JAVA @17

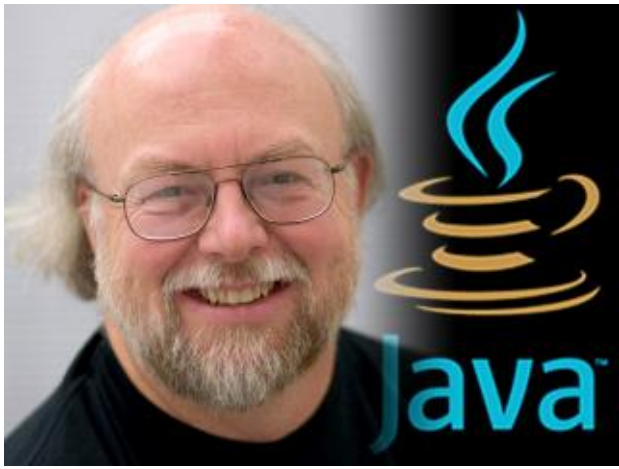
Introduction to Java

Objective

After completing this session you will be able to understand,

- Evolution of Java
- Features of Java
- How to execute a Java program.

Introduction



- Java was designed by Sun Microsystems in the **early 1990s**.
- Basic aim of java was to solve the problem of **connecting many household machines** together.
- Earlier Name of Java : **OAK**
- Creator of Java: **Mr. James Gosling** (the father of Java)
- As there was another language called Oak , they decided to rename OAK. New name was given to OAK , OAK was renamed Java in **1994**
- Java was publicly released on **May 27, 1995**
- Java was targeted at **Internet development**
- It has syntax similar to c and c++ languages, but has vastly improved features.

Goals

- Java is an object oriented.
- A single representation of a program could be executed on multiple operating systems
- It should fully support network programming
- It should execute code from remote sources securely
- It should be easy to use
- It should be “robust and secure”.
- It should be “architecture-neutral and portable”.
- It should execute with “high performance”.
- It should be “interpreted, threaded, and dynamic”.

Versions of Java

Java Version	Release Date	Year
JDK Beta		1995
JDK 1.0	January 23	1996
JDK 1.1	February 19	1997
J2SE 1.2	December 8	1998
J2SE 1.3	May 8	2000
J2SE 1.4	February 6	2002
J2SE 5.0	September 30	2004
Java SE 6	December 11	2006
Java SE 7	July 28	2011
Java SE 8 (LTS)	March 18	2014
Java SE 9	September	2017
Java SE 10	March	2018
Java SE 11 (LTS)	September	2018

Versions of Java

Java Version	Release Date	Year
Java SE 12	March	2019
Java SE 13	September	2019
Java SE 14	March	2020
Java SE 15	September	2020
Java SE 16	March	2021
Java SE 17 (LTS)	September	2021
Java SE 18	March	2022
Java SE 19	September	2022
Java SE 20	March	2023
Java SE 21 (LTS)	September	2023
Java SE 22	March	2024
Java SE 23	September	2024

Java Editions

The following are the Java frameworks,

- **Java SE** - Java Platform, Standard Edition or Java SE is a widely used platform for programming in the Java language. This is the Java Platform used to deploy portable applications for general use. In practical terms, Java SE consists of a virtual machine, which is used to run Java programs, together with a set of libraries.

  Library Examples: java.lang.*, java.net.*, jav.io.*, java.util.*.

- **Java EE** - Java Platform, Enterprise Edition or Java EE is a widely used platform for server programming. The applications developed using JEE stack can be deployed in N tier fashion in appropriate application servers and remotely accessed.

  Examples: EJB, Servlet, JSP, JSF , JMS etc.

- **Java ME** - Java Platform, Micro Edition, or Java ME, is a Java platform designed for embedded systems such as mobile devices , PDA etc. Target devices range from industrial controls to mobile phones and set-top boxes.

  Examples: Connected Limited Device Configuration (CLDC), Mobile Information Device Profile (MIDP), Information Module Profile (IMP).

Java Code

What happens after you develop a Java code?

Java Code is compiled and converted to a byte code rather than a native code.

English is an language which can understood by many people across the world. Similarly byte code is a format that can be run in many platforms Unix, Windows, Linux and also irrespective of hardware's.

Java JDK vs JRE

What is Java JDK?

JDK stands for **J**ava **D**evelopment **K**it is a package used for developing java applications and converting the java code to Byte codes. The conversion is typically done using Java compilers.

For developing and running Java applications go for the bulkier Java JDK.

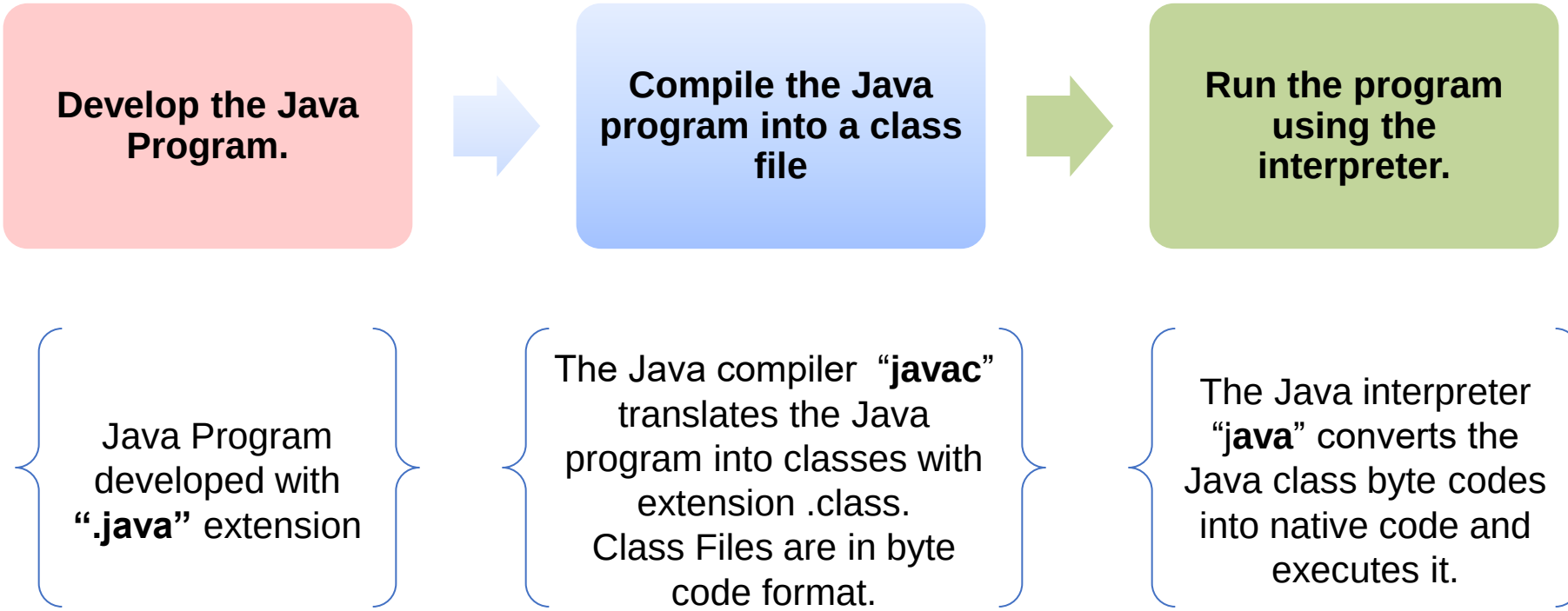
What is JRE?

JRE stands for Java runtime environment is used for executing java applications . It converts the java byte code to the necessary native code based on the underlying platform.

If you want java applications to be executed go for lighter versions JRE.

There are different JRE versions for different platforms such as Linux, Windows, Unix etc.

Basic steps to develop a Java program



Components of Java

The components which makes Java program run are,

- Java API's
- Java Class File
- **J**ava **V**irtual **M**achine (**JVM**) – *Heart of Java technology.*

What are Java API's?

They are application interface which developer uses to develop java programs.

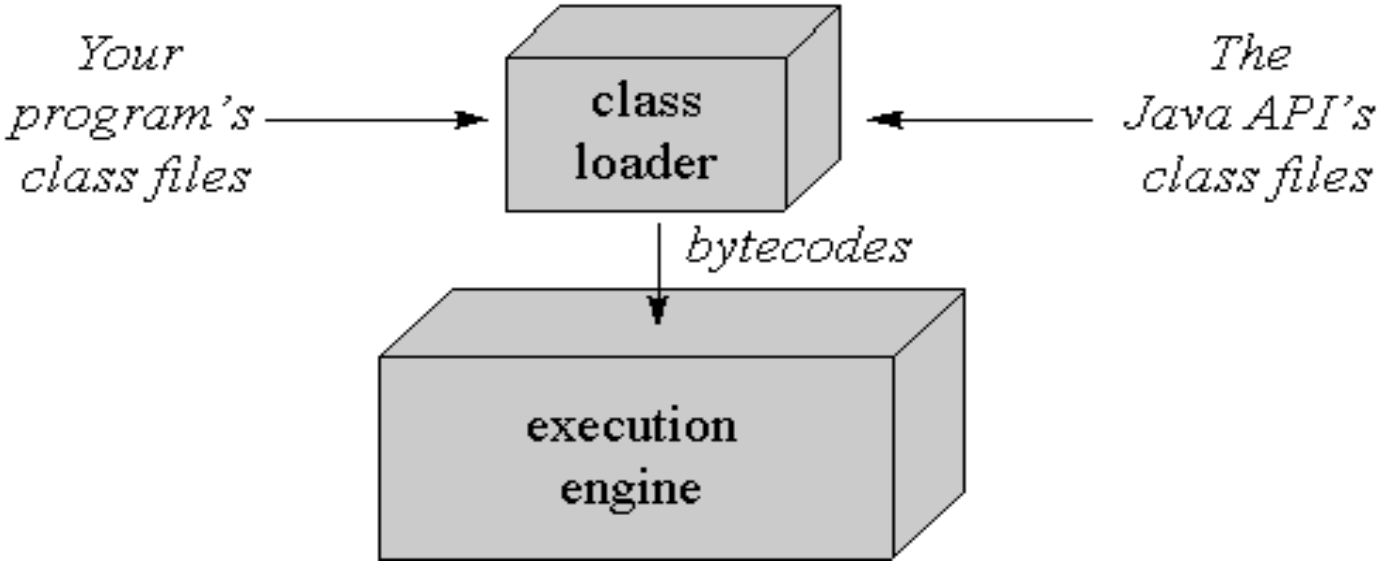
Example: System.out.println("Hello World");

Java developers uses System.out.println API to print messages on the console.

What is JVM?

Java Virtual Machines is the heart of the java platform. It is a abstract computer which,

- ✓ Loads class files using class loader
- ✓ Executes the class file using the execution engine.

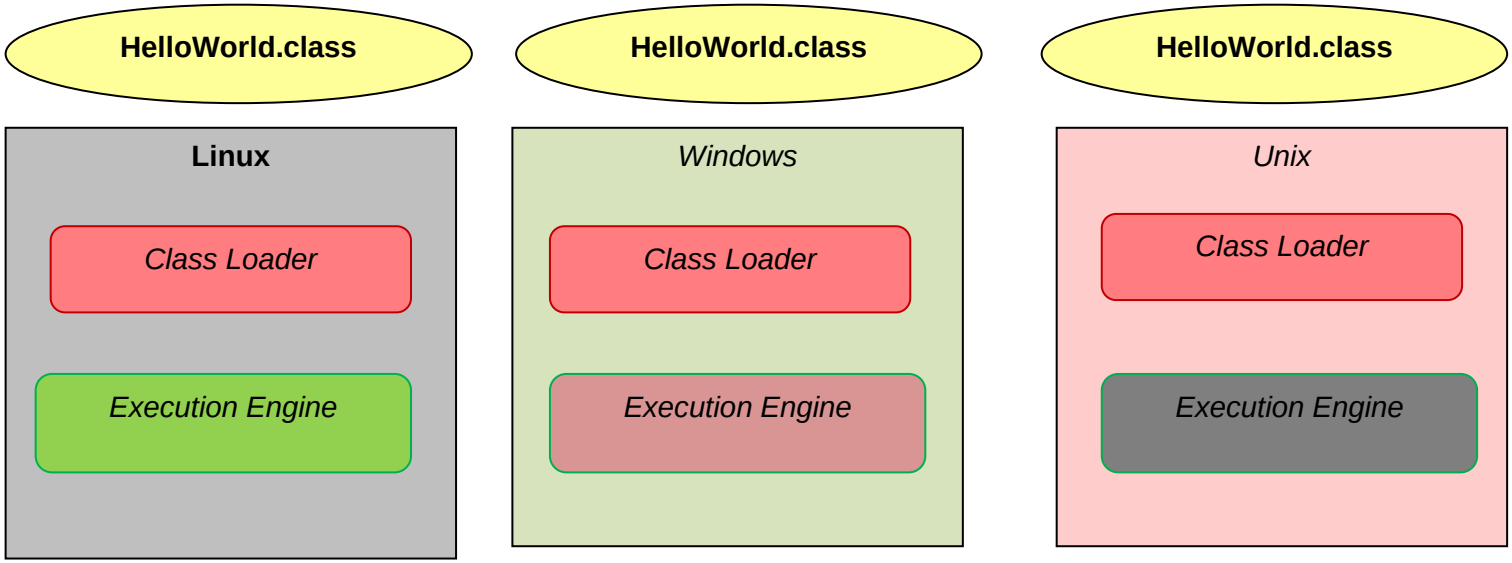


How is Java portable?

Java Platform = Java virtual Machine + Java API

Example: Linux Platform = JVM for Linux + Java API for Linux.

In the below example the same "**helloWorld**" program is developed once and run in different operating systems. The execution engine specific to each OS will translate the byte code to respective OS native code.



Example – First Java Program

1. Create a folder “workspace” in D or C drive.
2. Open notepad and type the following program in the notepad.

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

3. Save the notepad file with the file name HelloWorld.java.
4. Open a command window, type CMD in command window.
5. Set the path and class path variable.

IMPORTANT NOTE: The file should be saved under the same name as of the class name.

Java path settings

What is Path?
Path represents the folders to be searched for running the java or javac commands. Needed for compiling Java program.

How to set path: set path =%path%; C:\Users\hp\Links\jdk\jdk-11.0.2\bin;

Where %path% - To ensure that the new path is appended with existing path variables set.

NOTE: Assuming Java home is C:\Users\hp\Links\jdk\jdk-11.0.2

What is class path?

Class path is the path where the class and Java API's are loaded. Needed for executing class files.

How class path is set: set classpath =%classpath% ;

Where %classpath% - To ensure that the new class path does not to override the existing class path variables set.

Example – First Java Program

6. Compile the program : In command prompt go to the folder "JavaWorks" and compile the program as follows.

javac HelloWorld.java

Where javac is the command line tool for compiling java programs.

7. Run the program: From the same folder run the program as follows,

java HelloWorld

Where java is the command line tool used for running the program.

8. Latest Java 11 version can proceed the single command for compile and execute,

java HelloWorld.java

9. In the console you can see the message printed Hello World.

10. Change the message in the program as "Hello <Your Name>" and repeat steps 6 through 8 and see the program displaying a different output.

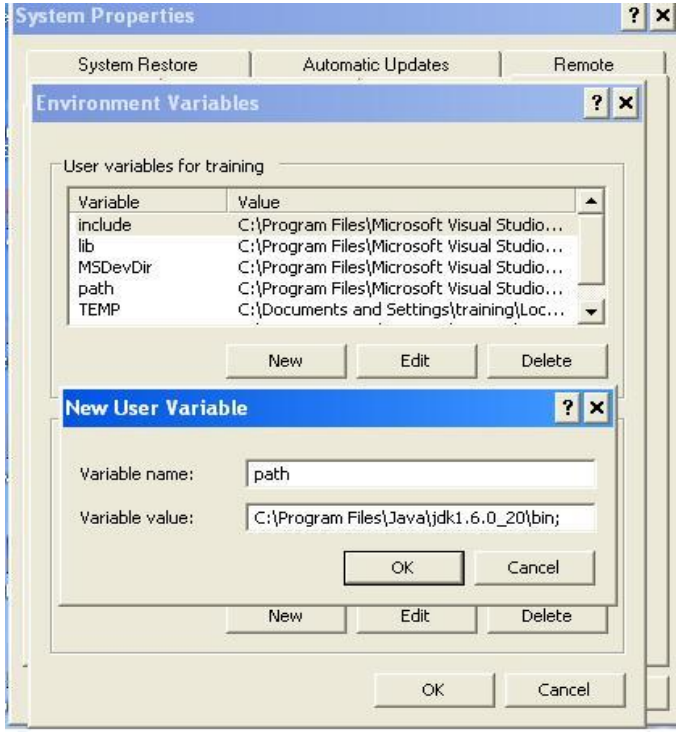
Alternate for path settings

Java path can be set in the environment variables of system properties

Right click My Computer → Click Properties→ Click Advanced → Click Environment Variables → Click New(User Variables) → Add a new variable as mentioned below

- Variable Name = path
- Variable Value = %JAVA_HOME%\bin;

NOTE: Assuming Java home is C:\Users\hp\Links\jdk\jdk-11.0.2.
This variable will be reflected across all the applications in the desk top.
On setting this the variable in environment parameter it gets reflected in all the command windows.



Lets Analyze the Code

- The main method is the starting point of any java application. Any java application to be executed using “Java” command needs a class with main method.

- The application cannot run without a main method.

- Once the Helloworld.class file is executed, the interpreter searches the main method and invokes it.

Syntax:

```
public  static void  main(String [] args)
{
   // The  program implementation
   // goes here
}
```

Lets Analyze the Code

In the program you would have noticed a statement

```
System.out.println("Hello World");
```

This is a java API used for printing messages on the console. This prints messages with a line break.

The other variant of this method is

```
System.out.print("Hello World"); //This prints messages without a line break.
```

Example:

```
System.out.print("A");
System.out.print("B");
System.out.print("C");
```

This displays the message "ABC" in the console.

```
System.out.println("F");
System.out.println("G");
System.out.println("H");
```

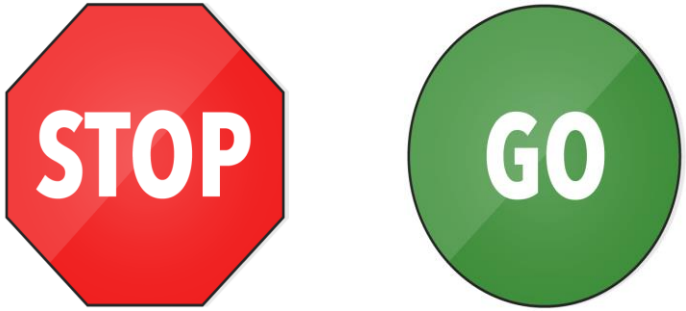
This displays the message as

F

G

H

Time To Reflect



Trainees to reflect the following topics before proceeding.

- What is the different between Java JDK and JRE?
- What is Java ME & Java EE?
- What are the types of Application Developed using Java?
- Which is responsible for the memory management in Java?

What are Java classes?

Classes are the fundamental building blocks of a Java program. Java application are built using one or more java classes, a class contains data and application business logics.

- Data are represented as variables in classes.
- Application business logic are implemented as methods in classes.
- A class is a blue print for making objects.

Classes Example: Employee, Department

NOTE:

- A single physical .java file can have more than one classes.
- The java file should be named after the class which is declared public.
- There cannot be two classes defined as public in the same java file.

Structure of a Class

The Structure of the class is as follows,

```
class <ClassName>
{
        // The class implementation goes here
}
```

This is the class declaration.

This is the class body implemented within the curly braces

The body of the class contains,

- **Variables** – This is a container for storing class data.
- **Methods** – Application behavior implemented and this changes the data (variables values).

A Sample Java Class

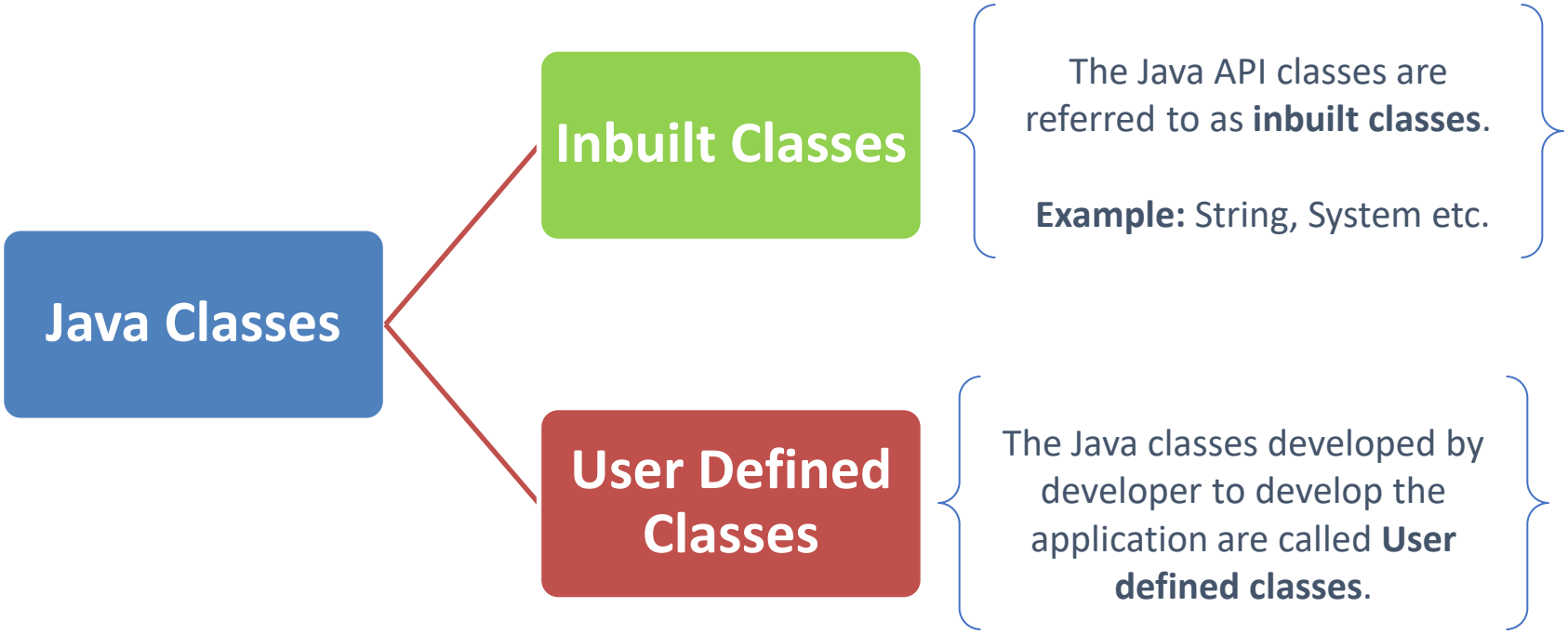
```
class Employee
{
    long empSalary=0;
    int empId=1025;

    void calculateSalary()
    {
        // logics of salary
        //calculation goes in here
        empSalary= 45000;
        system.out.println("Salary ="+empSalary);
    }
}
```

These are variables declared to store the employee id and the employee salary. The variable has a declaration and data type like integer (or) String.

This is the method definition.
The salary is stored as 10000 in the variable empSalary and printed in the console.

NOTE: Java is case sensitive. For example, "**Employee**" and "**employee**" are two different classes.

Types of Classes

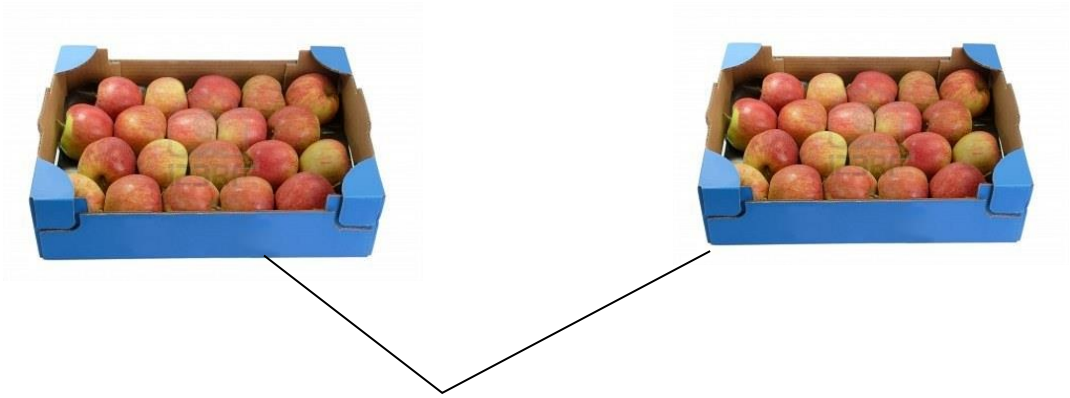


Packages

Packages are used for logically grouping the classes together into a single unit.

In the Java API, classes are grouped into packages.

Example:



Both the apple looks similar, **If you mix the apples you will end up with a confusion to differentiate the varieties. The box where they are placed helps us to differentiate them. Similarly though the java classes have the same name using the packages (box) where they reside the classes can be differentiated and used.**

Benefits of using packages

- Packages helps to organize classes into a folder structure which will be easy to maintain.

- Two different packages can have classes with the same name. If there is a naming clash, then classes can be accessed with their fully qualified name.

- Packages provide a level of security, because you can restrict the class usage, which you develop in such a way that only the classes in the same package can access it.

Creating an Object

Object creation is a three step process,

- **Declaration -** Giving a name to the object to be created.
- **Instantiation -** The new keyword and constructor creates a instance of the object.
- **<Initialization> -** The values will be initialized using the constructor.

You will learn more about the constructors and objects in the next sessions.

Main class for Employee

```
class EmployeeProgram {
   public static void  main (String[] args) {
         Employee empObject =null;
         empObject = new Employee();
         empObject.empSalary = 20000;
         empObject.calculateSalary();
   }
}
```

Declares a employee object.

Instantiates and initialize the employee object using constructor.

Sets the employee salary as 20000.

Triggers the calculate salary method in the employee object.

This program creates a employee object and invokes the calculateSalary method on the object

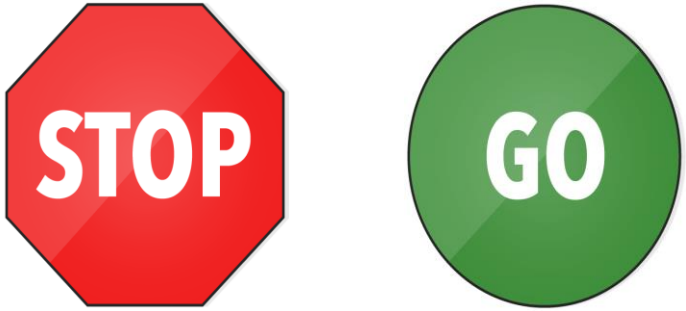
Example – Creating an Object

1. Create a java class “Student” add a integer variable “regId”.
2. Create a method “displayRegId” which will print the “regId” of the student in the format
   - “The student registration id is <RegId>”
3. Create a java class “StudentMain” add a main method which will
   - Create a object instance of the Student class
   - Set the registration id to value “1290”
   - Invoke the method ““displayRegId””.
4. The message needs to be displayed in the console.
   - Expected Output: “The student registration id is 1290”

Example – Creating an Object

1. Create a Class “Calculator” with two integer variables “op1” & “op2”.
2. Create a method “displayOperand” and add the logic of displaying the variable values in the below format,
    - “The value of operand 1 is <op1 Value>”
    - “The value of operand 2 is <op2 Value>”
3. Create a main class “CalculatorMain” implement the following logic
    - Set the value of “op1” as 98 and “op2” as 43.
    - Invoke the method displayOperand.
4. The message should be displayed as mentioned in point # 2.

Time To Reflect



Trainees to reflect the following topics before proceeding.

- How can java class be logically grouped?
- What are the benefits of java packages.
- What is the keyword used for creating objects?.

Thank you

You have successfully completed

Introduction to Java