JDK, JRE & JVM

Java Development Kit(JDK)

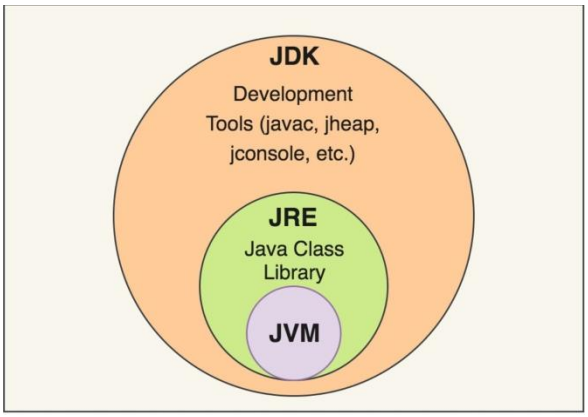
JDK is a tool used to develop Java applications and has a compiler(javac).

Java Runtime Environment(JRE)

JRE is a software that provides Java class libraries and other resources that a Java program needs to run.

Java Virtual Machine(JVM)

JVM is a virtual machine that allows a computer to run a Java program.



Compile/Run

Java Compiler

The compiler converts Java code(.java) into byte code(.class)

Compile

```
Javac ClassName.java
```

Java is Platform-independent - The byte code(Intermediate-level language) is platform-independent. This byte code is then fed into JVM of the respective OS and gets converted into binary code(Machine-level language). Hence, Java is **WORA(Write Once Run Anywhere)**.

Run

```
Java ClassName
```

Variables

Containers for storing data.

```
dataType variableName;
```

Java is a **statically-typed** language **-** The data type of the variable is known at compile-time.

Data Types

Primitive Data Types

These are the basic built-in data types.

Integer – stores whole numbers.
byte, short, int, long(l)

Float – stores decimal numbers.
float(f),double

Boolean – stores true or false.
Boolean

Character – stores a character.
char('')

Reference Data Types

Reference type variables store the address(refer) of an object.

Operators

Operators are used to perform operations on operands.

Arithmetic Operators

+ : Addition
– : Subtraction
* : Multiplication
/ : Division
% : Modulo Division(returns remainder)

Assignment Operators

= , += , -=, *= , /= , %=

Comparison Operators

== : equal to
!= : not equal to
> : greater than
>= : greater than or equal to
< : lesser than
<= : lesser than or equal to

Logical Operators

&& : logical AND
|| : logical OR
! : logical NOT

Ternary Operator

```
condition ? true value : false value;
```

Increment and Decrement Operators

pre-increment/pre-decrement : ++variableName / --variableName
– process after increment/decrement

post-increment/post-decrement : variableName++ / variableName--
– process before increment/decrement

Conditional statements

if statement

```
if(condition) {
    //code to be executed if condition is true;
  }
```

if-else statement

```
if(condition) {
    //code to be executed if condition is true;
  }
else {
    //code to be executed if condition is
false;
  }
```

if-else-if ladder

```
if(condition-1) {
    //code to be executed if condition-1 is
true;
  }
else if(condition-2) {
    //code to be executed if condition-2 is
true;
  }
```

```
.
.
.
else {
   //code to be executed if none of the
conditions is true;
  }
```

switch statement

```
switch(expression) {
   case x:
     {
      //code
      break;
     }
   case y:
     {
      //code
      break;
     }
   case z:
     {
      //code
      break;
     }
   default:
     {
      //code
     }
  }
```

Loops

for loop

```
for(initialisation;condition;alteration) {
   //code
  }
```

while loop

```
initialisation;
```

```
while(condition) {
   //code
   alteration;
  }
```

do while loop

```
initialisation;
do {
   //code
   alteration;
  }
while(condition);
```

for each loop

used to traverse through a collection of elements.

```
for(dataType variableName : collectionName)
  {
   //code
  }
```

break

used to break out of a loop.

continue

skip the current iteration and jump to the next iteration.

System Output

```
System.out.print(arguments);
```

User Input

```
import java.util.Scanner;
Scanner refVariableName = new Scanner(System.in);
dataType variableName = refVariableName.methodName();
```

Methods

A method is a block of code used to perform certain actions.

```
returnType methodName(parameters)
  {
     //method body
  }
```

method call

```
methodName(arguments);
```

parameters : values inside a method.

arguments : values passed to a method.

Java is strictly **pass-by-value**.i.e.only a copy of the argument is passed and not the actual argument.

Local variables

variables inside a method.