Lecture 2 - Introduction to JAVA

Basics :

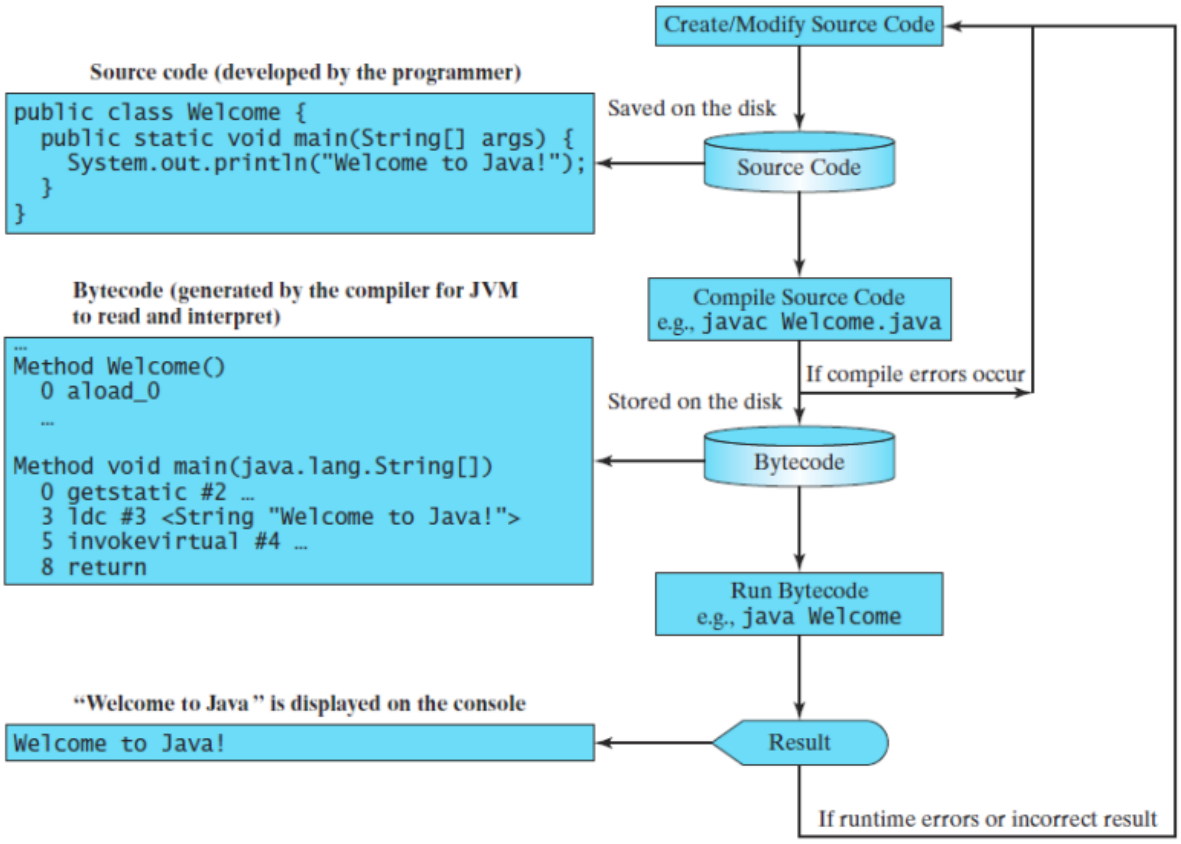
- Invented by James Gosling, 1994
- **Portability/WORA**(write once run anywhere) language, Can run anywhere in any device where we can run JVM (Java virtual Machine)
- Widely used in industry
- High level lang
- Develop software running:
    - Desktop computer
    - Server
    - Mobile devices

Three main steps in java:

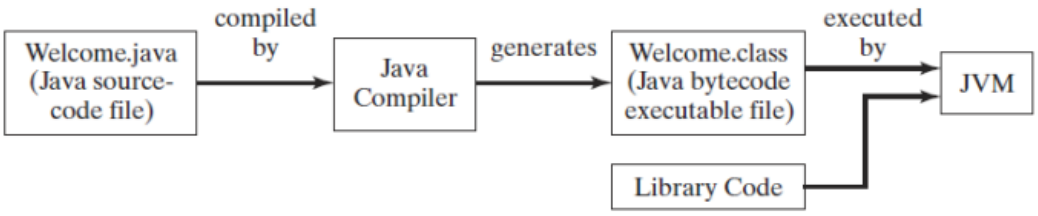
1. Writing the source code using text editor
2. Translating source code into JAVA bytecode using compiler
    a. Bytecode is similar to machine instructions(Low level) but is architectures neutral and can run on any platform that has JVM
3. Executing Bytecode

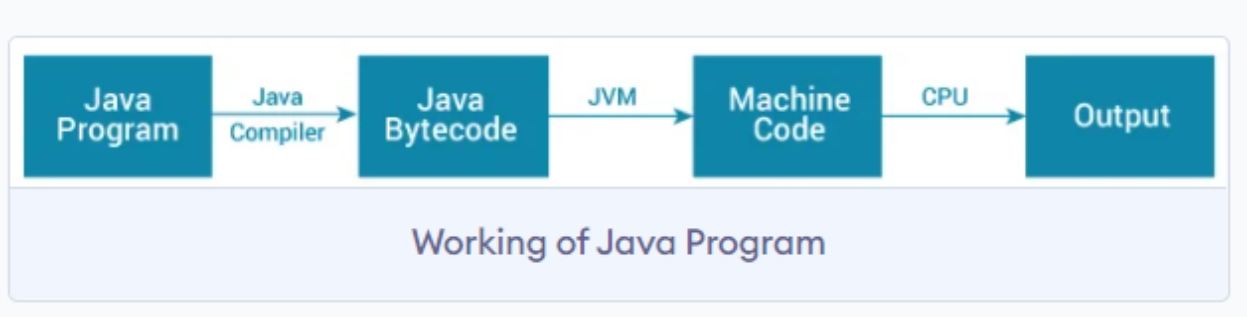
a. JVM is interpreter: it translates bytecode into target machine language code one at a time rather than whole program as a single unit

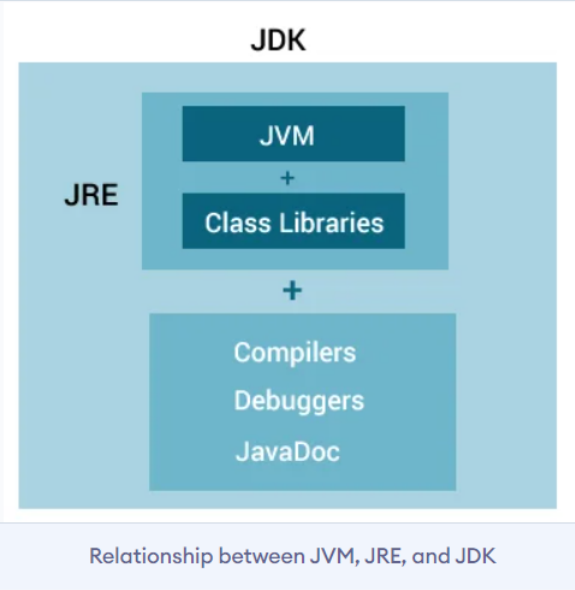
b. Each step is execute immediately after it is translated



Compiler converts Java file to bytecode







Class includes:

1. Data fields to represent state of objects
2. Methods to represent the behavior if objects
3. Special type of methods - Constructor
    a. not return type
    b. 0 or more arguments

c. Should have the same name of class

d. invoked using the **new** operator

main Method

Execution starts here, this is the entry point of the program

Default values

null - reference type

0 - numeric type

false - Boolean type

'\u0000' - char type

- Java assigns no default value to local variable inside a method

```
public double dis(){
  int n;
  System.out.println(n); // !Error cause it is  a local variable
}
```

Static Modifier

```
public Point{
  int x;
  int y;
  static int s;
  // To keep track of number of objects
  static int counter

  public Point()
    {
      counter++; // Increases
    }
}


// inside the main file -> psvm
```

```
Point p1 = new ....
Point p2 = new ....
p2.s = 10;
p1.s = 20;

System.out.println(Point.s);   // Result will be 20

System.out.println(p2.s);      // Result will be 20

System.out.println(p1.s);      // Result will be 20
```

- Inside **static method** we cannot invoke **non-static method/field**
- We can **invoke static method** by using **instance**(Objects) or by the **class** name
- I can access static method from non static method

Arrays

Is a Data structures that represent a collection of the same types of data.

```
int[] A = new int[20];
```

Size of Array

```
A.length
```

Array initializer

```
int[] A = {1,2,4,76,6};
```

For each loop

```
for(int a: A){
    sysout(a);
}
```

```
what if?
    int[] A;
    A.add(10);
```