Java

# ROAD-TO-NINJA

## Beginner (Part 1)

## New-Out-Of-Box (NOOB)

Organised by :

<Dev Con/>

Supported by :

MaGIC
Malaysian Global Innovation and Creativity Centre

# ABOUT ME

| | |
|---|---|
| Name : | **Mohd Azman Kudus** |
| Age : | 30 years |
| Java exp : | 7 years |

Question?

# AGENDA (1/2)

- **COMPUTER & SOFTWARE**
  - Computer : Hardware & Software
  - Studies : Science, Engineering and Technology

- **PROBLEM AND SOLUTION**
  - Analytical, Lateral and Team skills
  - IDEAL & SMART model

- **JAVA?**
  - Purpose,  history & case studies

- **HELLO WORLD**
  - Basic syntax
  - Java Virtual Machine (JVM)
  - Install & run

# AGENDA (2/2)

## CODE STRUCTURE

- Sequence
- Containers

## DATA

- Data types, literals and conversion
- Encapsulation

## OPERATION

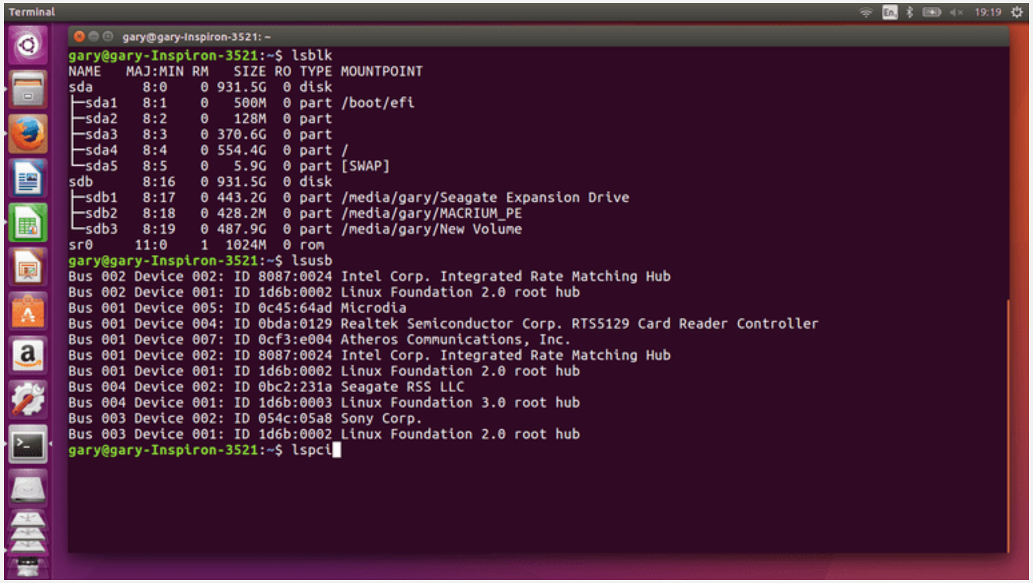
- Basic arithmetic
- Branching
- Repetition
- Input & Output

# COMPUTER

TO START
PRESS ANY KEY
"To start, press any key."
Where's the "any" key?
90s90s90s

# COMPUTER HARDWARE

# COMPUTER SOFTWARE

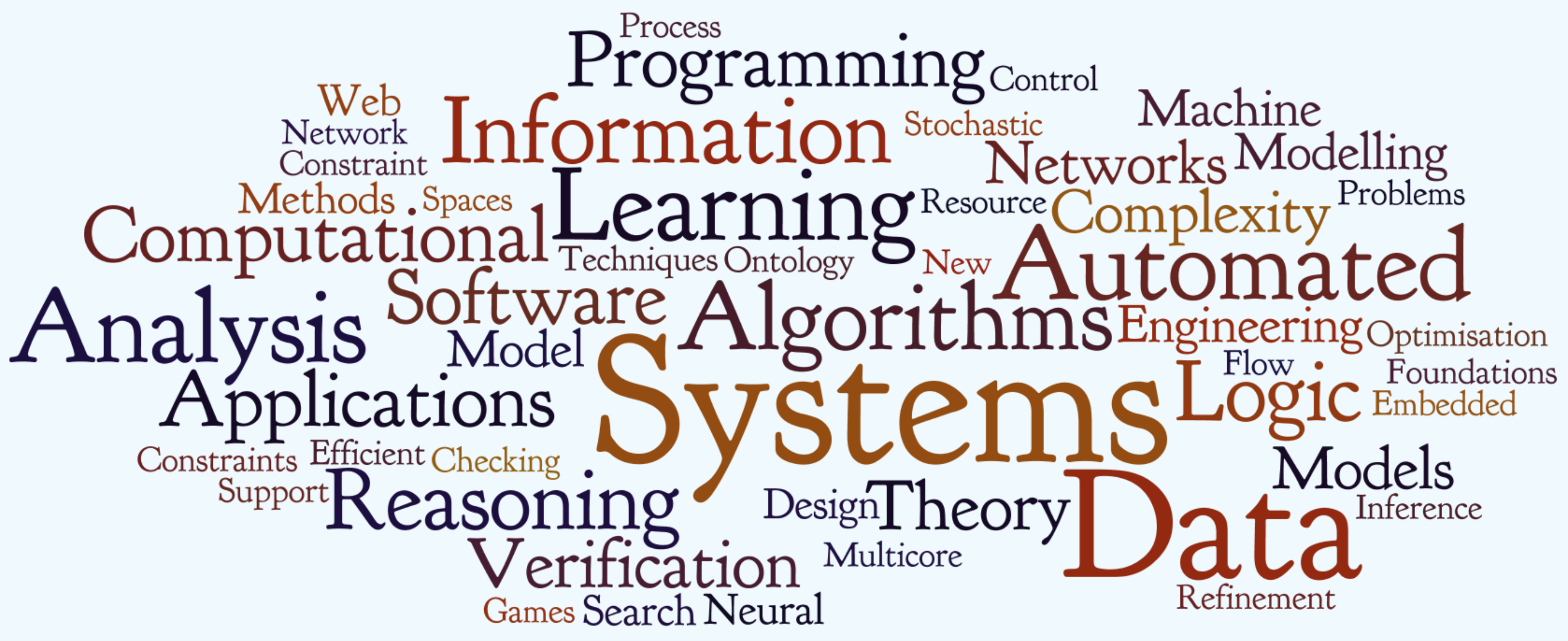


```
gary@gary-Inspiron-3521:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 931.5G  0 disk
├─sda1   8:1    0   500M  0 part /boot/efi
├─sda2   8:2    0   128M  0 part
├─sda3   8:3    0 370.6G  0 part
├─sda4   8:4    0 554.4G  0 part /
└─sda5   8:5    0   5.9G  0 part [SWAP]
sdb      8:16   0 931.5G  0 disk
├─sdb1   8:17   0 443.2G  0 part /media/gary/Seagate Expansion Drive
├─sdb2   8:18   0 428.2M  0 part /media/gary/MACRIUM_PE
└─sdb3   8:19   0 487.9G  0 part /media/gary/New Volume
sr0     11:0    1  1024M  0 rom
gary@gary-Inspiron-3521:~$ lsusb
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 005: ID 0c45:64ad Microdia
Bus 001 Device 004: ID 0bda:0129 Realtek Semiconductor Corp. RTS5129 Card Reader Controller
Bus 001 Device 007: ID 0cf3:e004 Atheros Communications, Inc.
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 002: ID 0bc2:231a Seagate RSS LLC
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 002: ID 054c:05a8 Sony Corp.
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
gary@gary-Inspiron-3521:~$ lspci
```

Every computing hardware requires power and software to perform it's task/role.

e.g.: BIOS, operating system, driver, business applications

# COMPUTER STUDIES (1/2)



# COMPUTER STUDIES (2/2)

- **COMPUTER SCIENCE**
  - Focus on discrete mathematics and computational applications.
  - Usually build algorithm or software as a solution to specific problems set.

- **COMPUTER ENGINEERING**
  - Focus on computer equipment, usually involve most of engineering study fields

    e.g. : civil, mechanical, electrical and electronic

- **INFORMATION TECHNOLOGY**
  - Focus on specific area of computing environment which does not involve any research or development of hardware or software.

    e.g. : Support, administration, networking, security

# PROBLEM?

# PROBLEM SOLVING SKILLS (1/2)

## ANALYTICAL THINKING

- Evaluate and make decision.
- Use logical and methodical approach.

## LATERAL THINKING

- Creative, out-of-box thinking
- Discard obvious, skip traditional thinking, ignore preconceptions.

## TEAM

- Key component in problem solving.
- Not necessary analytic/lateral

  e.g.: Management, communication, negotiation.

# PROBLEM SOLVING SKILLS (2/2)

*Innovation distinguishes between a leader and a follower.*

*Creativity is just connecting things. When you ask creative people how they did something, they feel a little guilty because they didn't really do it, they just saw something.*

*It seemed obvious to them after a while. That's because they were able to connect experiences they've had and synthesise new things.*

*Steve Jobs*

# IDEAL SOLUTION MODEL (1/6)

I Identify

D Define

E Explore

A Act

L Look

# IDEAL SOLUTION MODEL (2/6)

## I Identify issues

✓ Investigate causes until the root cause
✓ Gather relevant information
✓ Break problems into parts

# IDEAL SOLUTION MODEL (3/6)

## D Define goal

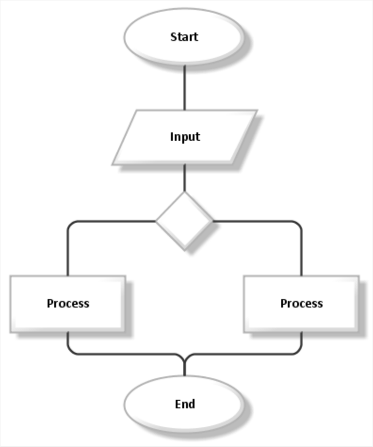
- ✓ Set target output or solution.
- ✓ Ideation process.
- ✓ SMART model

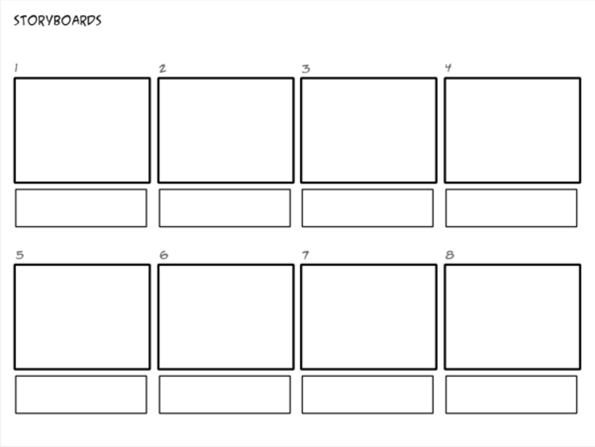
| | | |
|---|---|---|
| **S** | **Specific** | Goal must be clear |
| **M** | **Measurable** | Ability to track, stay focus and motivated |
| **A** | **Achievable** | Realistic and attainable. Sometimes will stretch abilities. |
| **R** | **Relevant** | Care on the progress. |
| **T** | **Time bound** | Target date and priority |

# IDEAL SOLUTION MODEL (4/6)

## E Explore options

- ✓ Explore and prepare solutions draft.
- ✓ Use presentation medium (flow chart, pseudo-code, story board)
- ✓ Decide on final and best solution.

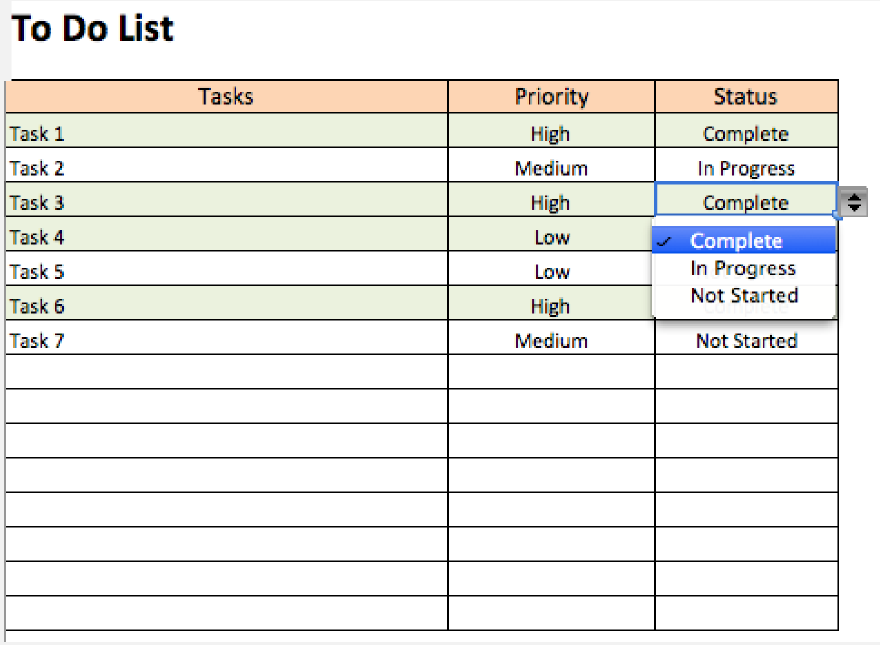




# IDEAL SOLUTION MODEL (5/6)

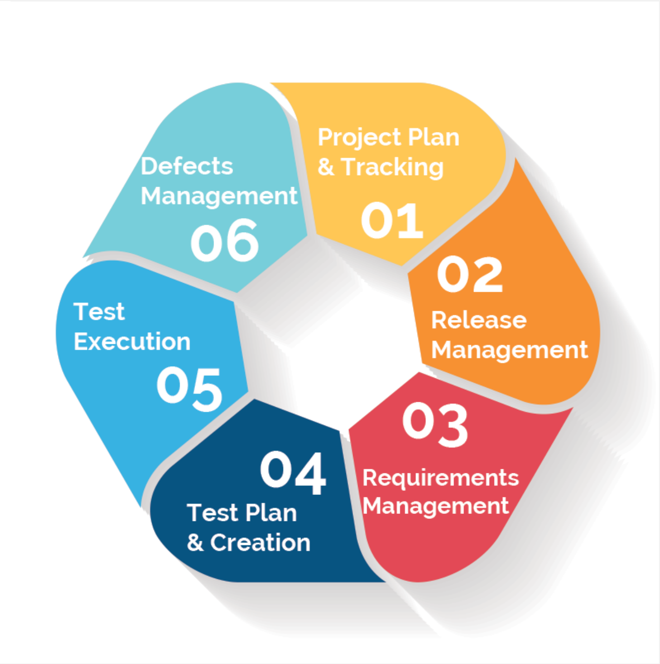
## A Act on best solution

- ✓ To-do list
- ✓ Build - Test - Repeat
- ✓ Progress monitoring



**To Do List**

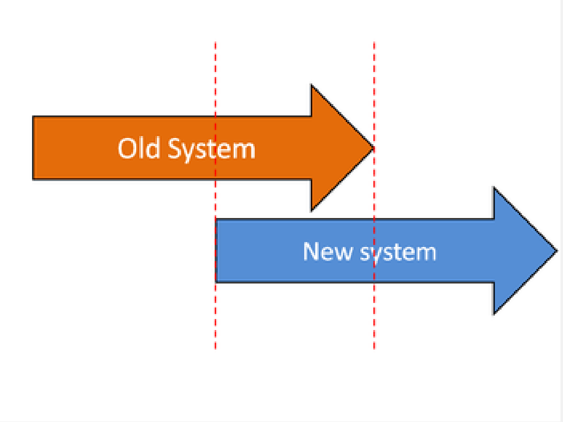
| Tasks | Priority | Status |
|---|---|---|
| Task 1 | High | Complete |
| Task 2 | Medium | In Progress |
| Task 3 | High | Complete |
| Task 4 | Low | |
| Task 5 | Low | |
| Task 6 | High | Complete |
| Task 7 | Medium | Not Started |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |



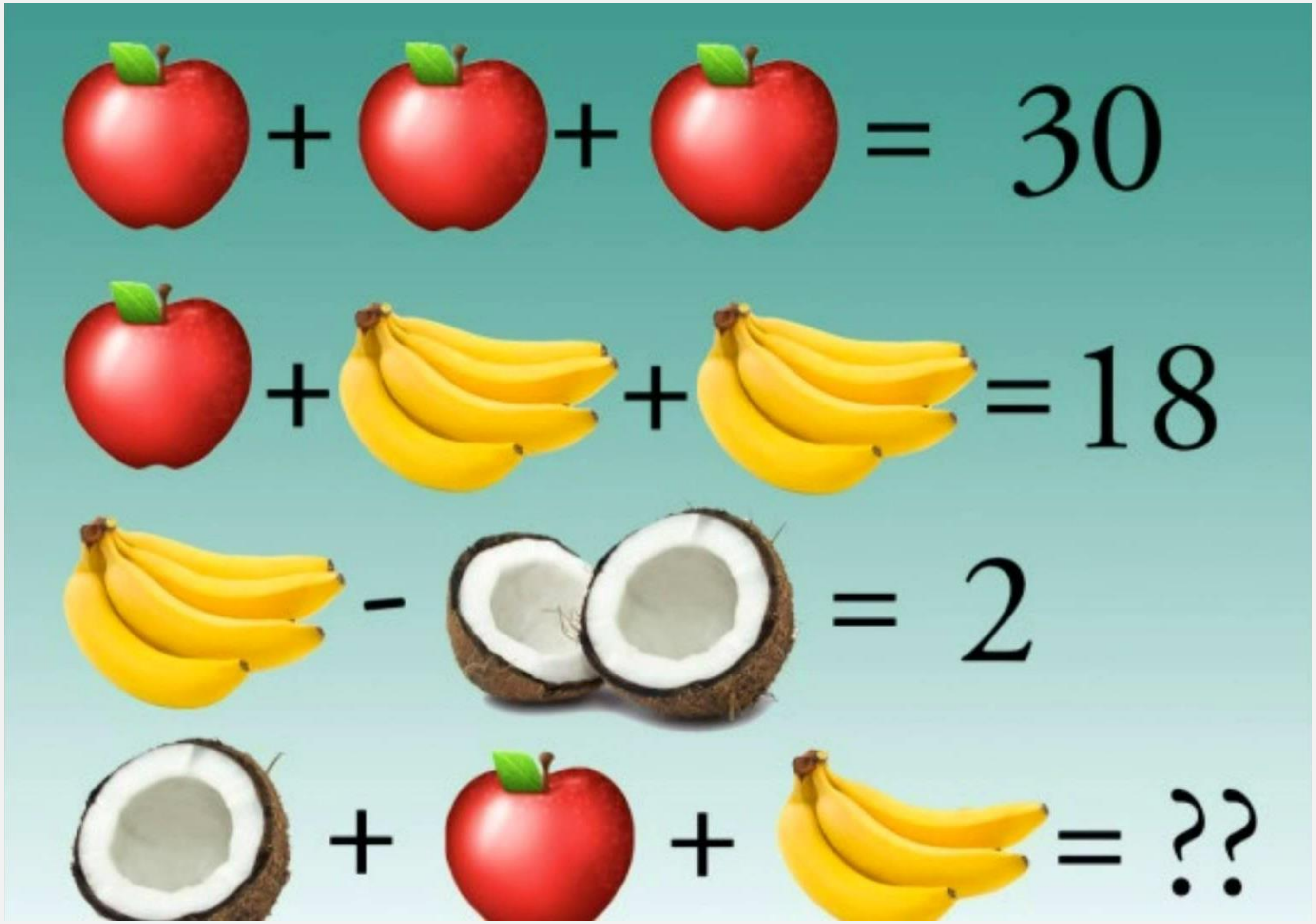
# IDEAL SOLUTION MODEL (6/6)

## L Look and learn

- ✓ Working smoothly?
- ✓ Improvement?



# EXERCISE (1/2)



# EXERCISE (2/2)

I

- 1 coconut + 1 apple + 3 bananas = ?
- 1 coconut = ?, 1 apple = ?, 1 banana ?

D

- Get value of each coconut, apple and banana

E

- Get apple value first? Or banana first? Or coconut first?

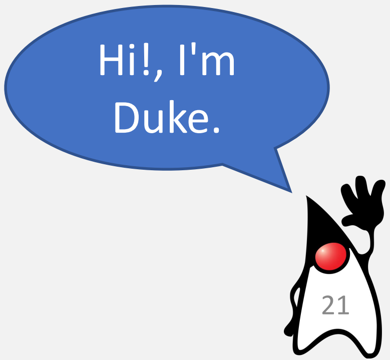
A

- Get value of an apple (Line 1), then banana (Line 2), then coconut (Line 3)
- Solve equation Line 4

L

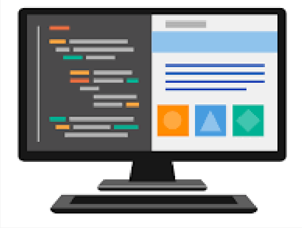
- Evaluate with other equation. Go deeper.

# JAVA : A CUP OF COFFEE



# IMPLEMENTATIONS

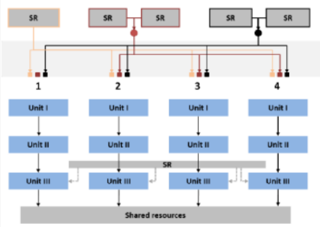
Web



Desktop

Mobile

Development tools



Batch

SmartCard

Science

Embedded

Database

Networking

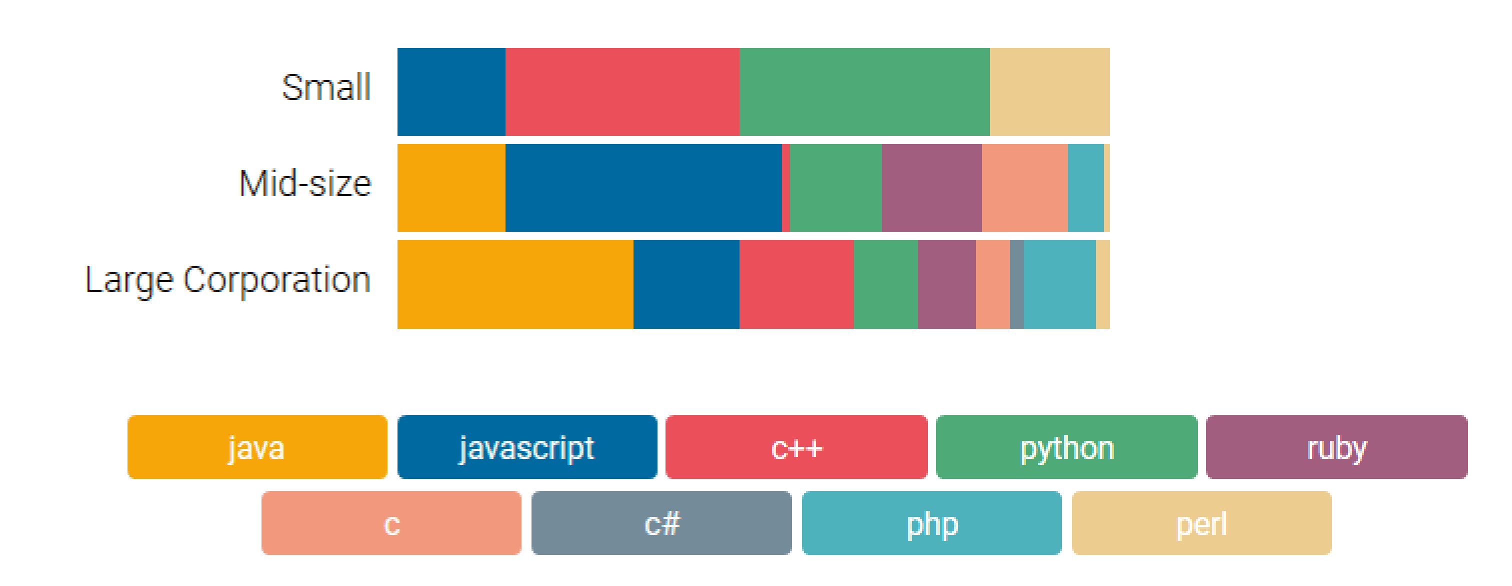
# HISTORY

The Green team

- Initially named **Oak** - Sun Microsystem set-top box (1991)
- Rename to **Java** after Oak Technology (1994)
- Versions :
  - Java Standard Edition (Java/J2SE/JavaSE)
  - Java Enterprise Edition (J2EE/JavaEE/JakartaEE)
  - Java Micro Edition (J2ME) (CDLC)
  - Java Card

Java 1.0a (1994)
Java 1.1 (1997)
J2SE 1.2 (1998)
J2EE 1.2 (1999)
J2SE 1.3 (2000)
J2EE 1.3 (2001)
J2SE 1.4 (2002)
J2EE 1.4 (2003)
JavaSE 5 (2004)
JavaEE 5 (2006)
JavaSE 6 (2006)
JavaSE 7 (2011)
JavaEE 7 (2013)
JavaSE 8 (2014)
**JavaEE 8 (2017)**
JavaSE 9 (2017)
**JavaSE 10 (2018)**

# COMPANIES



UBER

CISCO

ORACLE

IBM

intel

hp

Google

redhat

ECLIPSE FOUNDATION

JUNIPER NETWORKS

accenture

THE APACHE SOFTWARE FOUNDATION

ATLASSIAN

AZUL SYSTEMS

ZEROTURNAROUND

PHILIPS

amazon

LinkedIn

NETFLIX

Canon

ebay

Symantec

# GOAL (1/5)

## Simple

Can be programmed without extensive programmer training.
Grasped quickly and be productive from the very beginning.

## Object Oriented

Distributed, encapsulated, message-passing.

## Familiar

Very close to natural language, similar C++ look and feel.

# GOAL (2/5)

## Robust

Extensive compile-time checking, run-time checking.

No explicit defined pointer data types or arithmetic.

Automatic garbage collection.

## Secure

Security features designed and run-time system.

Can't be invaded from outside.

# Architecture Neutral

Execute on a variety of hardware architectures.
Compiler-to-Bytecode.

# Portable

Java Virtual Machine (JVM).
Install on OS or bundle with application.

# GOAL (4/5)

## High Performance

Interpreter can run at full speed without needing to check the run-time environment.

# GOAL (5/5)

## Interpreted

Execute Java bytecodes from successful compilation.

## Threaded

Multithreading which allow concurrency within single execution/process.

## Dynamic

On demand link even though strict compile-time static check.

# JAVA VIRTUAL MACHINE (JVM)

Run Java program
Run non-Java language then compile to Java bytecode

**Class loader**

- Load all classes which being use by a Java program
- Verify import
- Allocate memory
- Initialize classes and variables and invoke main class

**Just-In-Time compiler**

- Translate Java bytecode into machine language to speed up execution

**Heap**

- Memory area that allocated for direct memory location

**Implementation**

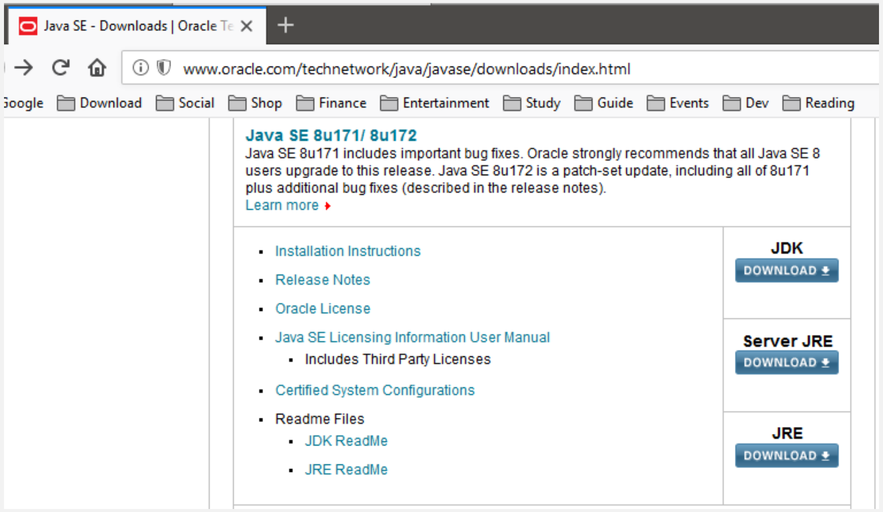
- HotSpot by Oracle
- OpenJ9 by Eclipse
- IcedTea (default OpenJDK)

# OTHER JVM LANGUAGES

# INSTALLATION (1/2)

Windows (32/64bit), MacOS, RedHat based linux

http://www.oracle.com/technetwork/java/javase/downloads/index.html


Java SE - Downloads | Oracle Te
www.oracle.com/technetwork/java/javase/downloads/index.html
Google
Download
Social
Shop
Finance
Entertainment
Study
Guide
Events
Dev
Reading

**Java SE 8u171/ 8u172**

Java SE 8u171 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release. Java SE 8u172 is a patch-set update, including all of 8u171 plus additional bug fixes (described in the release notes).

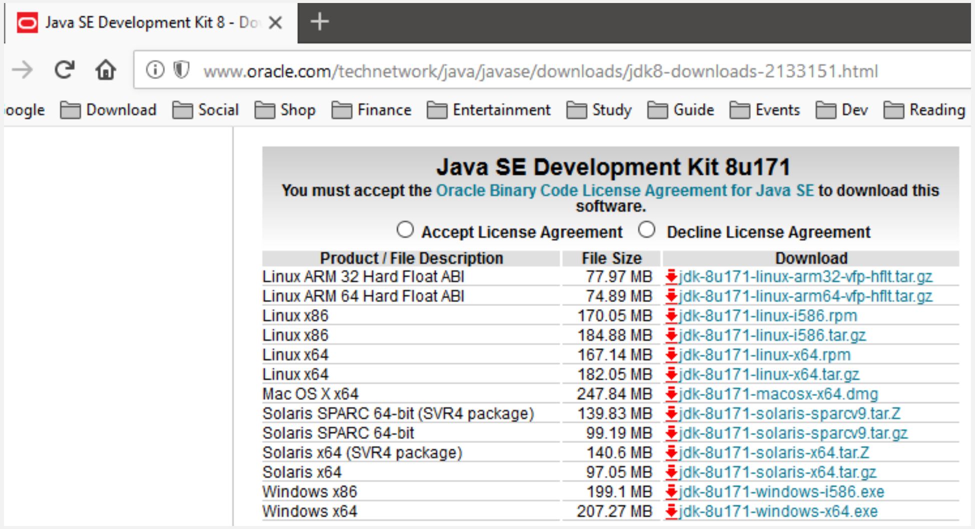
Learn more

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme Files
  - JDK ReadMe
  - JRE ReadMe

JDK DOWNLOAD

Server JRE DOWNLOAD

JRE DOWNLOAD


Java SE Development Kit 8 - Do
www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
oogle
Download
Social
Shop
Finance
Entertainment
Study
Guide
Events
Dev
Reading

**Java SE Development Kit 8u171**

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ○ Decline License Agreement

| Product / File Description | File Size | Download |
|---|---|---|
| Linux ARM 32 Hard Float ABI | 77.97 MB | jdk-8u171-linux-arm32-vfp-hflt.tar.gz |
| Linux ARM 64 Hard Float ABI | 74.89 MB | jdk-8u171-linux-arm64-vfp-hflt.tar.gz |
| Linux x86 | 170.05 MB | jdk-8u171-linux-i586.rpm |
| Linux x86 | 184.88 MB | jdk-8u171-linux-i586.tar.gz |
| Linux x64 | 167.14 MB | jdk-8u171-linux-x64.rpm |
| Linux x64 | 182.05 MB | jdk-8u171-linux-x64.tar.gz |
| Mac OS X x64 | 247.84 MB | jdk-8u171-macosx-x64.dmg |
| Solaris SPARC 64-bit (SVR4 package) | 139.83 MB | jdk-8u171-solaris-sparcv9.tar.Z |
| Solaris SPARC 64-bit | 99.19 MB | jdk-8u171-solaris-sparcv9.tar.gz |
| Solaris x64 (SVR4 package) | 140.6 MB | jdk-8u171-solaris-x64.tar.Z |
| Solaris x64 | 97.05 MB | jdk-8u171-solaris-x64.tar.gz |
| Windows x86 | 199.1 MB | jdk-8u171-windows-i586.exe |
| Windows x64 | 207.27 MB | jdk-8u171-windows-x64.exe |

Ubuntu Linux

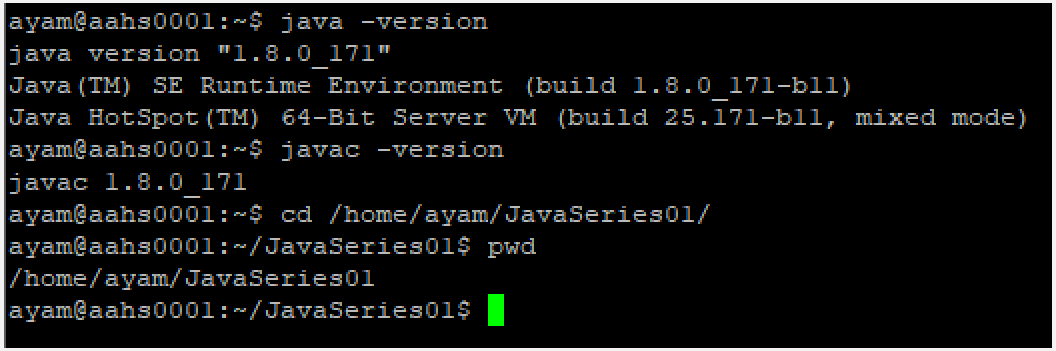
```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

# INSTALLATION (2/2)

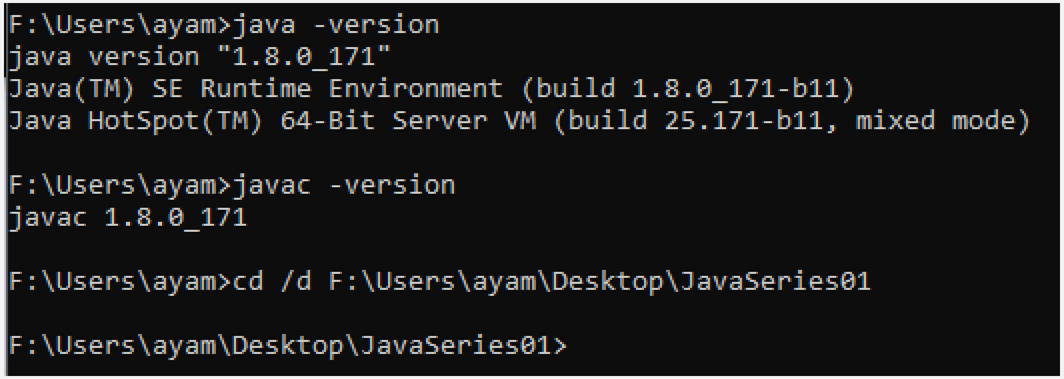
1. Open terminal / command prompt
   **`java -version`**
   **`javac -version`**

2. Create new folder from file explorer/manager

3. Change directory to folder from step (2)
   Windows: **`cd /d <folder_path>`**
   Others: **`cd <folder_path>`**



```
ayam@aahs0001:~$ java -version
java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)
ayam@aahs0001:~$ javac -version
javac 1.8.0_171
ayam@aahs0001:~$ cd /home/ayam/JavaSeries01/
ayam@aahs0001:~/JavaSeries01$ pwd
/home/ayam/JavaSeries01
ayam@aahs0001:~/JavaSeries01$
```



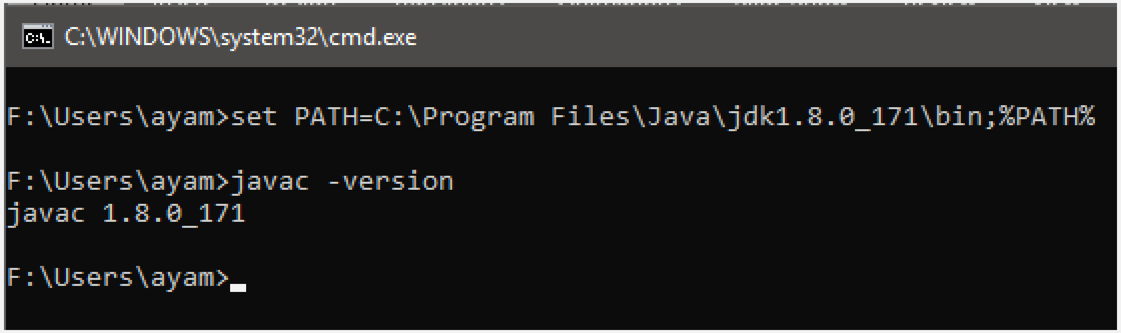
```
F:\Users\ayam>java -version
java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)

F:\Users\ayam>javac -version
javac 1.8.0_171

F:\Users\ayam>cd /d F:\Users\ayam\Desktop\JavaSeries01

F:\Users\ayam\Desktop\JavaSeries01>
```

# INSTALLATION (2/2)

Error with javac on windows,

**'javac' is not recognized as an internal or external command**

```
set PATH=C:\Program Files\Java\jdk1.8.0_171\bin;%PATH%
```



C:\WINDOWS\system32\cmd.exe

```
F:\Users\ayam>set PATH=C:\Program Files\Java\jdk1.8.0_171\bin;%PATH%

F:\Users\ayam>javac -version
javac 1.8.0_171

F:\Users\ayam>_
```

# GENERAL SYNTAX

; {} ()

1. Statement must ends with semicolon.
2. Class must be enclosed with curly braces.
3. Branching/repetition must enclosed with curly braces except for single line.
4. Method with or without arguments must be enclosed with round braces.

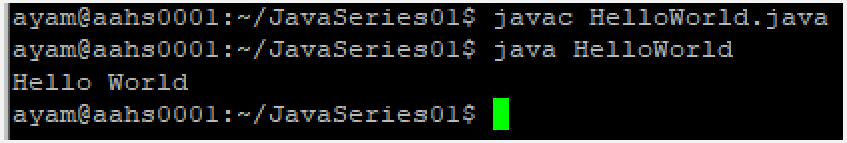
# HELLO WORLD

1. Open text editor
2. Type the following code

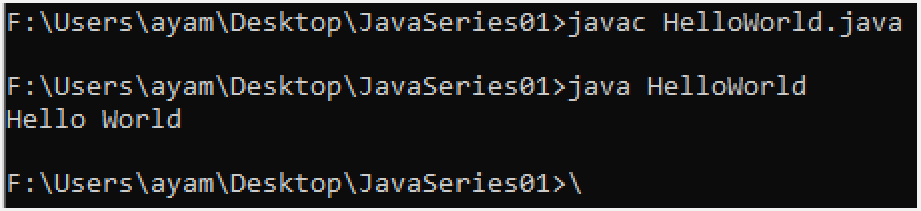
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

3. Save as HelloWorld.java
4. Go back to terminal/command prompt

```
javac HelloWorld.java
java HelloWorld
```

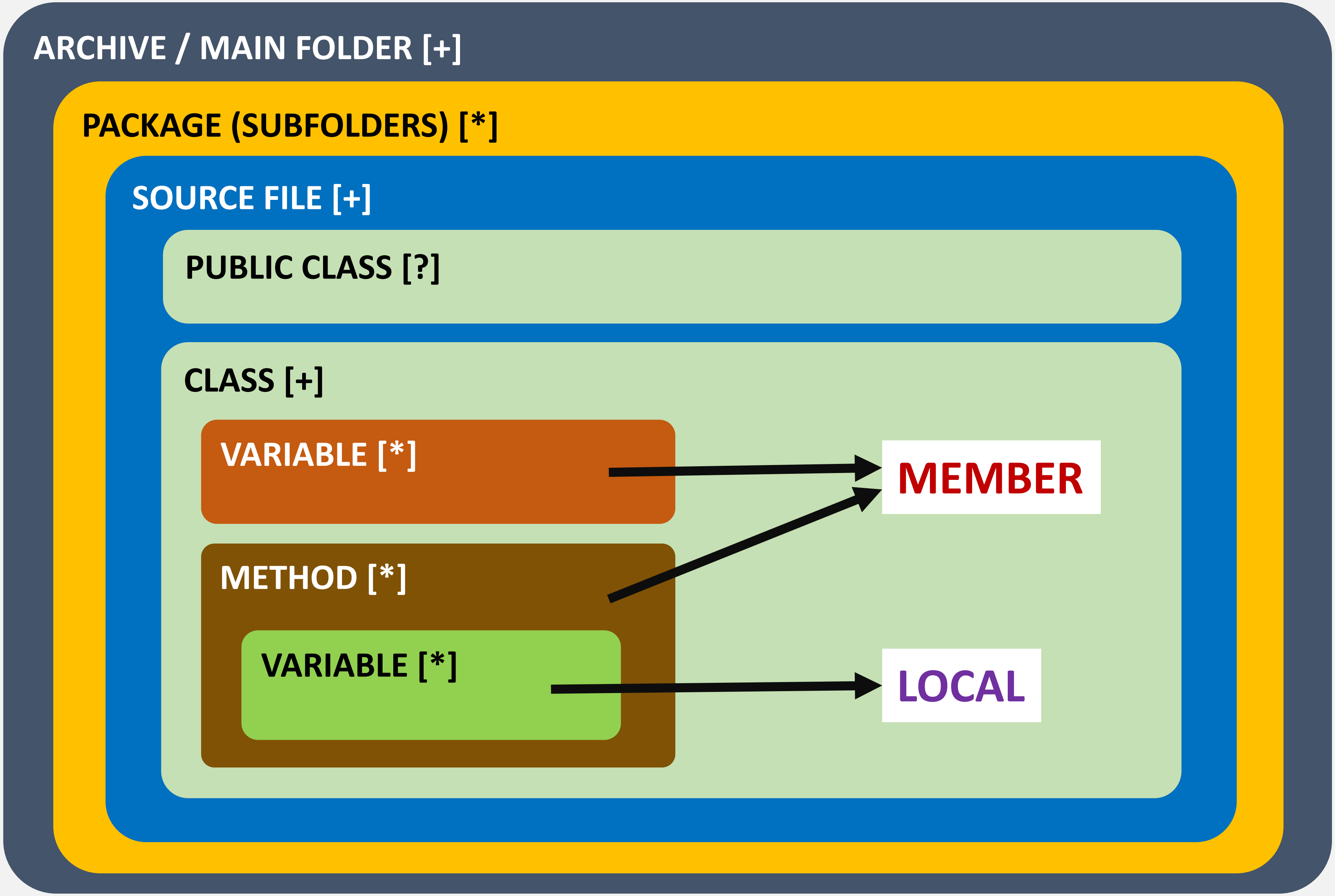


```
ayam@aahs0001:~/JavaSeries01$ javac HelloWorld.java
ayam@aahs0001:~/JavaSeries01$ java HelloWorld
Hello World
ayam@aahs0001:~/JavaSeries01$
```



```
F:\Users\ayam\Desktop\JavaSeries01>javac HelloWorld.java

F:\Users\ayam\Desktop\JavaSeries01>java HelloWorld
Hello World

F:\Users\ayam\Desktop\JavaSeries01>\
```

# CONSTRUCTION (1/3)



ARCHIVE / MAIN FOLDER [+]

PACKAGE (SUBFOLDERS) [*]

SOURCE FILE [+]

PUBLIC CLASS [?]

CLASS [+]

VARIABLE [*]

METHOD [*]

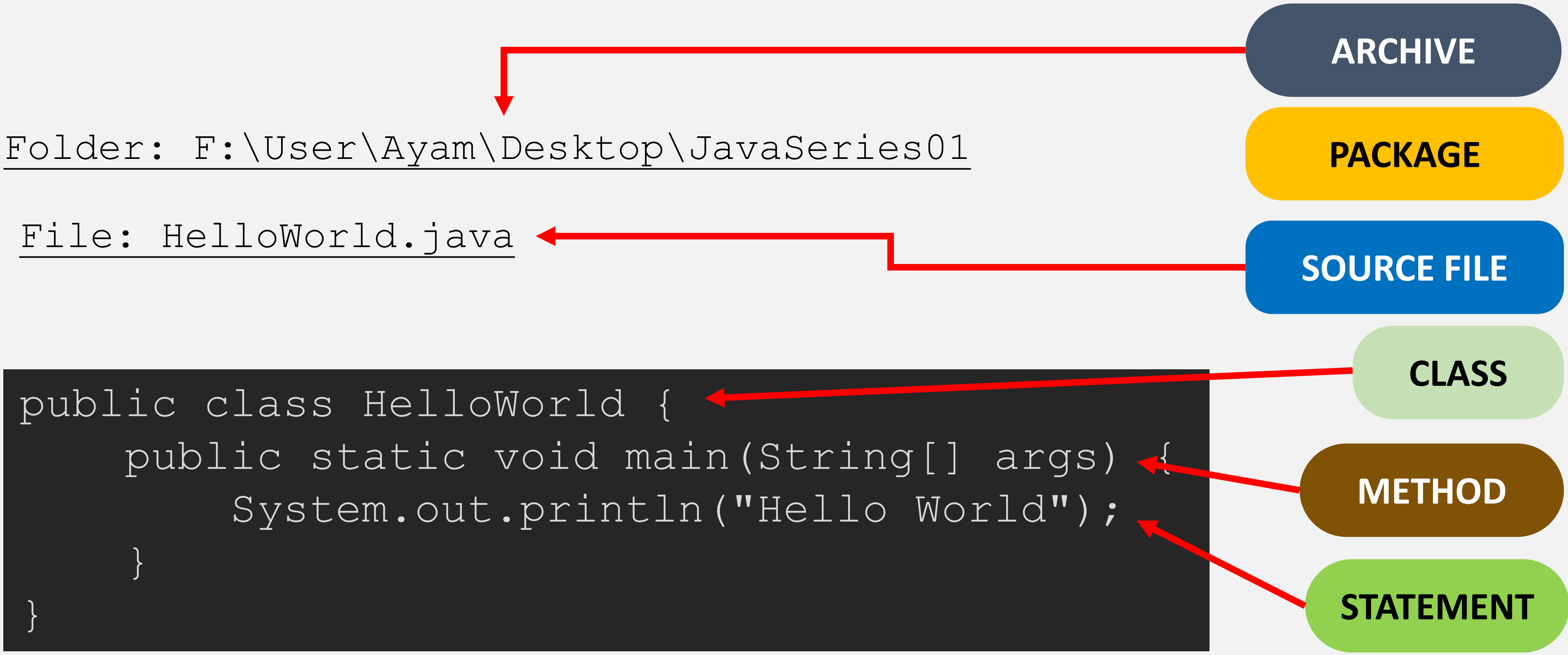
VARIABLE [*]

MEMBER

LOCAL

[+] Once or many [*] None or many [?] None or once

# CONSTRUCTION (2/3)



Folder: F:\User\Ayam\Desktop\JavaSeries01

File: HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

# CONSTRUCTION (3/3)

```java
package hello;                                       // file placement
import java.util.Date;                               // library import
public class HelloWorld {                            // class declaration
    private static final String MESSAGE = "Hello ";  // constant
    private String text;                             // member variable
    public HelloWorld() {                            // default constructor
        text = "World";
    }
    public HelloWorld(String text) {                 // normal constructor
        this.text = text;
    }
    public void setText(String text) {               // setter
        this.text = text;
    }
    public String getText() {                        // getter
        return text;
    }
    public void print() {                            // procedure
        System.out.println(MESSAGE + text);
    }
    public static void main(String[] args) {         // main method
        HelloWorld hello = new HelloWorld();         // new instance
        hello.print();                               // method calls
        hello.setText("Ayam");
        hello.print();
    }
```

# VARIABLE

**<scope> <modifier> <datatype> <name> = <value_or_literal>;**

```
public class Variable1 {
    public static void main(String[] args) {
        String message = "Hello World";
        System.out.println(message);
    }
}
```

```
public class Variable2 {
    private static String message = "Hello World";

    public static void main(String[] args) {
        System.out.println(message);
    }
}
```

# BASIC DATA TYPE (1/2)

## Primitive (non-null)

| Type | Detail | Default | Range | Example |
|---|---|---|---|---|
| **byte** | 8-bit signed integer | 0 | -128 ... 127 | -123, 0, 123 |
| **short** | 16-bit signed integer | 0 | -32768 ... 32767 | -123, 0, 123 |
| **int** | 32-bit signed integer | 0 | $-2^{31}$ ... $2^{31}-1$ | -123, 0, 123 |
| **long** | 64-bit signed integer | 0 | $-2^{63}$ ... $2^{63}-1$ | -123L, 0L, 123L |
| **float** | 32-bit IEEE 754 floating point | 0 | | -1.23f, 0.0f, 1.23f |
| **double** | 64-bit IEEE 754 floating point | 0 | | -1.23, 0.0, 1.23<br>-1.23d, 0.0d, 1.23d |
| **char** | 16-bit Unicode character | \u0000 | | 'a', '\n' '\u00FF' |
| **boolean** | 1-bit flag | false | true or false | true |

## String

- Character sequence, maximum size up to allocated memory
- Default is "null"

# BASIC DATA TYPE (2/2)

## Print the following variables and values

```
byte by = 1;

short sh = 20;

int in = 300;

long lo = 4000;

float fl = 1.23;

double do = 456.789;

char ch = 'a';

boolean bo = true;

String st = "Hello";
```

# LITERALS (1/2)

## Numeric literals

| Type | Example |
|---|---|
| Binary | `0b10100101` |
| Octal | `0123` |
| Decimal | `123` |
| Hexadecimal | `0x10AF` |
| Exponential | `1.234e5` |

## Character / String literal

| Type | Example |
|---|---|
| Unicode (up to UTF-16) | `\u0010` |
| Backspace | `\b` |
| Tab | `\t` |
| Line feed | `\n` |
| Carriage return | `\r` |
| Form feed | `\f` |
| Double quote | `\"` |
| Single quote | `\'` |
| Backslash | `\\` |

# LITERALS (2/2)

## Print the following variables and values

```java
byte bin = 0b1010;

short oct = 0789;

int hex = 0x12CD;

float exp = 1.23e2;

String uc = "Hello\u007FWorld";

String lf = "Hello\nWorld";

String cr = "Hello\rWorld";

String tb = "Hello\tWorld";

String bs = "Hello\bWorld";

String dq = "\"Hello World\"";

String sq = "\'Hello World\'";
```

# WRAPPER (1/2)

- Allow null values
- Provide basic methods for value manipulation
- Auto conversion
  - Autoboxing = Primitive to wrapper
  - Unboxing = Wrapper to primitive

| Primitive | Wrapper |
|---|---|
| **`byte`** | `Byte` |
| **`short`** | `Short` |
| **`int`** | `Integer` |
| **`long`** | `Long` |
| **`float`** | `Float` |
| **`double`** | `Double` |
| **`char`** | `Character` |
| **`boolean`** | `Boolean` |

# WRAPPER (2/2)

## Print the following variables and values

```
byte by1 = 1;
Byte by2 = null;
by2 = by1
byte by3 = by2;

char ch1 = 'a';
Character ch2 = null;
ch2 = ch1
char ch3 = ch2;
```

# DATA CONVERSION (1/3)

**String to primitive**

```
byte a = Byte.parseByte("1");
short b = Short.parseShort("1");
int c = Integer.parseInt("1");
long d = Long.parseLong("1");
float = Float.parseFloat("1.0");
double = Double.parseDouble("1.0");
```

**Primitive to String (Decimal**

```
String a10 = Byte.toString(a);
String b10 = Short.toString(b);
String c10 = Integer.toString(c);
String d10 = Long.toString(d);
String e10 = Float.toString(e);
String f10 = Double.toString(f);
```

# DATA CONVERSION (2/3)

**Primitive to String (Binary)**

```
String c2 = Integer.toBinaryString(c);
String d2 = Long.toBinaryString(d);
```

**Primitive to String (Octal)**

```
String c8 = Integer.toOctalString(c);
String d8 = Long.toOctalString(d);
```

**Primitive to String (Hexadecimal)**

```
String c16 = Integer.toHexString(c);
String d16 = Long.toHexString(d);
String e16 = Float.toHexString(e);
String f16 = Double.toHexString(f);
```

# DATA CONVERSION (3/3)

```
Print 10 + 100 in binary
Print 11 + 111 in octal
Print 12 + 122 in hexadecimal
```

# ARRAY

- Single variable with multi values
- Fixed size

```
public class Array1 {
    public static void main(String[] args) {
        int[] array = new int[3];
        array[0] = 1;
        array[1] = 4;
        array[2] = 9;
        System.out.println(array[0]);
        System.out.println(array[1]);
        System.out.println(array[2]);
    }
}
```

```
public class Array2 {
    public static void main(String[] args) {
        int[] array = new int[] { 1, 4, 9 };
        System.out.println(array[0]);
        System.out.println(array[1]);
        System.out.println(array[2]);
    }
}
```

# METHOD (1/6)

```
<scope> <return_datatype> <name> (<arguments>) <exception_throws> {
    // method body
}
```

```
arguments = <data_type> <name> [, <data_type> <name>]
exception_throws (optional) = throws <exception> [, <exception>]
```

```
public class Method1 {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
public class Method2 {
    public static void main(String[] args) {
        String message = "Hello World";
        print(message);
    }
    private static void print(String message) {
        System.out.println(message);
    }
}
```

# METHOD (2/6)

❑ **Main - Execution entry point**

```
public class MainMethod {
    public static void main(String[] args) {
        System.out.println(ClassTwo.two);
    }
}
```

# METHOD (3/6)

- **Procedure - Execute without returning any value.**

```
public class Procedure1 {
    public static void main(String[] args) {
        print();
    }
    private static void print() {
        System.out.println("Hello World");
    }
}
```

```
public class Procedure2 {
    public static void main(String[] args) {
        print("Hello World");
    }
    private static void print(String message) {
        System.out.println(message);
    }
}
```

# METHOD (4/6)

❑ **Function - Execute with returning value.**

```java
public class Function1 {
    public static void main(String[] args) {
        System.out.println(message());
    }
    private static String message() {
        return "Hello World";
    }
}
```

```java
public class Function2 {
    public static void main(String[] args) {
        System.out.println(message("World"));
    }
    private static String message(String message) {
        return "Hello " + message;
    }
}
```

# METHOD (5/6)

❑ **Constructor - To create an instance of an object**
**Default constructor : No arguments**

```
public class DefaultConstructor {
    private String message;

    public DefaultConstructor() {
        message = "Hello World";
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public static void main(String[] args) {
        DefaultConstructor dc = new DefaultConstructor();
        System.out.println(dc.getMessage());
        dc.setMessage("Ayam Goreng");
        System.out.println(dc.getMessage());
    }
}
```

# METHOD (6/6)

## Normal constructor : No arguments

```
public class NormalConstructor {
    private String message;

    public NormalConstructor(String message) {
        this.message = message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public static void main(String[] args) {
        DefaultConstructor dc = new DefaultConstructor();
        System.out.println(dc.getMessage());
        dc.setMessage("Ayam Goreng");
        System.out.println(dc.getMessage());
    }
}
```

# BASIC ENCAPSULATION (1/5)

- Control visibility or accessibility of a class/variable/method

**Public - everyone can access**

```
public class ClassOne {
    public static void main(String[] args) {
        System.out.println(ClassTwo.two);
    }
}
```

```
public class ClassTwo {
    public static int two = 2;
}
```

# BASIC ENCAPSULATION (2/5)

## Private - only member can access

```java
public class ClassOne {
    private static int one = 1;

    public static void main(String[] args) {
        System.out.println(one);
        System.out.println(ClassTwo.two);
    }
}
```

```java
public class ClassTwo {
    private static int two = 2;
}
```

# BASIC ENCAPSULATION (3/5)

## Default - only classes within same package can access

```
public class ClassOne {
    public static void main(String[] args) {
        System.out.println(ClassTwo.two);
    }
}
```

```
public class ClassTwo {
    static int two = 2;
}
```

# BASIC ENCAPSULATION (4/5)

## Static - share value along runtime

```
public class ClassOne {
    public static void main(String[] args) {
        System.out.println(ClassTwo.two);
    }
}
```

```
public class ClassTwo {
    public static int two = 2;
}
```

# BASIC ENCAPSULATION (5/5)

**Final - define once along runtime, cannot change (member variable or within constructor)**

```
public class ClassOne {
    private static final String MESSAGE = "Hello World";

    public static void main(String[] args) {
        System.out.println(MESSAGE);
    }
}
```

# BASIC ARITHMETIC OPERATION (1/5)

## Addition

```
public class Add {
    public static void main(String[] args) {
        int a = 10 + 20;
        System.out.println(a);
    }
}
```

## Subtraction

```
public class Subtract {
    public static void main(String[] args) {
        int a = 10 - 20;
        System.out.println(a);
    }
}
```

# BASIC ARITHMETIC OPERATION (2/5)

## Multiplication

```
public class Multiply {
    public static void main(String[] args) {
        int a = 10 * 20;
        System.out.println(a);
    }
}
```

## Division

```
public class Divide {
    public static void main(String[] args) {
        int a = 10 / 20;
        System.out.println(a);
    }
}
```

# BASIC ARITHMETIC OPERATION (3/5)

## Modulus / Balance

```
public class Modulus {
    public static void main(String[] args) {
        int a = 10 % 20;
        System.out.println(a);
    }
}
```

## Greater

```
public class Greater {
    public static void main(String[] args) {
        boolean a = 10 > 20;
        System.out.println(a);
    }
}
```

# BASIC ARITHMETIC OPERATION (4/5)

## Less

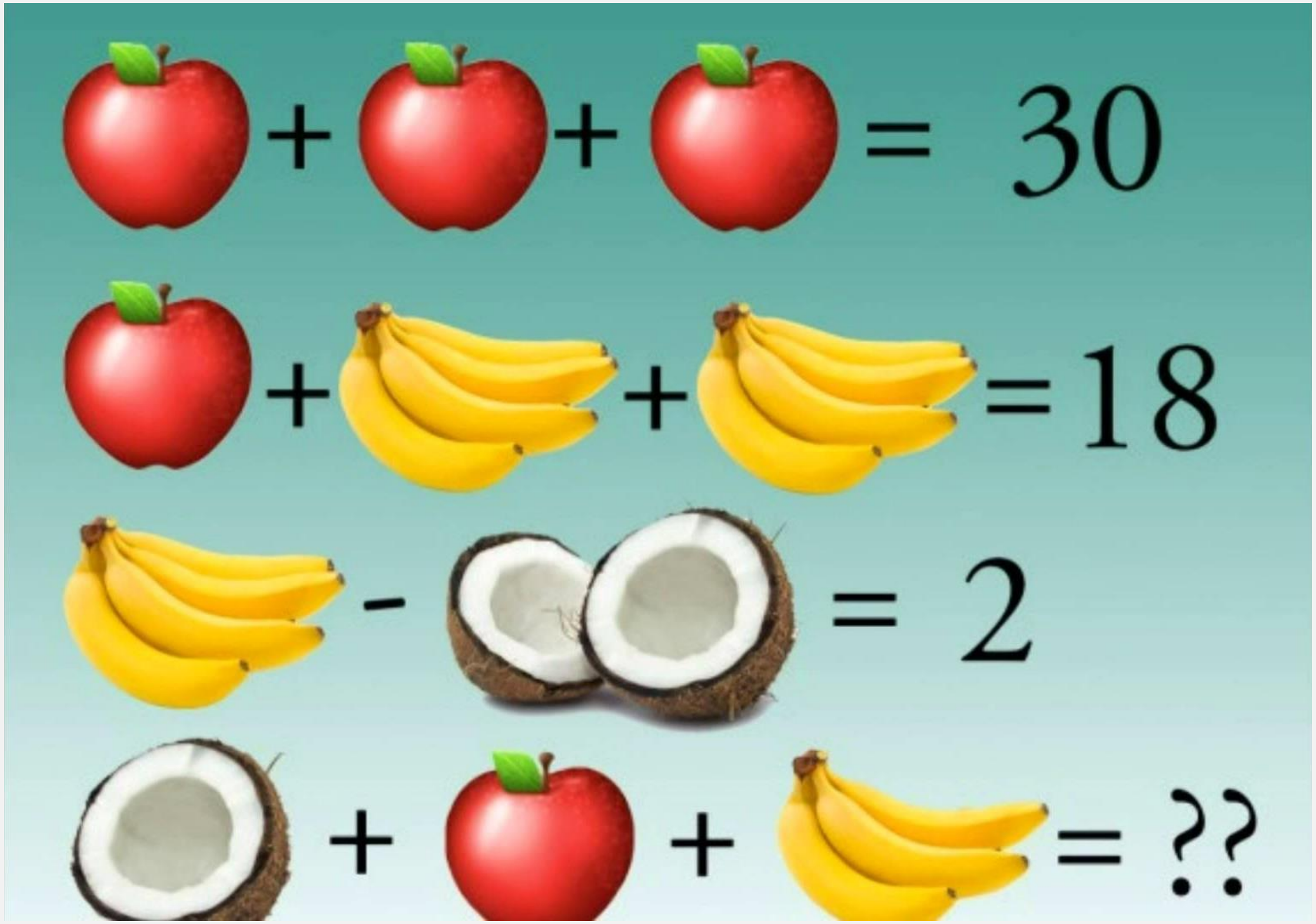
```
public class Less {
    public static void main(String[] args) {
        boolean a = 10 < 20;
        System.out.println(a);
    }
}
```

## Equal

```
public class Equal {
    public static void main(String[] args) {
        boolean a = 10 == 20;
        System.out.println(a);
    }
}
```

# BASIC ARITHMETIC OPERATION (5/5)

**Write and solve the following equation**

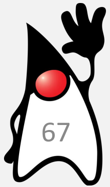


# CONTROL (1/9)

**Control statement must contains expressions in boolean value.**
**Directly reference to boolean variable or evaluation.**

## IF : single expression without fallback

```
public class If {
    public static void main(String[] args) {
        int a = 10;
        if (a > 5) {
            System.out.println("OK");
        }
    }
}
```



# CONTROL (2/9)

## IF-ELSE : single expression with fallback

```java
public class IfElse {
    public static void main(String[] args) {
        int a = 10;
        if (a > 5) {
            System.out.println("OK");
        }
        else {
            System.out.println("ERROR");
        }
    }
}
```

# CONTROL (3/9)

## IF-ELSE-IF : multi expressions without fallback

```
public class IfElseIf {
    public static void main(String[] args) {
        int a = 3;
        if (a > 5) {
            System.out.println("OK");
        }
        else if (a < 4) {
            System.out.println("ERROR");
        }
    }
}
```

# CONTROL (4/9)

## IF-ELSE-IF-ELSE : multi expressions with fallback

```
public class IfElseIfElse {
    public static void main(String[] args) {
        int a = 3;
        if (a > 5) {
            System.out.println("OK");
        }
        else if (a < 4) {
            System.out.println("ERROR");
        }
        else {
            System.out.println("WARNING");
        }
    }
}
```

# CONTROL (5/9)

## Multi expressions in single statement

## OR - Any one expression satisfied then execute

```java
public class Or {
    public static void main(String[] args) {
        int a = 7;
        if (a > 5 || a < 3) {
            System.out.println("OK");
        }
    }
}
```

# CONTROL (6/9)

## AND - All expressions satisfied then execute

```
public class And {
    public static void main(String[] args) {
        int a = 7;
        if (a > 5 && a < 10) {
            System.out.println("OK");
        }
    }
}
```

## NOT - Expression not satisfied then execute

```
public class Not {
    public static void main(String[] args) {
        int a = 7;
        if (!(a > 10)) {
            System.out.println("OK");
        }
    }
}
```

# CONTROL (7/9)

## Evaluate the following expression with sample inputs

```
print "A" if number less than 5
print "B" if number more than 20
print "C" if number is either 7 or 10
print "D" if number is between 14 and 18 (inclusive)
Otherwise print "E"
```

**Inputs:**

- **1**
- **5**
- **10**
- **11**
- **15**

# CONTROL (8/9)

## SWITCH : Case matching

```
public class Switch {
    public static void main(String[] args) {
        int a = 3;
        switch (a) {
            case 1: System.out.println("one");
                    break;
            case 2: System.out.println("two");
                    break;
            case 3: System.out.println("three");
                    break;
            case 4: System.out.println("four");
                    break;
            case 5: System.out.println("five");
                    break;
            default: System.out.println("ERROR");
        }
    }
}
```

# CONTROL (9/9)

## Evaluate the following expression with sample inputs

```
if character is 'a' then print "Apple"
if character is 'b' then print "Boy"
if character is 'c' or 'd' then print "Cat Duck"
otherwise print "Nothing"
```

**Inputs:**

- **a**
- **b**
- **c**
- **d**
- **e**

# REPITITION (1/5)

## WHILE : repeat until expression failed

```
public class While1 {
    public static void main(String[] args) {
        int a = 1;
        while (a < 10) {
            System.out.println(a);
            a++;                    // a = a + 1;
        }
    }
}
```

```
public class While2 {
    public static void main(String[] args) {
        int a = 10;
        while (a > 0) {
            System.out.println(a);
            a--;                    // a = a - 1;
        }
    }
}
```

# REPITITION (2/5)

## DO-WHILE : run once then repeat until expression failed

```
public class DoWhile {
    public static void main(String[] args) {
        int a = 1;
        do {
            System.out.println(a);
            a++;                   // a = a + 1;
        }
        while (a < 10);
    }
}
```

# REPITITION (3/5)

**FOR : Run until out of bound**

- **Initialization**
- **Termination**
- **Increment**

```
public class For {
    public static void main(String[] args) {
        for (int i = 1; i < 10; i++) {
            System.out.println(i);
        }
    }
}
```

# REPITITION (4/5)

## FOR-EACH : Run until no more element within array/collection

```
public class ForEach {
    public static void main(String[] args) {
        int[] array = new int[] {1,2,3};
        for (int i : array) {
            System.out.println(i);
        }
    }
}
```

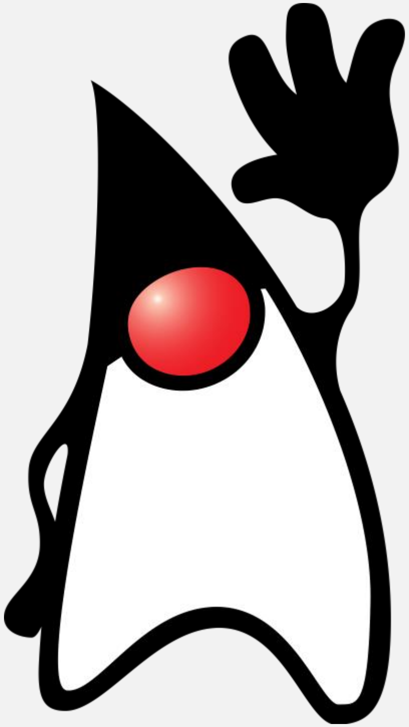
# REPITITION (5/5)

1. Print from 100 to 111 using while loop
2. Print odd numbers from 100 to 111 using do while loop
3. Print number which divisible by 3 from 100 to 111 using for loop
4. Create integer array with values from 1 to 100 and print number which in tens using for-each loop

# NESTED

- **Inner level of conditional or repetition statements.**
- **Can go as deep as we can.**

```
public class Nested {
    public static void main(String[] args) {
        int[] array = new int[] {1,2,3};
        for (int i : array) {
            if (i == 2) {
                int j = 100;
                while (j < 105) {
                    System.out.println(i + j);
                    System.out.println("" + i + j);
                }
            }
        }
    }
}
```



THAT'S ALL FOR TODAY
SEE YOU IN THE NEXT CLASS