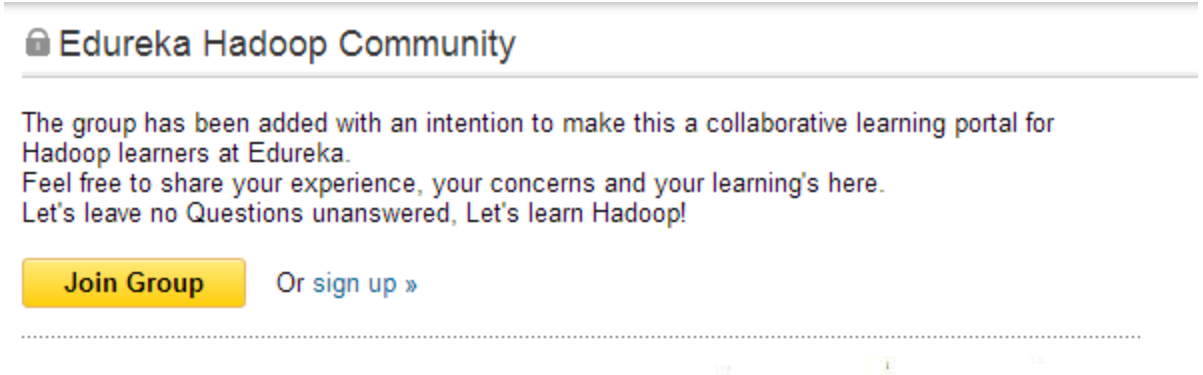
edureka!

Java Review
(Essentials of Java for Hadoop)

Have You Joined Our LinkedIn Group?



What is Java ?

edureka!

- **Java**

  - Java is not just a programming language but it is a complete platform for object oriented programming.

- **JRE**

  - Java standard class libraries which provide Application Programming Interface and JVM together form JRE (Java Runtime Environment).

- **JDK**

  - JDK (Java development kit) provides all the needed support for software development in Java.

Java Virtual Machine (JVM)

edureka!

- Runs the **Byte Code**.
- Makes Java **platform independent**.
- Handles **Memory Management**.

How Java works ?

edureka!

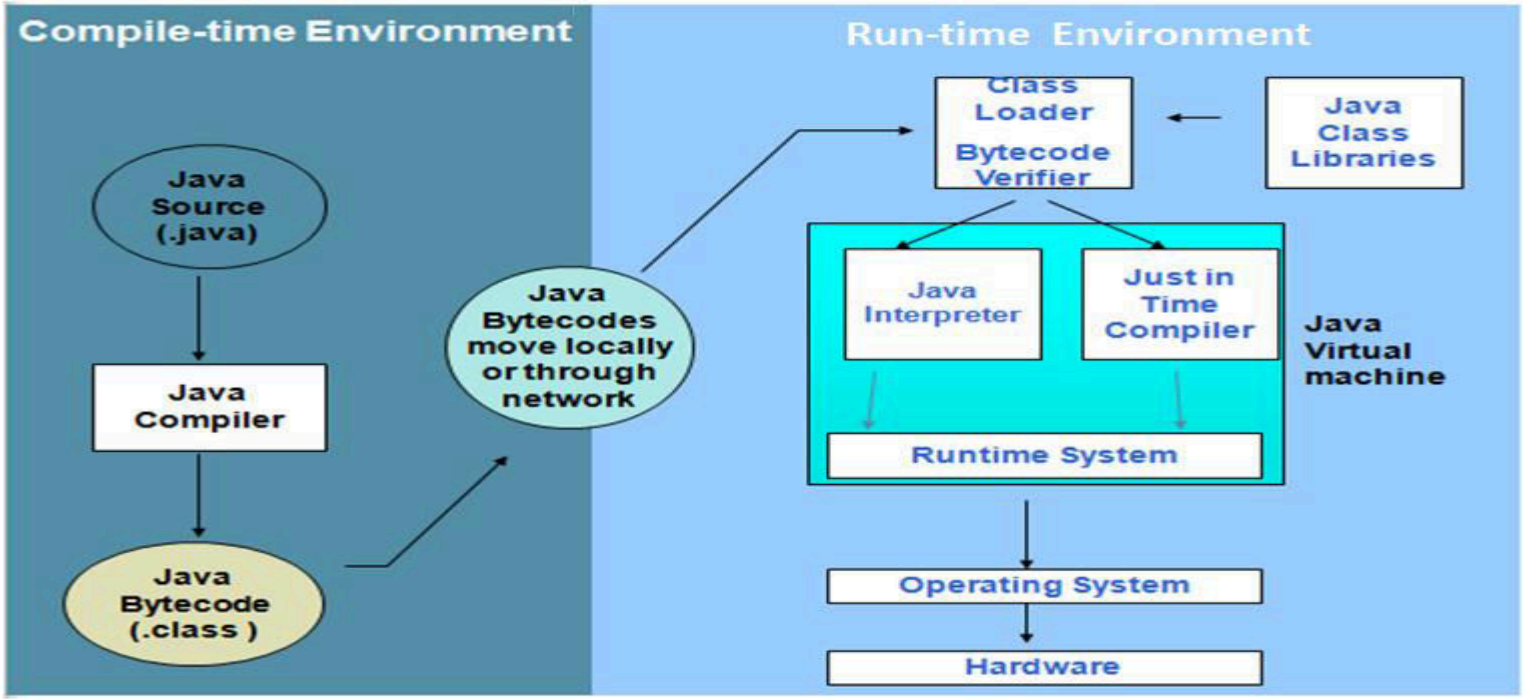
- Java compilers convert your code from human readable to something called “**bytecode**” in the Java world.

- “**Bytecode**” is interpreted by a **JVM**, which operates much like a physical CPU to actually execute the compiled code.

- **Just-in-time** (JIT) compiler is a program that turns Java *bytecode* into instructions that can be sent directly to the *processor*.

How Java works ?

edureka!



edureka!

Basic Java Syntax

Data Types in Java

edureka!

- Data types define the **nature** of a value
- We need different data-types to handle real-world information

Name	Size (in bits)	Range
long	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	–2,147,483,648 to 2,147,483,647
Short	16	–32,768 to 32,767
byte	8	–128 to 127
double	64	4.9e–324 to 1.8e+308
float	32	1.4e–045 to 3.4e+038
char	16	0 to 65,536
boolean	??	true/false

Naming Convention of Variables

edureka!

- Can start with a letter, an underscore(_), or a dollar sign ($)
- Cannot start with a number.

✓ *long _LongNumber = 9999999;*

✓ *String firstName = “John”;*

✓ *float $Val = 2.3f;*

✓ *int i, index = 2;*

✓ *double gamma = 1.2;*

✓ *boolean value2 = false;*

Operators

edureka!

- Provide a way to perform different operations on variables

- Categories of Java Operators

Assignment Operators	=					
Arithmetic Operators	-	+	*	/	%	
Relational Operators	>	<	>=	<=	==	!=
Logical Operators	&&	||	&	|	^	
Unary Operators	+	-	++	--	!	

Assignment and Arithmetic Operators

- Used to assign a value to a variable
- Syntax
  - <*variable*> = <*expression*>

Assignment Operator	=

- Java provides eight Arithmetic operators:
  - for addition, subtraction, multiplication, division, modulo (or remainder), increment (or add 1), decrement (or subtract 1), and negation.

Arithmetic Operators	-	+	*	/	%	++	--

Relational Operators

edureka!

- Used to **compare** two values.
- Binary operators, and their operands are numeric expressions.

Relational Operators	>	<	>=	<=	==	!=

Logical Operators

edureka!

- Return a **true** or **false** value based on the state of the variables
- There are **six** logical operators

	Conditional AND	Conditional OR	AND	OR	NOT	Exclusive OR
Logical Operators	&&	||	&	|	!	^

Static versus Non-static Variables

edureka!

- **Static** variables are shared across all the objects of a class
  - There is only *one copy*
- **Non-Static** variables are not shared
  - There is a *separate copy for each individual live object.*
- Static variables **cannot** be declared **within** a method.

Statements & Blocks

edureka!

- A simple **statement** is a command terminated by a semi-colon:

  name = “Fred”;

- A **block** is a compound statement enclosed in curly brackets:

  {

  name1 = “Fred”; name2 = “Bill”;

  }

- Blocks may contain other blocks.

Flow of Control

edureka!

- Java executes one statement after the other in the order they are written.
- Many Java statements are flow control statements:

Alternation:

if, if else, switch

Looping:

for, while, do while

Java Basic Constructs

edureka!

- If…else
- Switch Statement
- For Loop
- While Loop
- Do...While Loop

If else– Syntax

edureka!

```
if ( <condition> )
{
        // Execute these statements if <condition> is TRUE
}
else
{
        // Execute these statements if < condition> is FALSE
}
```

switch– Syntax

edureka!

```
switch (expression)
{
case cond1:
            block_1;
            break;

case cond2:
            block_2;
            break;
...
default:
            block_default;
}
```

For– Syntax

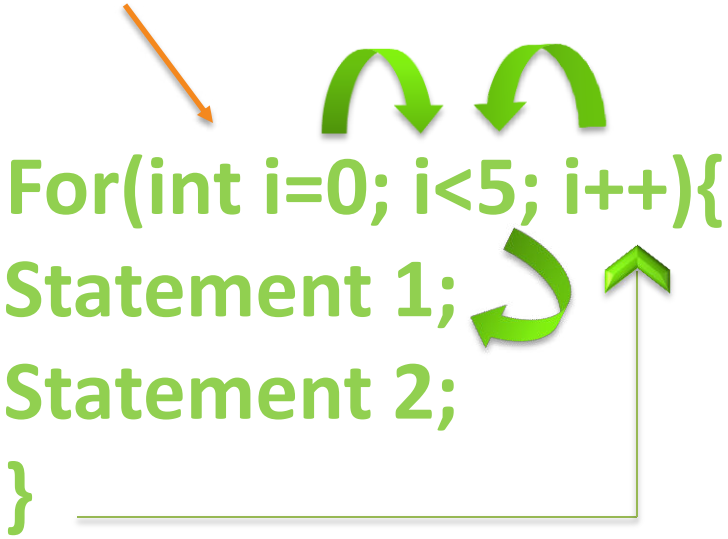
edureka!

```
for (initialization; condition; increment/decrement)
{
        statement 1;
        statement 2; . .
 }

Sample:
for( int i=0; i<5; i++ )
{
        System.out.println(i);
}
```

For loop - Working

edureka!



While – Syntax

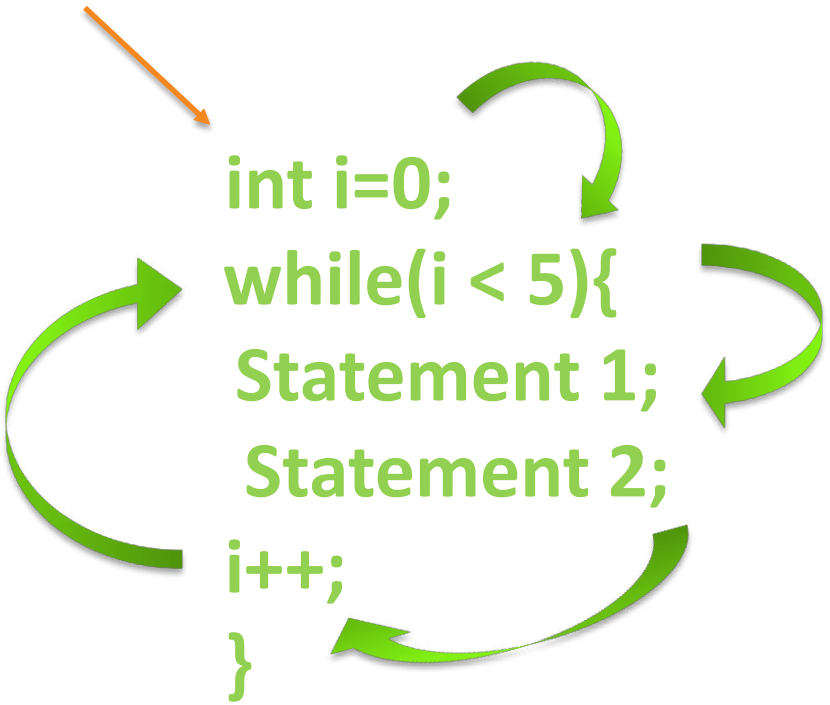
edureka!

```
while (condition)
{
        statement 1;
        statement 2; . .
 }

Sample:
int i=0;
while (i<5)
{
        System.out.println(i);
        i++;
}
```

While Loop - Working

edureka!



Do While – Syntax

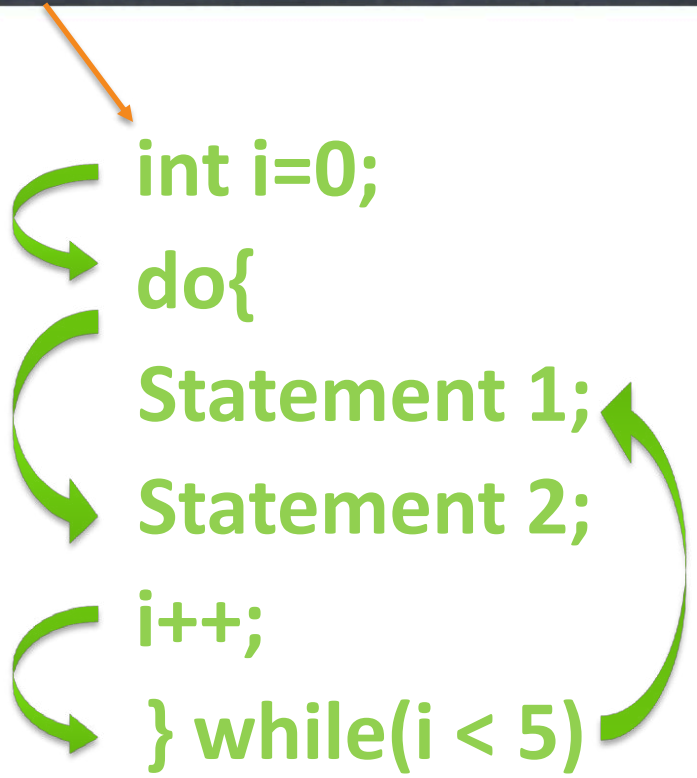
edureka!

```
do
{
        statement 1;
        statement 2; . .
 } while (condition) ;

Sample:
int i=0;
do
{
        System.out.println(i);
        i++;
} while (i<5);
```

Do While Loop - Working

edureka!



Arrays

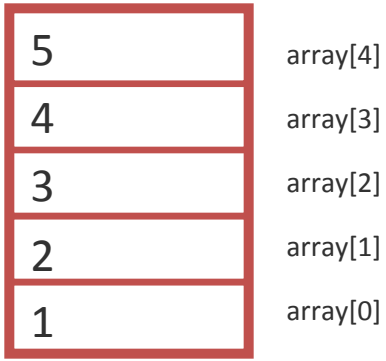
edureka!

- An **array** is a list of similar things.
- An array has a fixed:
  - *name*
  - *type*
  - *length*
- These must be declared when the array is created.
- Array size **cannot** be changed during the execution of the code.

Example of an Array

edureka!

```
int array [] =  new int[5];
for(int i=0; i<5 ; i++)
{
    array[i] = i+1;
}
```



edureka!

Java Methods & Classes

An Example class

edureka!

```
package com.edureka.entity;          // package

public class Car{                    //class declaration
   String name;
   String color;
   float weight;
   ...
   public void move(){                    //
   method
       ...
   }
}
```

Methods

edureka!

- A **method** is a named sequence of code that can be invoked by other Java code.
- A **method** takes some parameters, performs some computations and then optionally returns a value (or object).

Ex:

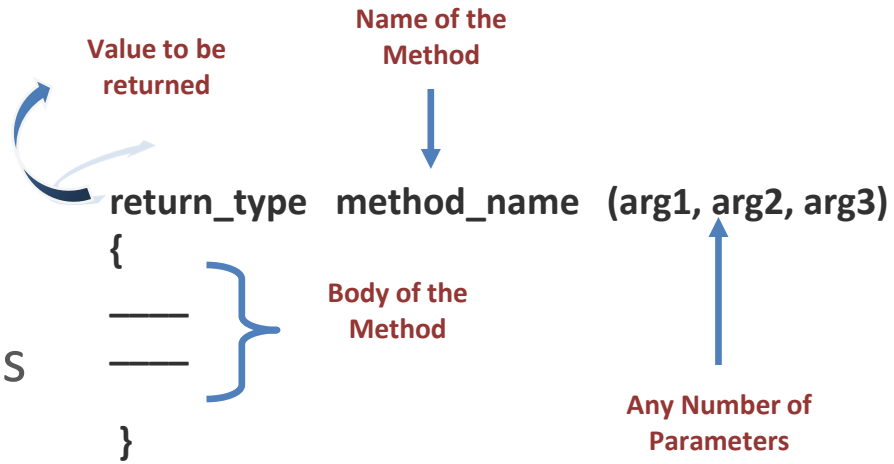
```
public float convert_to _Celsius( float temp) {
        return(((temp * 9.0f) / 5.0f) + 32.0f );
    }
```

Methods

edureka!

Methods have five components:

- Modifiers
- The return type
- The method name
- The parameter list in parenthesis
- The method body, enclosed between braces



Modifiers

edureka!

- **public**: any method (*in any class*) can access the field.
- **protected**: any method in the *same package* or any *derived class* can access the field.
- **private**: only methods *within the class* can access the field.
- default is that only methods in the same package can access the field.

- Q& A..?

Thanks..!