History of java

- It is an object oriented programming (OOP) language . By the term object oriented programming we mean that the programming components are like real world entities / the actual objects that we see around .
- Previous to OOP , there was procedural paradigm of which the common example is C language . Here the programming component at the heart is function . One function calling the other function and so on.
- When the code base grew , it became cumbersome to maintain procedural codes as it was tightly coupled and a single mistake used to create havoc everywhere unlike OOP where an erroneous entity is a problem to a part of the functionality , not the whole of it. OOP and java took a lead here.

History of java (comparison with c++)

- C++ codes get compiled (compilation means conversion from source code to machine level code where machine code is a set of instructions in bits which can be straight away executed by the CPU) to native machine specific codes which renders it unusable to run on a distributed platforms ie. different operating systems also . It becomes platform dependent.
- The above problem renders C++ codes unusable for internet applications as the targeted audience for the application is going to be any machine with no particular operating system.
- Java got a lead over c++ on this point which was one of the major leads for adoption of java over c++. How this happens will be clear when we study JVM in subsequent slides.

JVM

- JVM (Java Virtual Machine) is the component of the Java platform which is mainly responsible for running the java codes.
- A .java file/source file which is in human understandable format is first compiled into a .class file by java compiler (“javac” is the java compiler).The .class file is an architectural neutral byte code (java byte code is understandable by jvm which is installed on any java enabled machine).
- Java byte codes are intended to be executed by the JVM which is inside the target device (say mobiles,browser,computers,microwave,fridge)

Java compiler

- Java compiler (javac) is primarily responsible for converting the source code into the architectural neutral byte codes.The java bytecodes can be run by any machine with jvm (java virtual machine) installed on it as it can understand and run java bytecodes generated through java compiler.
- Syntax checking : It is catching most of the errors at the compile/static time (mainly syntax related errors that we are getting while coding not at running the code) . Example if you miss a “;” at the end of a statement you won’t get this error while running the code rather you will get this error while compiling the code.
- In eclipse when you are writing the code and you miss a “;” , it will show you a red error on that line and also the error statement - very helpful feature.

Architecture of JVM

- Jvm has two parts , one is the interpreter and the other one is the jit (just in time) compiler.
- Interpreter (jvm component) : converts the byte code to machine code and executes it directly .Please note that here we are discussing here the interpreter inside jvm not a general interpreter which can run even a source code like the interpreter used for LISP language.
- JIT (jvm component) : Just in time compiler induces speed into the run time execution of java codes.It converts the bytecodes into native machine codes and then caches(holding an image of the machine code in the CPU) it and runs it.
- Running the machine codes is already a faster process and most jvm's adopt it for higher speed . Moreover if a code has to execute 1000 times then JIT of JVM runs the cached native machine code every time and saves lots of time.

Main features of Java

1. Object oriented programming.
2. Distributed platform / internet applications.
3. No overhead of pointers unlike c/c++ . (I am assuming each of us faced some pointer issues in our first year of engineering irrespective of any branch).
4. Multithreading support. (A deck dedicated to it talks about it later)
5. Type safety (Generics) . (A deck dedicated to it talks about it later)
6. Open source language and rich set of libraries . Most of the open source project are in java . Example : Apache camel,Apache Hadoop,Apache Mahout and the list goes on
7. Automatic garbage collection. (in C++ the memory allocated to objects had to be manually deallocated which was bad as it was prone to human error , java does itself)

Basic programming constructs

1. For loop example : Used for looping/iterations . “//” is used for code comments . Following code snippet and comments show looping in action.

```
package javatutorial;
public class ForLoopDemo { // clas will be discussed later in the slide “ Classes and Objects “
public static void main(String args[]){  // This is the starting point of execution , the keywords used will be discussed later
        // start of “for” loop
         for(int i=1;i<=5;i++){ // here i variable is initialized to 1 , 5 is the number of loop count and i++ means incrementing by 1
                System.out.println("the value of i is "+i); // System.out.println is used to print in java
                System.out.println("i am getting printed "+i+" many time(s) ");
         } // end of for loop
}
}
```

For loop output view

```
// This output can be analyzed with the above code to understand the looping mechanism .
/*
the value of i is 1
i am getting printed 1 many time(s)
the value of i is 2
i am getting printed 2 many time(s)
the value of i is 3
i am getting printed 3 many time(s)
the value of i is 4
i am getting printed 4 many time(s)
the value of i is 5
i am getting printed 5 many time(s) */
```

If-else construct

If-Else is a programming construct where the code inside the parenthesis ”{}” followed by “if” or the “else” keyword executes as per the condition specified in front of the “if” keyword in circular brackets “()”. Following code snippet shows that.

```
public class IfElseDemo {
        // demonstrates the if else demo. Used to do decision based ( if-else based) coding.
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                int i;
                //take either of the values : zero or non zero
                i=0; //  zero assigned to i invokes the if part
```

If-else continued ..

```
if(i==0){
                        System.out.println(" apple gets printed when i is 0 ");
                }
                else{
                   System.out.println(" car gets printed if i is not 0");
                }
        }
}

// OUTPUT to demonstrate and understand if - else condition
/*apple gets printed when i is 0 */
```

Switch construct

- In switch construct , we pass an integer value and for a range of integer values that can be passed , we write specific cases . Example if we write switch(1) , then the code written inside case 1 gets executed.Following code snippet demonstrates the working of a switch programming construct.

```
package javatutorial;
public class SwitchDemo {
                public static void main(String[] args) {
    int day=0;
    //day=1; will envoke tuesday
    //day=6; will envoke sunday and similarly
```

Switch case code

```
switch(day){
    case 0: System.out.println("day is Monday");
    break;
    case 1: System.out.println("day is Tuesday");
    break;
    case 2: System.out.println("day is Wednesday");
    break;
    case 3: System.out.println("day is Thursday");
    break;
    case 4: System.out.println("day is Friday");
    break;
```

Switch case code

```
 case 5: System.out.println("day is Saturday");
       break;
       case 6: System.out.println("day is Sunday");
       break;
       }
           }

}

// OUTPUT to show the working of switch programming construct.

/*day is Monday*/
```

Exercise

- What is the difference between jvm and compiler ?
- What is the run time execution engine for java ?
- What is the difference between jit and interpreter ?