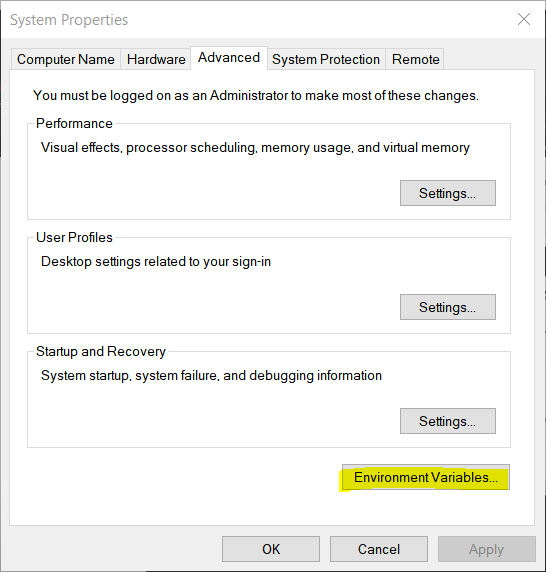
**Core Java**

* **Java:**
  + Java is made by Sun Microsystems in 1995 and later on bought by Oracle and currently Java trademark is held by Oracle. Java is updated every 6 months.
  + Java is mainly used for Mobile development, Web development as well as Enterprise applications. Kotlin, Scala, Groovy come under java technology. Java is the most readable language, we can read code line by line and it is easy to understand.
  + Java has multiple features such as **WORA (Write Once Run Anywhere), multiple thread, Collection API concept, Exception handling and many more**.
  + Java is an object oriented programming language, so it means everything should be in an object and to create an object we need to have a class.
* **Setup Java Environment:**
  + **IDE:** To write Java, we can use Notepad and WordPad for simple programming but to work with java professionally, we can use different **IDEs(Integrated Development Environment)** such as **VS Code(Light weight editor), Eclipse and IntelliJ,** where we can type the code, compile the code, run the code and debug the code.
    - **VS Code URL:** [**VS Code**](https://code.visualstudio.com/)
  + **Compiler:** To compile the java code, we need to install **JDK (Java Development Kit)** 
    - **JDK URL:** [**Oracle JDK v24**](https://www.oracle.com/java/technologies/downloads/%23java24)
  + Once we complete the installation of IDE and JDK, to verify it, we can run “**JAVA --VERSION**” command to “**CMD**” and it will give you version of Java and JDK installed to the system also by running “**JAVAC --VERSION**” to get the current version and confirmed that compiler installed correctly.

A screenshot of a computer program

AI-generated content may be incorrect.

* + While we check the versions and if we get an error then it means that path for java environment has not set up very well.
  + To set up environment variables, follow the following steps:
    - Start > Search “**Edit System Environment Variables**”
    - It will open below window, click on “Environment Variables”.



* + - * Select “path” and click on “**Edit…**” button under “**System variables**”

A screenshot of a computer program

AI-generated content may be incorrect.

* + - * Click on “**New**” and add “**C:\Program Files\Java\jdk1.8.0\_161\bin**” path or copy the path of **“…\Java\jdk1.8.0\_161\bin**” then click “**OK**”.

A screenshot of a computer program

AI-generated content may be incorrect.

* + - * Click on “**New…**” under “**User variables for…**” then add below values, and click on “**OK**” and “OK” for main window:
        + **Variable Name:** JAVA
        + **Variable value:** “**C:\Program Files\Java\jdk1.8.0\_161\bin**” path or copy the path of **“…\Java\jdk1.8.0\_161\bin**”

A screenshot of a computer

AI-generated content may be incorrect.

* **How Java Code Works on Backend:**
  + Java have JVM (Java Virtual Machine), using that we can run java application.
  + Under JVM there is OS (Operating System) comes and under that there is HW (Hardware).
  + Java is a platform independent, means when we write a Java code, it will run on any machine. Only this is required for it is that respective machine need to have JVM.
  + **JVM does not understand the Human readable code or instruction, but it accept only BYTE code. So, to convert this java code into Byte code it uses “Java Compiler”**
  + So, the basic flow is,
    - **Users create Java code > javac compiler compiled code to BYTE code > BYTE code goes on JVM > JMV look for “public static void main(String args[]” syntax to execute the code**.
  + In the real time scenario, there are 100s of files in the java project, so JVM starts with the first file/Main method file or file which contains main method file.
  + While running java code, it also requires some libraries and in java JRE (Java Runtime Environment) fulfill this request. Also, JVM is the part of JRE. JDK is only use by developers.
  + Main method contains one specific signature **“public static void main(String args[])”**

**(NOTE: In java there is only one Main method and only one file which have main method from where java code starts the execution.)**

**A diagram of a computer program

AI-generated content may be incorrect.**

* + As java follow WORA (Write Once-Run Anywhere), once developer develop application using JDK, and to use that application to other machine, we just have to install JRE and JVM (JDK does not required), and we can easily execute the code at any platform/OS.
* **Java Code Basic Code Structure:**
  + Java code file extension is **“.java”.**
  + Java code execution point is Main method of java where **“public static void main(String args[])”** main method comes.
  + To compile the code use **“javac FileName.java”** and it will create **“.class”** file.
  + Run code use **“java FileName”** command.
  + Java Basic Code Syntax:

**A screen shot of a computer program

AI-generated content may be incorrect.**

* **Java Variables:**
  + Variables are used to store values, which can be either provided by the user, predefined or assigned during runtime.
  + **To define values in java, follow below syntax:**
    - DataType VariableName = Value;
    - **The value of variables also called LITERALS.**

**(NOTE: equal to ‘=’ is the assignment operator. It will take the value form right side and assigned to left side in variable.)**

**A screen shot of a computer code

AI-generated content may be incorrect.**

* **Data types in Java:**
  + **There are 2 types of data types in java:**
    - Primitive data type
    - Non- primitive data type
  + **List of Primitive data type:**
    - **Integer**: Stores values without decimal point (0-9)
      * **Subtype of Integer:**
        + **byte**: supports 1 byte.

**Range**: -27 to 27-1 (-128 to 127)

* + - * + **short**: supports 2 bytes.

**Range**: -32,768 to 32,767

* + - * + **int**: supports 4 bytes.

**Range**: -2,147,483,648 to 2,147,483,647

* + - * + **long**: supports 8 bytes.

**Range**:-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

**(Note: While define, need to add “l” at the end of value.** Ex: long l = 12356l;**)**

* + - **Float**: Stores values with decimal point (0.0-9.0)
      * **Subtype of Float:**
        + **float**: supports 4 bytes.

**As Java define double as a default datatype, and we want to define float data type variable then we need to add “*f*” at the end of the value.**

**If we do not add “*f*”, then it will consider as double data type.**

**Ex:** *float fl= 5.6f;*

* + - * + **double**: supports 8 bytes.

**(Note: While defining decimal value variable, by default java supports “double” data type because while doing calculations, double data type store longer digits after decimal point then float.)**

* + - **Character**: Stores only single character (A’, ‘b’)
      * **Subtype of Character:**
        + **char**: supports 2 bytes.
        + **To define a character data type, we need to define value between single quotes (‘H’) because double quotes (“”) are only for String data type.**
        + **Also character data type stores only one single character.**
    - **Boolean**: Stores only either “**True**” or “**False**”
      * **Subtype of Boolean:**
        + **boolean**: supports 1 bit.
        + **In other programming languages, the values of boolean data type is 0 for false and 1 for true but in java the values are “True” or “False”**

**(NOTE: IF WE TRY TO STORE OUT OF RANGE VALES TO THE SMALL DATATYPE THEN IT WILL THROOW AN ERROR. FOR EXAMPLE, AS BYTE STORE (-128 to 127) RANGE AND IF WE STORE “129” IN IT, THEN IT WILL THROW AN ERROR.)**

* + **In integer data type, if we have long numbers, then we can add underscore (\_) between the numbers while defining the values.**
    - **Ex: int num = 10\_00\_00\_000;**
      * **Output = 100000000;**
  + **If we define a alphabet into the character data type then we perform the addition or subtraction operation, it will simply change the alphabet value**
    - **Ex: char c = ‘a’;**

**c++;**

* + - * **Output: b**

**A screenshot of a computer program

AI-generated content may be incorrect.**

* **Data Type Conversion in Java:** *Small -> Large*
  + In java there are two techniques to convert a variable from one data type to another.
    - Type Conversion
    - Type Casting
  + NOTE: We can not perform conversion/casting from char to Boolean datatype, it only supports in the range of numerical conversion between byte to double data type.
  + **Type Conversion:**
    - Convert small data type into large data type without any data loss.
    - It is also known as **Implicit Casting** or **Widening conversion**.
    - Order of implicit conversion:
      * **byte -> short -> int -> long -> float -> double**
    - Example: Converting integer data type to double data type.
    - This type of conversion happens automatically because there is no risk of data loss.

**A computer screen shot of a program

AI-generated content may be incorrect.**

* + **Type Casting:** *Large -> Small*
    - Manually converting a large data type into smaller one.
    - It is also known as **Explicit Casting** or **Narrowing Conversion.**
    - It does not convert automatically because it leads to data loss.
    - Order of Explicit Casting:
      * **double -> float -> long -> int -> short -> byte**
    - Example: converting double data type to integer data type.
    - Syntax:
      * Big DataType Variable = Value;

SmallDataType VariableName = (SmallDataType) Value/BigDataTypeVariable;

**A screen shot of a computer program

AI-generated content may be incorrect.**

**NOTE: Here in this example, we lose 0.99 after converting double to integer value.**

* **Assignment Operators:**
  + List of Operators:
    - Arithmetic Operators
    - Relational Operators
    - Logical Operators
  + **Arithmetic Operators:**
    - **Addition: +**

**A screenshot of a computer program

AI-generated content may be incorrect.**

* + - **Subtraction: -**

**A screen shot of a computer program

AI-generated content may be incorrect.**

* + - **Multiplication: \***

**A screen shot of a computer program

AI-generated content may be incorrect.**

* + - **Division: /**

**A screenshot of a computer program

AI-generated content may be incorrect.**

* + - **Modulus/Remainder: %**

**A screen shot of a computer program

AI-generated content may be incorrect.**

* + **Relational Operators:**
    - Relational operators mainly compare two values or expressions. And it always returns Boolean result (true or false).
    - We can use assignment operator in IF…ELSE conditions, ITERATION LOOPS(FOR, WHILE, DO…WHILE).
    - **Equal to: ==**

**(NOTE: SINGLE EQUAL TO(=) USE FOR ASSIGNMENT OF VALUE WHILE DOUBLE EQUAL TO(==) USE FOR COMPARITION OF VALUES.)**

* + - **Not Equal to: !=**
    - **Greater than and equal to: > >=**
    - **Less than and equal to: < <=**

**A screen shot of a computer program

AI-generated content may be incorrect.**

* + **Logical Operators:**
    - It is mainly used to combine multiple conditions and return a Boolean result (true or false).
    - **Logical AND: &&**
      * When we want all conditions to be true, we need to use the logical AND operator.
    - **Logical OR: ||**
      * When at least one condition among all conditions needs to be true, we use the logical OR operator.
    - **Logical NOT: !**
      * While using NOT operator whatever result we get, it will give opposite of it. For example, if we get the answer TRUE and use the ! OR operator, then it will give FALSE answer.

**A screen shot of a computer program

AI-generated content may be incorrect.**

* **Conditional Statements:**
  + Conditional Statements are used to make decisions or execute a block of code based on certain conditions. It executes a specific block of code on whether a condition is true or false.
  + We can add multiple conditions in the single IF statement using logical operators.
    - EX.: if (x > 10 && x < 20)
  + **Types of conditional statements in Java:**
    - **if statement:**
      * There will be only one block of IF condition. If condition will true for that block the it will execute it otherwise it will move forward to other block.
      * Syntax:
        + if (Condition) { …. }

**A screen shot of a computer program

AI-generated content may be incorrect.**

* + - **if…else statement:**
      * There will be only one and IF and one ELSE block of code. If condition true for IF block then it will skip ELSE block and complete the execution.
      * While condition false for IF block, then it will execute the ELSE block and complete the execution.
      * Syntax:
        + if (Condition) {

------------------

} else{

------------------

}

**A screen shot of a computer program

AI-generated content may be incorrect.**

* + - **Ternary Operator:**
      * The Ternary Operator is a short-hand version of if-else statement.
      * It is used to assign a value based on a condition in a single line.
      * Syntax:
        + Variable = (Condition) ? Value\_IF\_True : Value\_IF\_False;
        + Condition = A Boolean expression like a > b, x == y
        + ? = Separates condition from true value
        + Value\_IF\_True = Value assigned if condition is true
        + : = separates true value and false value
        + Value\_IF\_False = Value assigned if condition is false
      * It is only use for simple IF-Else condition and mostly use to assign the values.

A screen shot of a computer program

AI-generated content may be incorrect.

* + - **if…else if…else statement:**
      * If there will be multiple conditions need to check one by one then we can use IF…ELSEIF…ELSE condition block.
      * Syntax:
        + if (Condition) {

------------------

} else if (Condition) {

------------------

} else {

------------------

}

**A screen shot of a computer program

AI-generated content may be incorrect.**

* + - **Nested if statement:**
      * A Nested if Statement means an if statement inside another if statement.
      * It is used when there are **multiple conditions** to check, and one condition depends on another condition.
      * Syntax:
        + if (Condition1) {

if (Condition2){

--------------------

}else{

-------------------

}

} else{

------------------

}

A screen shot of a computer program

AI-generated content may be incorrect.

* + - **Switch statement:**
      * A switch statement is used to execute one block of code among many options based on the value of a variable.
      * It is an **alternative to multiple if-else-if statements** when you are checking the same variable for **multiple constant values**.
      * Syntax:
        + switch (expression/condition){

case value1:

//Code for execute for case value 1

break;

case value2:

//Code for execute for case value 2

break;

case value3:

//Code for execute for case value 3

break;

…. …. …. …. …. …. ….

default:

//Code for execute if none of the case match

break;

}

A screen shot of a computer program

AI-generated content may be incorrect.

* **Looping Statements:**
  + Looping Statements are mainly used to repeat a block of code multiple times. Instead of writing the same code again and again, we can use loops.
  + For example, if we want to print numbers from 1 to 100, using a loop we can done this task withing few statements instead of writing same 100 print statements.
  + **Advantages of Loops:**
    - Reduce code duplication
    - Perform repetitive tasks like printing numbers, processing arrays, or reading input
    - Make programs more efficient and readable
  + **List of Loops in Java:**
    - For Loop
    - While Loop
    - Do…While Loop
    - Enhanced For Loop or For-Each Loop
  + ***for* Loop:**
    - For loop is mainly used when there is a number of iterations are known in advance.
    - For loop syntax first includes initialization, condition and increment/decrement in the single line then loop body will come.
    - **Syntax:**
      * for (initialization; condition; addition/subtraction)

{ //Code to execute }

A screen shot of a computer program

AI-generated content may be incorrect.

* + - For “*for*” Loop, if we want to skip any of the components (initialization, condition or increment/decrement) we can write outside of loop and skip into the for-loop bracket.
    - Syntax:
      * Initialization;

for (; //Condition; ){

//Code to execute

Increment/Decrement;

}

**A screen shot of a computer program

AI-generated content may be incorrect.**

* + ***while* Loop:**
    - While loop is mainly used when the number of iterations are unknown before execution, and the loop should be continued as long as condition is true.
    - In the while loop, condition is checked before executing the loop body.
    - So, it means that to execute the body of the loop, the condition must be fulfilled. Otherwise, it will not execute the loop body and move to the next statement.
    - While loop mainly use to read data from a file or read a data from the database because we don’t know what the exact number of characters file will be contains or don’t know how many lines of data will have in database. So, for this scenarios while loop is the best option.
    - Syntax:
      * Initialization; //If required

while (Condition){

//Code to execute

Increment/Decrement; //if required

}

**A screen shot of a computer

AI-generated content may be incorrect.**

* + ***do-while* Loop:**
    - Do while loop is similar to while loop but one major difference is in do-while Loop, first loop body will execute then check the condition.
    - This means that loop body will execute at least once, even if the condition is false.
    - **Syntax:**
      * Initialization;

do{

//Code to Execute.

} while (Condition);

A screen shot of a computer program

AI-generated content may be incorrect.

* + ***for-each* Loop:**
* **Classes and Objects:**
  + Java’s main feature is that Java is the Object-Oriented Programing Language.
  + **Class:**
    - In java developer will create a blueprint or class and JVM(Java Virtual Machine) creates an object of classes.
    - Developer creates class/Blueprint > .class file get compiled and creates a byte code > Byte code goes to JVM where will get the object
    - Class is a collection of elements, methods and variables.
    - Definition:
      * A class is a user-defined data type that acts as a blueprint for creating objects.
    - The class contains the attributes below:
      * Fields/Variables: that holds data
      * Methods: that perform action
    - Syntax:
      * class ClassName {

//Define Fields/Variables

dataType variableName;

//Methods

returnType(void, int, double, etc) methodName() {

//code block

}

}

* + **Object:**
    - Object is an instance of a class, and it is created using the “*new*” keyword.
    - Syntax:
      * ClassName objectName = new ClassName();
    - We can also only define a reference variable of class and later create an object. Follow the syntax below for it:
      * ClassName referenceVariable;

referanceVariable = new ClassName();

A screen shot of a computer program

AI-generated content may be incorrect.

**A computer screen shot of a program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

* **Polymorphism – Method Overloading and Method Overriding:**
  + Polymorphism is a feature of Object-Oriented Programming that allows one method, class or interface to behave differently based on the object that is calling it.
  + Polymorphism in Java is the ability of an object to take many forms, and it allows the same method or object to behave differently based on the context.
  + **There are two types of Polymorphism:**
    - Method Overloading – Compile time Polymorphism
    - Method Overriding – Run time Polymorphism
  + **Method Overloading – Compile time Polymorphism** 
    - In the overloading method, when there are multiple methods in the **same class have the same name but different parameters**. (Return type can be different)
    - It occurs within the same class only and also called Compile time polymorphism.
    - Class has multiple methods with the same name but different parameters.
      * Ex. class Addition{

public int add(int a, int b) {return a + b; }

public int add(int a, int b, int c) {return a + b + c; }

}

**A screenshot of a computer program

AI-generated content may be incorrect.**

* + **Method Overriding – Run time Polymorphism** 
    - In the inheritance when the **parent’s class and the child class have same method name with same parameters.**
    - Method overriding occurs when a child class/subclass provides a specific implementation of method that is already defined in its parent class.
    - It only happens in the inheritance.
    - **In method overriding, Method name, Return type and Parameters must be exactly same.**
    - It is also known as Runtime polymorphism and allows dynamic method dispatch.
    - The method in the child class **replaces** the one from the parent class during execution.
    - We can add **“@Override”** annotation above the overridden method in the child class.
    - Whenever we create an object of a child class and call an overridden method, the child class method overrides the parent class method because it has the same name and signature. As a result, the parent method is replaced by the child method at runtime.

A screen shot of a computer program

AI-generated content may be incorrect.