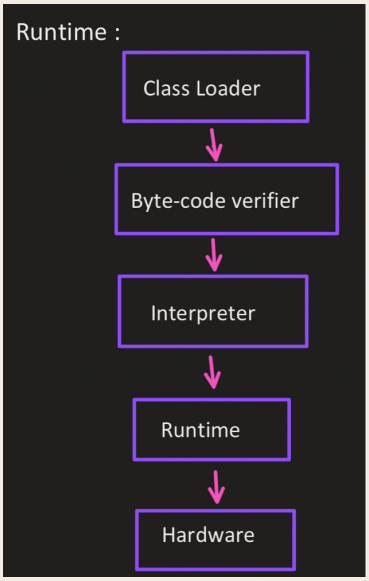
1. Reads .class file and generates binary code
2. An object of class is created in heap

2. Linking :

1. Searches for illegal practices in .class file.
2. Allocates memory for class variables and default variables.
3. Replace symbolic references from the type with direct references (used in functions)

3. Initialisation :

1. All static variables are assigned with their values defined in the code and static block.



JVM Execution :

1. Interpreter :

1. Line by line execution
2. When one method is called many times, then the interpreter will execute many times

2. JIT (Just-in-Time) :

1. It provides the direct machine code instead of re-interpretation of a method
2. Makes execution faster

3. Garbage Collection :

1. Removes object without reference variable from the heap memory

Basic Functions in Java

```java
package com.inclass;

public class Main {
```

```
    public static void main(String[] args) {
    System.out.println("Hey how are you?");
    }
}
```

- `package` shows the folder where the java files are present
- `public` allows the class to be accessed from anywhere
- `class` is name group of properties & functions
- `Main` is file name of the java file
- Generally to run the program java needs to make an object of "main' class, `static` helps java to run the program without making an object
- `main` is the entry point of the java program
- `String[] args` is an array of strings which allows the command line arguments to get stored
- `system` is class which consists of the basic functions

"System.java" documentation :

> - The `System` class contains several useful class fields and methods. It cannot be instantiated.
> - Among the facilities provided by the `System` class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

- `out` is ready to accept data and display as output

"out" function documentation :

> - The `standard` output stream. This stream is already open and ready to accept output data.
> - Typically this stream corresponds to display output or another output destination specified by the host environment or user.

- `println` prints a string and then terminates the line

"PrintStream.java" Documentation :

> - A `PrintStream` adds functionality to another output stream, namely the ability to print representations of various data values conveniently.
> - All characters printed by a `PrintStream` are converted into bytes using the given encoding or charset, or the platform's default character encoding if not specified. The `PrintWriter` class should be used in situations that require writing characters rather than bytes.

How to run a java program ?

```
driptanil  driptanil-Lenovo-V110-15ISK  ~  javac Main.java
```

Converts .java file to .class file (byte code)

```
driptanil  driptanil-Lenovo-V110-15ISK  ~  java Main
```

Runs the .class file

Scanner

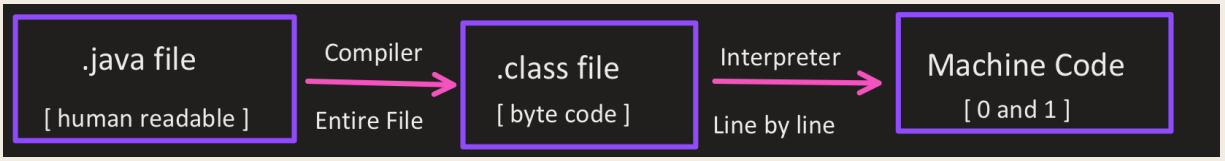
```
import java.util.Scanner;

Scanner input = new Scanner(System.in);
```

- `scanner` is a simple text scanner which can parse primitive types and Strings using regular expressions.

  `Scanner.java` Documentation :

  > - A simple text scanner which can parse primitive types and strings using regular expressions.
  > - `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches white-space. The resulting tokens may then be converted into values of different types using the various next methods.

- `new` creates a new instance of a function or a class
- `in` takes input from specially from keyboard or another source specified the user



`InputStream` function Documentation :

> - The `standard` input stream. This stream is already open and ready to supply input data. Typically this stream corresponds to keyboard input or another input source specified by the host environment or user

Datatypes

Primitives :

- The datatypes, which cannot be further splitted into array of elements. E.g. integer, float, character, Boolean datatypes etc.
- String is non primitive datatype

```
byte b = 42;
char c = 'a';
short s = 1024;
int i = 50000;
float f = 5.67f;
double d = 0.1234;
```

Typecasting :

Compressing a data type into a smaller data type explicitly.

```
int a = 100;
byte b = (byte) (a);

Output : b = 100
```

Automatic Type Promotion (in Java)

```
byte a = 40;
byte b = 50;
byte c = 100;
int d = a * b / c;

/* Output : d = 20 */
```

Instead of operation with byte, whole operation is promoted to integer type

Unicode

```
int num = 'A';
System.out.println("ASCII of A = " + num);

/* Output : ASCII of A = 65 */


System.out.println("こんにちは");

/* Output : こんにち */


byte b = 42;
char c = 'a';
short s = 1024;
int i = 50000;
float f = 5.67f;
double d = 0.1234;
double result = (f * b) + (i / c) - (d * s);
System.out.println("result = " + result);

/* Output : result = 626.77841464843375 */
```