**Core Java**

Basic Core Java

Core Java intro, Setup for Core Java, Writing, compilating and running first java program, Variables and Data types, Control flow statement (if, switch, while, do-while and for), Class, Object, methods, Array, Build in java classes

OOPs In Java

Inheritance, Polymorphism, Abstraction and Encapsulation. Constructor, Keyword (Super, this , final static), packages, imports.

Advance Topics in Core Java

Exception Handling, Threading, Collection, IO handling, JDBC.

JDK Download

<https://www.oracle.com/java/technologies/downloads/>

Open JDK (32bit OS)

<https://developers.redhat.com/products/openjdk/download>

jdk-8u312-x86 MSI OpenJDK 8 Windows 32-bit

JavaDocs

<https://docs.oracle.com/javase/8/docs/api/>

**Java Setup Setps**

1. Download JDK
2. Install JDK
3. Verify Installation
   1. Go To C driver -> Program File folder -> Java -> JDK
4. Setup a Path of Java into Environment Variable
   1. ***Set JAVA\_HOME***
      1. Right Click on “This PC” / ”My Computer”
      2. Select **Properties** option
      3. Select “Advance System Settings” from the right side of the new window.
      4. Click on “Environment Variables…” button on new Window.



* + 1. Click on “**New**” button from the **System variable** section of the new window.



* + 1. Set the Variable Name and Variable Value on new Window.

**Variable Name : JAVA\_HOME**

**Variable Value : <JDK-Path> “C:\Program Files\Java\jdk-11.0.11”**



* 1. ***Set Path*** 
     1. Follow above step from 1 to 4
     2. Select “Path” variable from the System variable section.
     3. Click On “Edit” button
     4. Click on “New” button from the new Window.
     5. Add following Variable values:

***%JAVA\_HOME%\bin -> Windows 8 and above***

***;%JAVA\_HOME%\bin; -> Windows 7 and less***



1. **Verify Path set or not**
   1. Open a CMD
   2. Type Command

“javac”

“java -version”



**Java Language**

1. Java is use to write a programs/ to develop an application (software).
2. Can develop web application, Console based application, Mobile Application, desktop application, embedded application.
3. Java introduce in 90s.
4. Java Editions
   1. Java SE (J2SE)
      1. Java Standard Edition
      2. Also Known as Core Java
      3. Can develop console based and desktop applications.
   2. Java EE (J2EE)
      1. Java Enterprise Edition
      2. Also known as Advance Java
      3. Can develop Web Applications.
   3. Java ME (J2ME)
      1. Java Micro Edition
      2. Can develop Embedded and mobile application.

**Writing First Java Program**

1. **Write Java Code**
   1. Open Editor (Notepad)
   2. **Write a java program**
      1. Create a Java class.
      2. Everything in java must be write inside a class excepts import and package statement.
      3. Syntax:

**public class <Class\_Name>**

**{**

**}**

1. Create a Main method inside java class.

Main method is the start point of java program.

* + - 1. Inside method you can write an executable statement.
      2. Syntax:

**public static void main(String args[])**

**{**

**}**

* + 1. Write an executable statement.

System.out.println(“Hello, Welcome to Core Java First Program”);

* 1. **Save the file**
     1. File must have **.java** extension.
     2. File name must be a public class name.

1. **Compile java code**
   1. Here, java code will be converted from higher level language to lower-level language (Byte/binary code)
   2. Also, it will verify the syntax of the program while compilation.
      1. Open a CMD
         1. The path of the CMD must be pointing to a location where you save your java file.
      2. Execute the following command

**javac FileName.java**

* 1. As an outcome of compilation step .class file will be generated which is a binary file.
  2. This file will be use further to execute java code.
  3. .class files create for the java classes from the java file.

1. **Run/Execute java code**
   1. Execute code using CMD
   2. To execute use following command

**java className**



1. One source file can have a more than one classes.
2. One source file must have only one public class all other classes must be a non public.
3. Source file name must be a same as public class name.
4. Foe every class from the source file will get a separate .class file after compilation.

**Useful Components for Java Program**

* + - 1. JDK:
         1. Java Development Kit
         2. This is useful for the developer, It is consist of many components which helps java developer to create and execute java program.
         3. JDK consist of JRE (till jdk1.8), JVM, API, dev tools like javac, javap, javadocs , javaw
         4. JDK must be present on developer’s system.
      2. JRE:
         1. Java Runtime Environment.
         2. JRE use to execute the Java programs and you cannot compile the java code just by using a JRE.
         3. JRE must be present at client system.
         4. JRE consist of JVM and API.
      3. JVM
         1. Java Virtual Machine.
         2. JVM is help us to execute the java program.
         3. During the execution of java program, JVM will create memory virtually on the RAM, create and manage a thread to execute a java program.
         4. JVM works in 3 steps

**Class Loader**: in this stage JVM load a .class file into a memory.

**Byte Code Verifier**: Verify the byte code file whether it is compiled from the valid compiler, valid version, it no modified after compilation or not.

**Execution**

* + - 1. API
         1. Application Programming Interface
         2. Predefine functionalities, these functionalities will be provided through a java classes, interface and jar (collection of .class file) files.

**Keyword, Literals and Identifier**

**Keyword**

1. Keywords are the predefine words or reserve words by Java language.
2. These words cannot be use for custom requirement.
3. There are 52 keywords present inside java.
4. Java is case sensitive languages. All the keywords are in small cases.
5. Examples:

public, class, void, static, if, while, continue, do, for, private, protected, default, case, break, int, short, long, byte etc.

**Literals**

1. Literals are the values.
2. There some values which is predefine by java. These values are also included in keyword.
3. There are 3 literals

**true, false, null**

**Identifier**

1. Identifier are the words which can be define by the user/developer for own purpose.
2. Identifiers are use for class name, method name, variable name, object name, constants.
3. There are rules and conventions for creating and using Identifier.
4. Rules for Identifier
   1. Identifier are case sensitive.
   2. Identifier must not be a keyword.
   3. Identifier can contain character, numbers and symbols ($ \_).
   4. Allowed symbols are **$** and **\_**
   5. Identifier must be start with character(alphabets) or symbol, Identifier never starts with number.
   6. Spaces are not allowed in identifier.
   7. There is not length limit to create an identifier.
   8. Examples

|  |  |
| --- | --- |
| **Identifier** | **Valid/Invalid** |
| Int | Valid |
| int | Invalid |
| FirstClass | Valid |
| $salary | Valid |
| 1stClass | Invalid |
| Demo\_Program | Valid |
| New-class | Invalid |
| \_$\_ | Valid |
| \_100 | Valid |
| email@id | Invalid |
| \_ | Valid |

1. Conventions to create class
   1. First character of class name should be in capital case.
   2. If class name is combination of more than one words. Then every first character of the word should be in capital case.
   3. Example:

First, String, System, EmployeeDetails, EmployeeAdderssInfo

1. Convention to create methods, variables, object
   1. First word should be in small case.
   2. If identifier is combination of more than one word then, from the 2nd word onwards every words starts with capital case.
   3. Example:

out, args, main, println, printDetails, showEmployeeDetails

1. Convention to create constants
   1. All constants should be in capital case.
   2. If it is a combination of multiple words then use ‘\_’ between the words (Sanky way).
   3. Example

PI, COMPNAY\_NAME, GRAVITY

public class First

{

public static void main(String args[])

{

System.out.println("First Program");

}

}

**Keyword**: public, class, static, void

**Identifier**: First, main, String, System, args, out, println

**Data Type and Variables**

**Data Type**

1. Is use to assign a type of data.
2. There are 2 categories of Data types.



**Integer Type**

1. Numeric type of value without any decimal point.
2. These values can be a negative or positive.
3. Example: 112, -234, 346, 0

**Decimal Type**

1. Numeric type of values with decimal point.
2. These values can be negative and positive
3. Example: 234234.23, 3.14, 9.8

**Textual Type**

1. Can store a single character/symbol or even an integer type value
2. In this data type ASCII values can be store.
3. Char values cannot be negative hence, it is also known as unsinged integer.
4. Example: N, 78, @, 64, 0

**Logical Type**

1. Can store **true** or **false** inside this type
2. In java 0 consider as false and 1 consider as true internally, but you cannot store 0,1 values as a Boolean value.
3. Example: true, false

**Variable**

1. Variables are use to store a value/data.
2. Use to print the values as an output.
3. Use to assign values to any another variable.
4. Use in a match or logical expression (calculation).
5. Variables are use with the data types only.
6. Syntax to primitive create variable

**Datatype Identifier(Name) = value;**

**byte id = 12;**

****

**int a; // declaration of variable.**

**a = 20; // initialization of variable**

**Formula To calculate data type Range**

**- 2bits-1 to 2bit-1-1**

Can find the range of values for any data type except char and Boolean

Byte: -28-1 to 2bit-1-1 : -128 to 127

**Rules For creating variable**

Rule for Float values

1. Float variable values must be use with a suffix ‘F’ or ‘f’

Rule for Long values

1. Long variable values must be use with a suffix ‘L’ or ‘l’

Rule for char values

1. Char values must be created inside single quotes.
2. Values must be a single char, symbol or any non-negative (ASCII) int value

**Important points**

1. The numeric value without any decimal point and data type is always get a memory of int.
2. The numeric value with decimal point and without data type is always get a memory of double.

**Casting in Primitive Data types**

1. Casting is a process in which one data type value converts into another data type.
2. There are 2 types of casting in java
   1. **implicit Casting**
      1. The casting perform internally automatically by Java is called implicit casting.
      2. There are not loss of any bits int this type of casting.
      3. Example:

int x = 10;

long y = x; // implicit casting

System.out.println(y);

* 1. **Explicit Casting**
     1. The casting has to perform the developer manually is called as explicit casting.
     2. There is always a loss of bits in this type of casting. And due to this it may give you logically incorrect output.
     3. Example:

byte b = 10;

b = **(byte)** (b + 1);

int x = 10;

short z = (short) x; // explicit casting

System.out.println(z);

**Values Representation**

1. Binary
   1. The Values which are start with 0b or 0B are the binary values
   2. These values are allowed in java from JDK 1.7 and onwards version.
2. Octal
   1. The Values which are start with 0 are the octal values
3. Decimal
   1. These are the regular values.
4. Hexa-Decimal
   1. The Values which are start with 0X or 0x are the hexa-decimal values

**Types Of Variables**

1. **Local variable**
   1. The variables which are created inside a method or at a methods parameter are called local variable.
   2. This variable scope (accessible) is only inside a method.
   3. Local variables have initialize before use.
2. **Instance Variable**
   1. The variables which are created inside a class and outside any method are called instance variable.
   2. Instance variable can be access inside class by any of the non-static method directly and can be access by object (instance) outside the class.
   3. Instance variable will be initialized by java automatically if no values provided.
3. **Class/Static Variable (global variable)**
   1. The variables which are created inside class and outside any method with static keyword are classed static/class variable.
   2. Static variables can be access inside class directly in any of the method. And also can be access outside class using only class name and without object of class.
   3. Static/class variable will be initialized by java automatically if no values provided.

**Example**

public class VariableDemo6

{

int a = 10; // Instance Variable

static int c = 30; // static/class Variable

public static void main(String ar[])

{

int b; // Local Variable

}

}

**Default Values**

|  |  |
| --- | --- |
| ***Data Type*** | ***Value*** |
| byte short int long | **0** |
| float double | **0.0** |
| char | **0 or \u0000** |
| boolean | **false** |
| Non-primitive | **null** |

**Note:** These default values are only applicable for a Instance and Static/class variable.

**Operators**

**Arithmetic Operator:** (always return some values except boolean)

+, -, \*, /, %

**Assignment Operator:** (always return some values except boolean)

=, +=, -=, \*=, /=

**Increment/Decrement Operator** (always return some values except boolean)

++, --

To increment and Decrement value by one.

These operators are use to increment/decrement the value and also assign the values.

There are two types of increment/decrement operations can be perform by this which Pre (++a, --a) and post (a++, a--)

Post operation: first assign the value then, increment/decrement the value of the variable.

Pre Operation: first increment/decrement the value of the variable then, assigns the value.

**Relational Operator:** (always returns Boolean (true/false) value)

>, <, >=, <=, ==, !=

These operators are used to compare the values.

These are use in the logical expression.

**Logical Operator**: (return numeric values and also Boolean values)

&, |

**Short Cruciate Operator:** (use for only logical expressions and always return Boolean value)

&&, ||

**Shifting Operator:** (Use for a numeric values)

>> (Right Shift) = Number / 2Bit Shift

<< (Left shift) = Number \* 2Bit Shift

**Comments**

1. To provide an information, write a documentation or to skip any line from the execution the comments are used.
2. There are 3 types of comments
   1. Single line comment
      1. To Comment out a single line

Syntax

// comments line

* 1. Multi line comment
     1. To comment out group line(multiple line)
     2. Syntax:

/\*

Lines

\*/

* 1. Documentation comments 
     1. Use to provide the documentation for the code.
     2. The information which is provided by this comment will be store inside java docs.
     3. Syntax:

/\*\*

lines

\*/

**Task**

1. Write a program display their sum of two number.

2. Write a program to calculate area and perimeter of Square

3. Write a program to calculate area and circumference of Circle.

4. Write a program calculate simple interest.

**Control Flow Statement**

1. To control the flow of java program execution.
2. There are 3 types
   1. Sequential statement execution
      1. This is the default execution of the java code.
   2. Conditional Statement
      1. You can manage the execution of line based on specific condition.
      2. To achieve this can use if statement and its variations, switch cases.
   3. Looping Statement
      1. You can execute a block of lines multiple time.
      2. To Achieve this can use while, do-while, for, enhance for loops.

**Conditional Statement**

1. Conditional Statement is use to execute a specific line or block of line based on condition/scenario.
2. There are multiple ways to achieve this
   1. If conditional statement and its variations like if, if-else, else-if, nested if.
   2. Switch cases.

**If Conditional Statement**

1. **if Statement** 
   1. if is a keyword, it must be followed with a Boolean expression.
   2. The expression which return either true or false is a Boolean expression
   3. The if curly brackets are optional if there is single line of execution is there of if statement. But it is always a good programming practice to use this brackets.
   4. The statement(s) from the if brackets only executes if condition is true.

Syntax:

if(Boolean Expression)

{

Statement(s)

}

1. **if-else statement**
   1. else is also a keyword.
   2. Else cannot be use alone and it must be followed with if condition.
   3. In this structure, if bracket is executes on the condition true, otherwise else statement(s) will executes.
   4. Syntax

if(condition)

{

Statement(s)

}

else

{

Statement(s)

}

**else-if statement**

1. If you wants to check the condition for the else part then can use else if structure.
2. This structure is also known as else-if ladder.
3. In this structure, if any of the condition becomes true then, it will never execute the other conditions.
4. Syntax

if(condition)

{

Statement(s)

}

else if(condition)

{

Statement(s)

}

else if(condition)

{

Statement(s)

}

else

{

Statement(s)

}

**Nested if statement**

1. One if has another if is called nested if.
2. You can also write if structure inside else.
3. Syntax:

if (Condition)

{

if(Condition) {

}

}

**Switch Cases**

1. Switch statement has a cases.
2. switch is a keyword, in switch structure case, break, default keywords are used.
3. Syntax:

switch(Value)

{

case label:

Statement(s)

break;

case label:

Statement(s)

break;

default:

Statement(s)

}

1. Rules to use Switch cases
   1. Switch value must be a byte, short, int, char, enum(JDK 1.5), String(JDK 1.7).
   2. Case labels must be unique.
   3. Case label data type must be same as value data type.
   4. Case labels are must be constant (fixed value).
   5. Break is not a part of syntax (not mandatory), you may get an logically incorrect output if you don’t write break statement.
   6. default can be write at any of the location in switch case. It need not to be at the end of switch.
   7. If multiple cases has similar execution then you can combine the multiple cases.

Task:

Print the following output using Switch case

char ‘M’ or ‘m’ print **Monday**

char ‘T’ or ‘t’ print **Tuesday or Thursday**

char ‘W’ or ‘w’ print **Wednesday**

char ‘F’ or ‘f’ print **Friday**

char ‘S’ or ‘s’ print **Saturday or Sunday**

**Looping statement**

1. These types of statements are used to execute a line of block of line multiple times.
2. There are 3 types of Looping statement
   1. while
   2. do-while
   3. for
   4. Enhance for loop (Used for only a collection)

**While Loop**

1. While loop is use to execute statements multiple time.
2. While loop is pre condition check i.e. it will check the condition before entering inside the while execution body.
3. Syntax:

Variable declaration and initialization (start point of the loop)

while(Boolean-Expression)

{

Statement(s)

Increment/decrement variable

}

**do-while Loop**

1. do-while loop is use to execute a statement multiple time.
2. Do-while is post condition check i.e. it will first execute the execution body once and then check for the condition.
3. In this case loop execution body executes at least once.
4. Syntax

Variable declaration and initialization (start point of the loop)

do

{

Statement(s)

Increment/decrement variable

}

while(Boolean-Expression);

**for loop**

1. For loop I use to execute statement(s) multiple time.
2. If you know start and end point of the loop then for loop will be prefer.
3. Syntax:

for (declaration &initialization **;** Boolean-expression **;** increment/decrement/statement(s))

{

Statement(s)

}

1. The while is divide into 3 parts by 2 semi-colons.
   1. 1st part is optional (can be blank) or can initialize the variable.
   2. 2nd part is optional (can be blank) or can write a Boolean expression. If it kept blank then by default it considers a true.
   3. 3rd part is also optional or can write increment/decrement or even a statements, which must be separated by comma (,)

Task

1. Write a program to print the table of any number using while and for.

Example: if num = 5

5

10

15

20

.

.

.

50

**Nested Looping**

**Nested For Loop**

1. One for loop inside another for loop.
2. These types of for loop are mostly use to work with row and column (table) structure.
3. Syntax:

for ( ; ; ) { ------------🡪 Row

for ( ; ; ) { ------🡪 Column

}

}

Task:

Print the pattern (Hint: pyramid patter user 2 inner foe loop int to manage spaces and another to manage \*)

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \* \*

**Enhance For Loop**

1. In loop is use to iterate values from the collection.
2. In this loop index will maintain internally.
3. This loop is use to iterate all the values from the collection, and you cannot perform indexed based iteration using this loop.
4. Using this loop you can avoid a changes of ArrayIndexOutOfBoundException at runtime.
5. Syntax

for( DataType variable **:** collection )

{

}

**Array**

1. Array is a group of values/object.
2. It’s a group of same type of values/object.
3. Array is fixed in size.
4. To use array you must know the size and the type of values.
5. Array are a non-primitive data type.
6. Array is an indexed based.
7. Index will be created and maintain internally.
8. Index are always start from 0.
9. If you wants set or get (access) the values from array you have to use index.
10. To create Array, have to follow 3 steps
    1. Array Declaration.
    2. Array Instantiation (Object creation).
    3. Array Initialization.
11. Arrays are of different types, like 1-D array, 2-D array, Multi-dimensional (Jagged Array).
12. If you handle index improperly then you will get an ArrayIndexOutOfBoundsException

**1-D Array**

1. In 1-D array can store multiple values in row format.
2. Syntax:

**Array Declaration**

**Data-type identifier[];**

Example: double percent[];

**Array Instantiation (Object Creation)**

**identifier = new Data-type[Size];**

Example: percent = new double[7];

**Array Initialization**

**identifier[index] = value;**

Example: percent[0] = 67.87;

1. While creating object of an array, we must have to assign size of array.
2. While declaration of array you can create array bracket anywhere except before data type.
3. To Access the values from array you have to use index.
4. After instance creation of an array every block of array will be initialize by default value.
5. Total number of values can be get using length function in array.
6. **array.length** this will return the total number of values present inside array.
7. **array.length - 1**  : To get the last index of array.



1. Different Ways to create 1-D Array

int arr[]; // declare

arr = new int[5]; // instance creation

arr[0] = 10; // initialization

arr[1] = 20;

arr[2] = 30;

arr[3] = 40;

arr[4] = 50;

int arr[] = new int[5]; // declare and instance creation

arr[0] = 10; // initialization

arr[1] = 20;

arr[2] = 30;

arr[3] = 40;

arr[4] = 50;

int arr[] = new int[] {10, 20, 30, 40, 50}; // declare ,instance creation and initialization

int arr[] = {10, 20, 30, 40, 50}; // declare and initialization (instance creation happens internally)

**2-D Array**

1. Can store values in row and column structure.
2. Syntax:

**Array Declaration**

**Data-type identifier[][];**

Example: double percent[][];

**Array Instantiation (Object Creation)**

**identifier = new Data-type[Row\_Size][Column\_Size];**

Example: percent = new double[3][4];

**Array Initialization**

**identifier[row\_index] [column\_index] = value;**

Example: percent[0][1] = 67.87;



1. Length in 2-D

percent.length // total number of rows

percent[0].length // total number of values in 0th row

3. Different Option to create 2-D Array

double percent[][];

percent = new double[3][4];

percent[0][0] = 56.78;

percent[0][1] = 77.78;

double percent[][] = new double[3][4];

percent[0][0] = 56.78;

percent[0][1] = 77.78;

double percent[][] = new double[][] { {78, 65, 78, 89}, {78, 87 , 67, 89}, {87, 67, 65, 90} } ;

double percent[][] = { {78, 65, 78, 89}, {78, 87 , 67, 89}, {87, 67, 65, 90} } ;

**Multi-Dimensional Array (Jagged Array)**

1. The Row size is fixed and column size is differ for every row.
2. Example:

double percent[][] = {

{67, 79, 80, 67},

{76, 56},

{56, 78, 99}

};