

Introduction to Java, Basics of Java Programming and OOPs

S. No.	Context	Page No.
1	INTRODUCTION TO JAVA	2 - 5
	<ul style="list-style-type: none">• Features of Java	
	<ul style="list-style-type: none">• The Java Virtual Machine	
	<ul style="list-style-type: none">• API Document	
	<ul style="list-style-type: none">• Starting a Java Program	
2	BASICS OF JAVA PROGRAMMING	6 - 13
	<ul style="list-style-type: none">• Data Types	
	<ul style="list-style-type: none">• Operators	
	<ul style="list-style-type: none">• Control Statements	
	<ul style="list-style-type: none">• Input and Output	
	<ul style="list-style-type: none">• Arrays and Strings	
3	OOPs	14
	<ul style="list-style-type: none">• Classes and Objects	
	<ul style="list-style-type: none">• Encapsulation	
	<ul style="list-style-type: none">• Abstraction	
	<ul style="list-style-type: none">• Inheritance	
	<ul style="list-style-type: none">• Polymorphism	

INTRODUCTION TO JAVA

Features of Java:

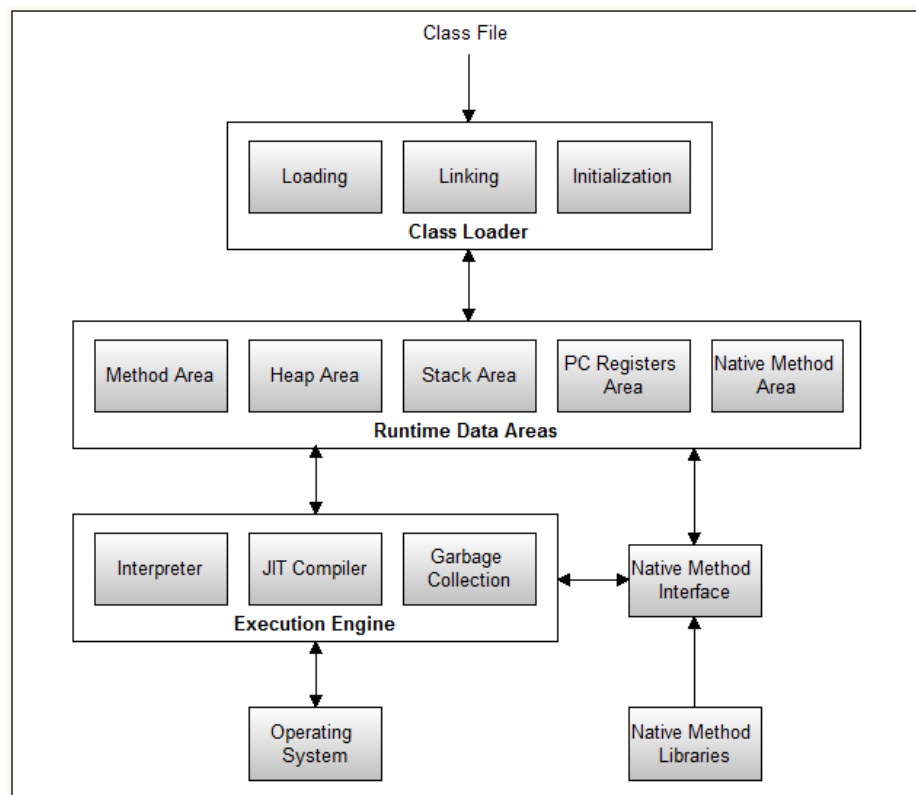
1. **Simple** - Java is simple programming language. Firstly, the difficult concepts of C and C++ have been omitted in Java. For example, the concepts of pointers, which is very difficult for both learners and programmers, has been completely eliminated from Java. Next, Java soft people maintained the same syntax of C and C++ in Java, so that a programmer who knows C or C++ will find Java already familiar.
2. **Object Oriented** - Java is an object-oriented programming language. This means Java programs use Objects and Classes. An object is anything that really exists in the world and can be distinguished from others. A group of objects exhibiting same behavior will come under the same group called a class.
3. **Distributed** - Information is distributed on various computers on a network. Using Java, we can write programs, which capture information and distribute it to the clients.
4. **Robust** - Robust means strong. Java programs are strong and they don't crash easily like a C or C++ program. Java has got excellent exception handling and memory management.
5. **Secure** - Security problems like eavesdropping, tampering, impersonation and virus threats can be eliminated or minimized by using java on Internet.
6. **System Independence** - Java's byte code is not machine dependent. It can be run on any machine with any processor and any operating system.
7. **Portability** - If a program yields the same result on every machine, then that program is called portable. Java programs are portable.
8. **Interpreted** - Java programs are compiled to generate the byte code. This byte code can be downloaded and interpreted by the interpreter in JVM. If we take any other language, only an interpreter or a compiler is used for execute the programs. But in Java., we use both compiler and interpreter for the execution.
9. **High Performance** - The-problem with interpreter inside the JVM is that it is slow. Because of this, Java programs used to run slow. To overcome this problem, along with the interpreter, Java Soft people have introduced JIT (Just in Time) compiler, which enhances the speed of execution. So now in JVM, both interpreter and JIT compiler work together to run the program.
10. **Multithreaded** - A thread represents an individual process to execute a group of statements. JVM uses several threads to execute different blocks of code. Creating multiple threads is called 'multithreaded'.

Introduction to Java, Basics of Java Programming and OOPs

11. **Scalability** - Java platform can be implemented on a wide range of computers with varying levels of resources-from embedded devices to mainframe computers. This is possible because Java is compact and platform independent.
12. **Dynamic** - Before the development of Java, only static text used to be displayed-in the browser. But when James Gosling demonstrated an animated atomic molecule where the rays are moving and stretching, the viewers were dumbstruck. This animation was done using an applet program, which are the dynamically interacting programs on Internet.

The Java Virtual Machine:

Java Virtual Machine (JVM) is the heart of entire Java program execution process. It is responsible for taking the (.class) file and converting each byte code instruction into the machine language instruction that can be executed by the microprocessor.



First of all, the (.java) programs converted into a (.class) file consisting of byte code instructions by the java compiler. Remember, this java compiler is outside the JVM: Now this .class file is given to the JVM.

Introduction to Java, Basics of Java Programming and OOPs

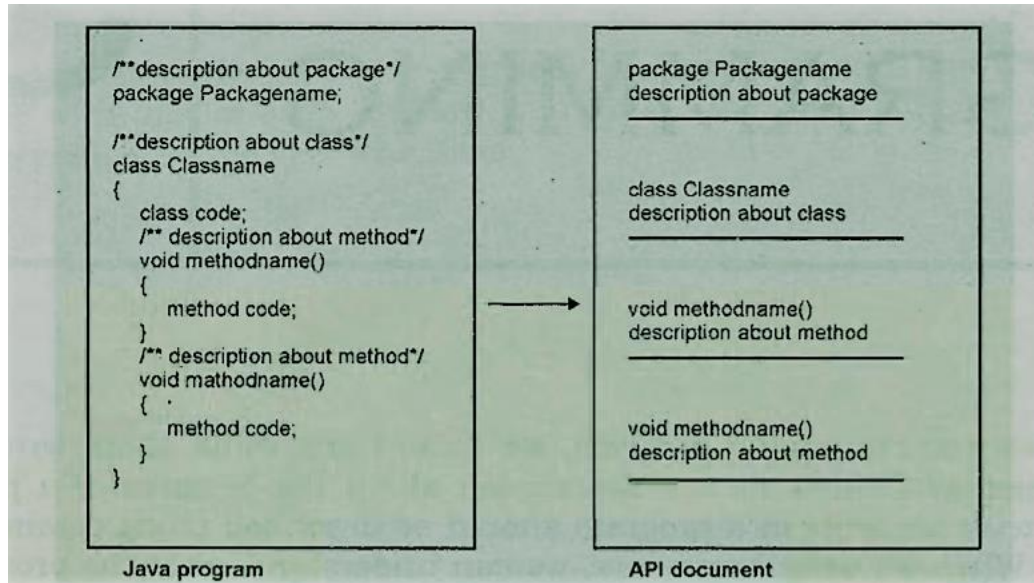
In JVM, there is a module (or program) called class loader sub system, which performs the following functions:

- First of all, it loads the. class file into memory.
- Then it verifies whether all byte code instructions are proper or not. If it finds any instruction suspicious, the execution is rejected immediately.
- If the byte instructions are proper, then it allocates necessary memory.to execute the program.

Execution engine contains interpreter and JIT (Just in Time) compiler, which are responsible for converting the byte code instructions into machine code so that the processor will execute them. Most of the JVM implementations use both the interpreter and JIT compiler simultaneously to convert the byte code.

API Document:

The API document generated from the. java program is similar to a help file where all the features are available with their descriptions. The user can refer to any feature in this file and get some knowledge regarding how he can use it in his program. To create an API document, we should use a special compiler called javadoc compiler.

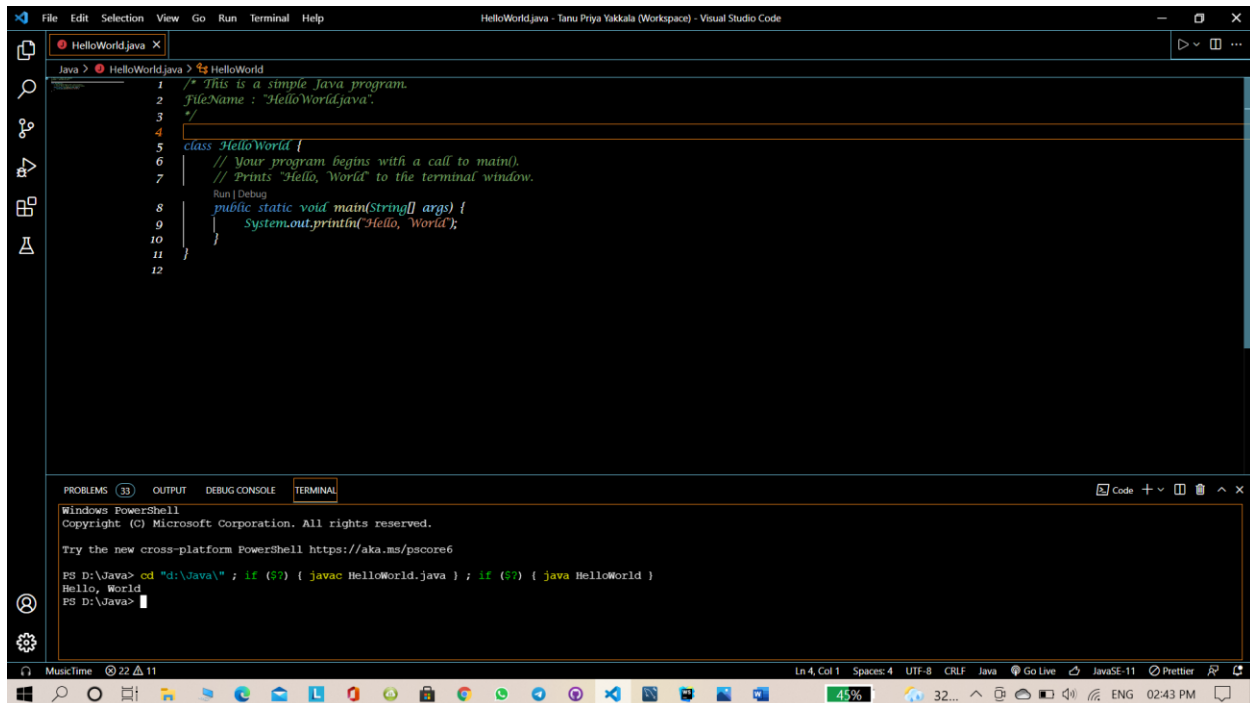


Introduction to Java, Basics of Java Programming and OOPs

Starting a Java Program:

The process of Java programming can be simplified in three steps:

- Create the program by typing it into a text editor and saving it to a file - HelloWorld.java.
- Compile it by typing “javac HelloWorld.java” in the terminal window.
- Execute (or run) it by typing “java HelloWorld” in the terminal window.



The screenshot displays the Visual Studio Code interface. The editor window shows the file `HelloWorld.java` with the following code:

```
1  /* This is a simple Java program.
2  * File Name : "HelloWorld.java".
3  */
4
5  class HelloWorld {
6      // Your program begins with a call to main().
7      // Prints "Hello, World" to the terminal window.
8      // Run | Debug
9      public static void main(String[] args) {
10         System.out.println("Hello, World");
11     }
12 }
```

The bottom panel shows the `TERMINAL` tab with the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

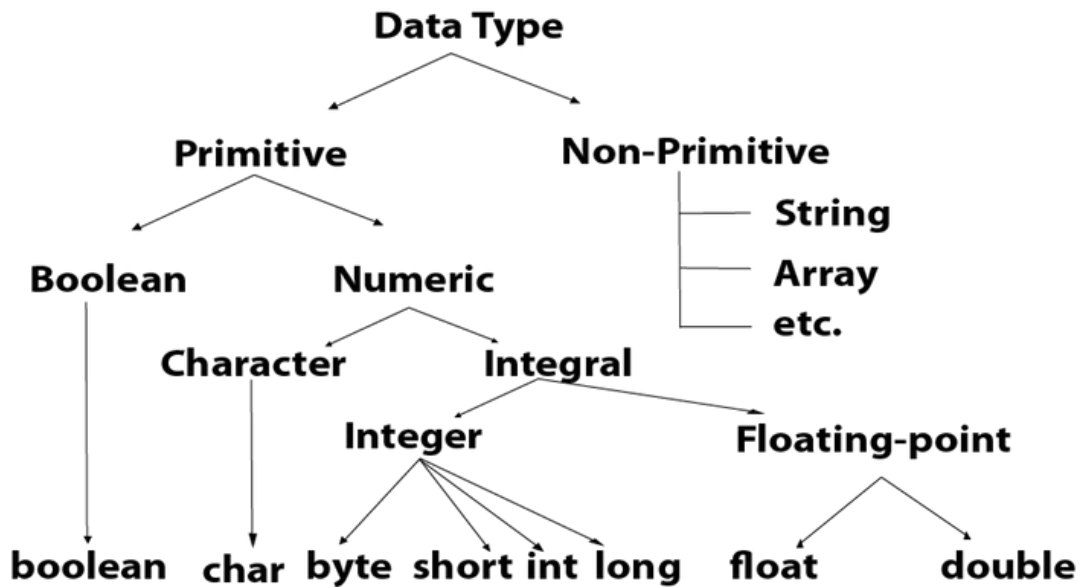
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Java> cd "d:\Java\" ; if ($?) { javac HelloWorld.java } ; if ($?) { java HelloWorld }
Hello, World
PS D:\Java>
```

The status bar at the bottom indicates the current file is `Ln 4, Col 1`, using `UTF-8` encoding with `CRLF` line endings. The system tray shows the date and time as `02:43 PM`.

Introduction to Java, Basics of Java Programming and OOPs

Basics of Java Programming



Data Type	Default Value	Default size
Boolean	false	1 bit
char	'\u0000'	2 bytes
byte	0	1 byte
short	0	2 bytes
int	0	4 bytes
long	0L	8 bytes
float	0.0f	4 bytes
double	0.0d	8 bytes

Introduction to Java, Basics of Java Programming and OOPs

Boolean - The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions. The Boolean data type specifies one bit of information, but its "size" can't be defined precisely.

Example: Boolean one = false

Byte - The byte data type is an example of primitive data type. It is an 8-bit signed two's complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128 and maximum value is 127. Its default value is 0.

Example: byte a = 10, byte b = -20

Short - The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.

Example: short s = 10000, short r = -5000

Int - The int data type is a 32-bit signed two's complement integer. Its value-range lies between -2,147,483,648 (-2^{31}) to 2,147,483,647 ($2^{31} - 1$) (inclusive). Its minimum value is -2,147,483,648 and maximum value is 2,147,483,647. Its default value is 0.

Example: int a = 100000, int b = -200000

Long - The long data type is a 64-bit two's complement integer. Its value-range lies between -9,223,372,036,854,775,808 (-2^{63}) to 9,223,372,036,854,775,807 ($2^{63} - 1$) (inclusive). Its minimum value is -9,223,372,036,854,775,808 and maximum value is 9,223,372,036,854,775,807. Its default value is 0. The long data type is used when you need a range of values more than those provided by int.

Example: long a = 100000L, long b = -200000L

Float - The float data type is a single-precision 32-bit floating point. Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating-point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F.

Example: float f1 = 234.5f

Introduction to Java, Basics of Java Programming and OOPs

Double - The double data type is a double-precision 64-bit. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.

Example: double d1 = 12.3

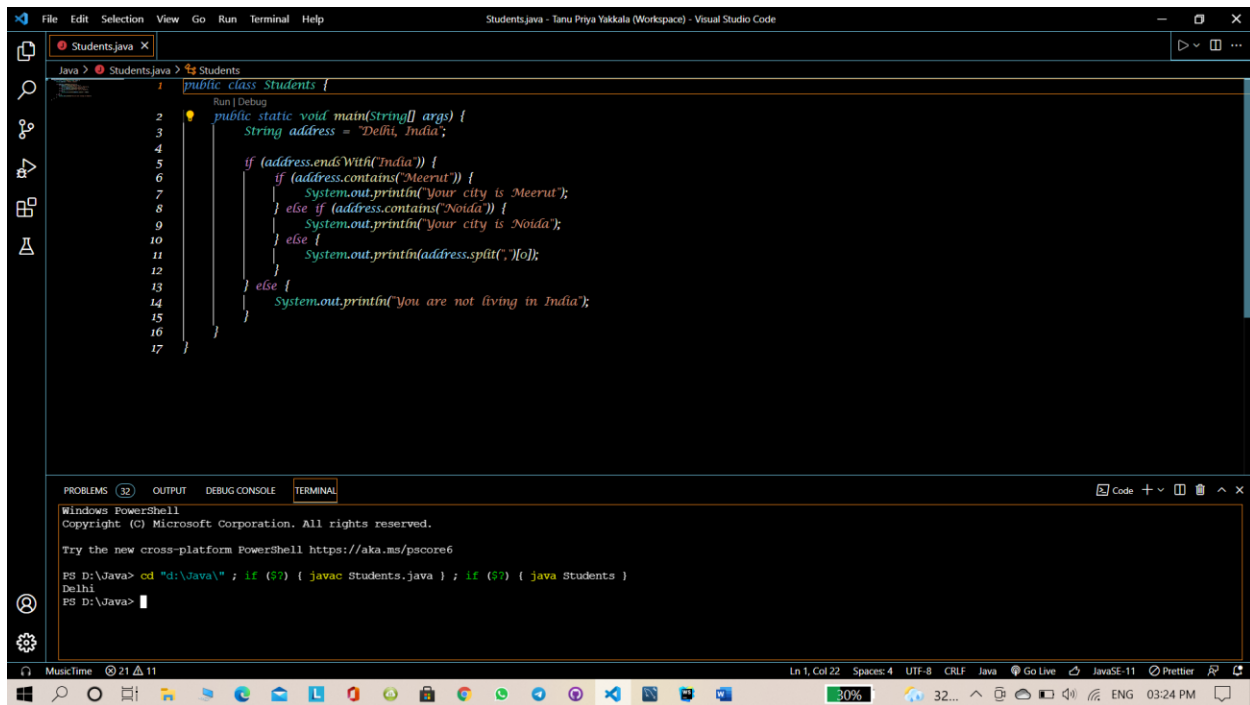
Char - The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive). The char data type is used to store characters.

Example: char letter = 'A'

Operators:

Operator	Category	Precedence
Unary Operator	postfix	expression++ expression--
	prefix	++expression --expression +expression -expression ~!
Arithmetic Operator	multiplication	* / %
	addition	+ -
Shift Operator	shift	<< >> >>>
Relational Operator	comparison	< > <= >= instanceof
	equality	== !=
Bitwise Operator	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	
Logical Operator	logical AND	&&
	logical OR	
Ternary Operator	ternary	? :
Assignment Operator	assignment	= += -= *= /= % = &= ^= = <<=
		>>= >>>=

Introduction to Java, Basics of Java Programming and OOPs

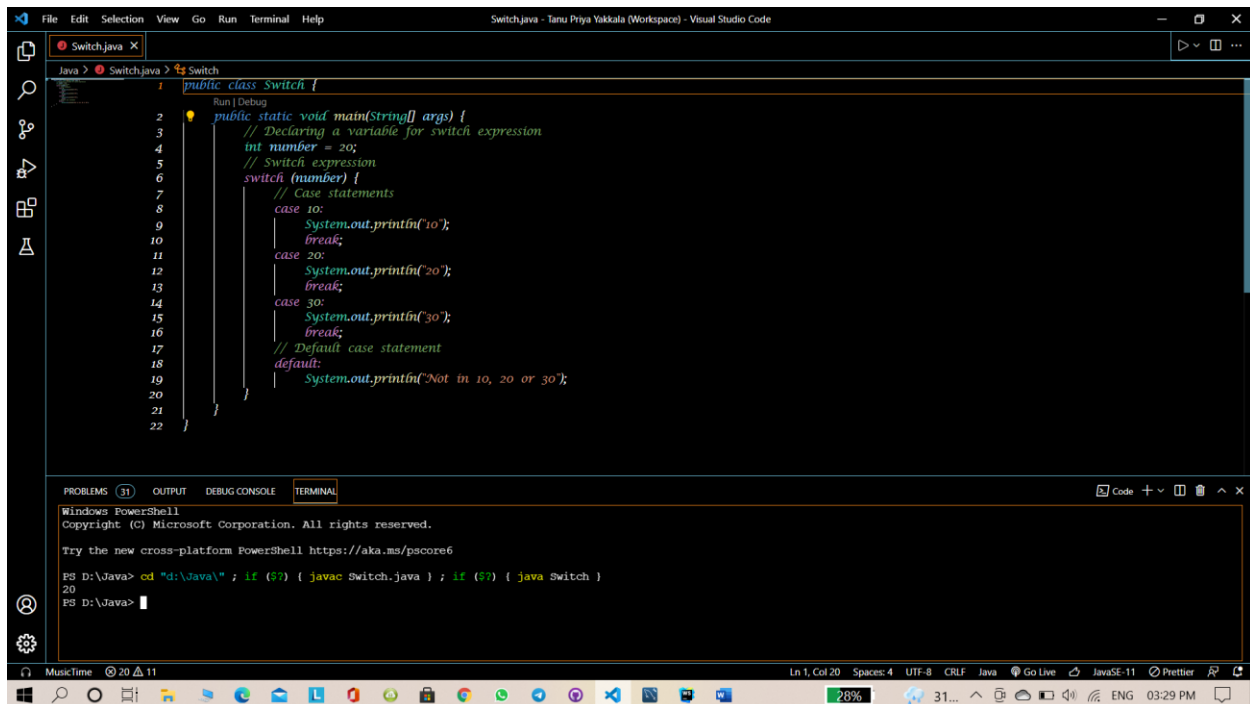


The screenshot shows the Visual Studio Code editor with a file named `Students.java` open. The code defines a `public class Students` with a `main` method that takes a `String[] args` parameter. Inside the `main` method, a `String address = "Delhi, India";` is declared. The code then uses a series of `if` statements to check the address: `if (address.endsWith("India"))`, `if (address.contains("Meerut"))`, `else if (address.contains("Noida"))`, and `else`. Each condition is followed by a `System.out.println` statement. The `else` block prints `"You are not living in India";`. The terminal at the bottom shows the command `cd "d:\Java\" ; if ($?) { javac Students.java } ; if ($?) { java Students }` being executed, and the output `Delhi` is displayed.

```
1 public class Students {
2     public static void main(String[] args) {
3         String address = "Delhi, India";
4
5         if (address.endsWith("India")) {
6             if (address.contains("Meerut")) {
7                 System.out.println("Your city is Meerut");
8             } else if (address.contains("Noida")) {
9                 System.out.println("Your city is Noida");
10            } else {
11                System.out.println(address.split(",")[0]);
12            }
13        } else {
14            System.out.println("You are not living in India");
15        }
16    }
17 }
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell <https://aka.ms/pscore6>
PS D:\Java> cd "d:\Java\" ; if (\$?) { javac Students.java } ; if (\$?) { java Students }
Delhi
PS D:\Java>

5. Switch Statement:

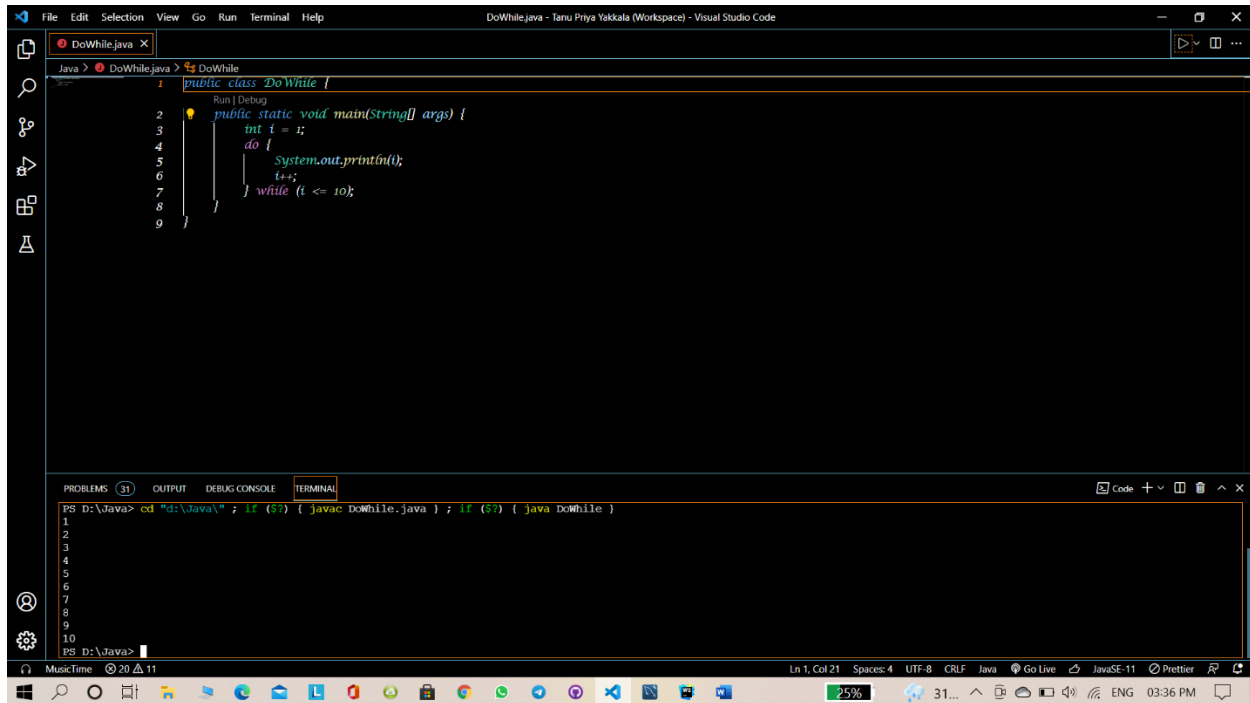


The screenshot shows the Visual Studio Code editor with a file named `Switch.java` open. The code defines a `public class Switch` with a `main` method that takes a `String[] args` parameter. Inside the `main` method, an `int number = 20;` is declared. The code then uses a `switch (number) {` statement with three `case` statements: `case 10:`, `case 20:`, and `case 30:`. Each case is followed by a `System.out.println` statement and a `break;` statement. A `default:` case is also included, which prints `"Not in 10, 20 or 30";`. The terminal at the bottom shows the command `cd "d:\Java\" ; if ($?) { javac Switch.java } ; if ($?) { java Switch }` being executed, and the output `20` is displayed.

```
1 public class Switch {
2     public static void main(String[] args) {
3         // Declaring a variable for switch expression
4         int number = 20;
5         // Switch expression
6         switch (number) {
7             // Case statements
8             case 10:
9                 System.out.println("10");
10                break;
11            case 20:
12                System.out.println("20");
13                break;
14            case 30:
15                System.out.println("30");
16                break;
17            // Default case statement
18            default:
19                System.out.println("Not in 10, 20 or 30");
20        }
21    }
22 }
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell <https://aka.ms/pscore6>
PS D:\Java> cd "d:\Java\" ; if (\$?) { javac Switch.java } ; if (\$?) { java Switch }
20
PS D:\Java>

Introduction to Java, Basics of Java Programming and OOPs



```
1 public class DoWhile {
2     public static void main(String[] args) {
3         int i = 1;
4         do {
5             System.out.println(i);
6             i++;
7         } while (i <= 10);
8     }
9 }
```

```
PS D:\Java> cd "d:\Java\" ; if ($?) { javac DoWhile.java } ; if ($?) { java DoWhile }
1
2
3
4
5
6
7
8
9
10
PS D:\Java>
```

Arrays and Strings:

Arrays - Array can contain primitives (int, char, etc.) as well as object (or non-primitive) references of a class depending on the definition of the array. In case of primitive data types, the actual values are stored in contiguous memory locations.

40	55	63	17	22	68	89	97	89
0	1	2	3	4	5	6	7	8

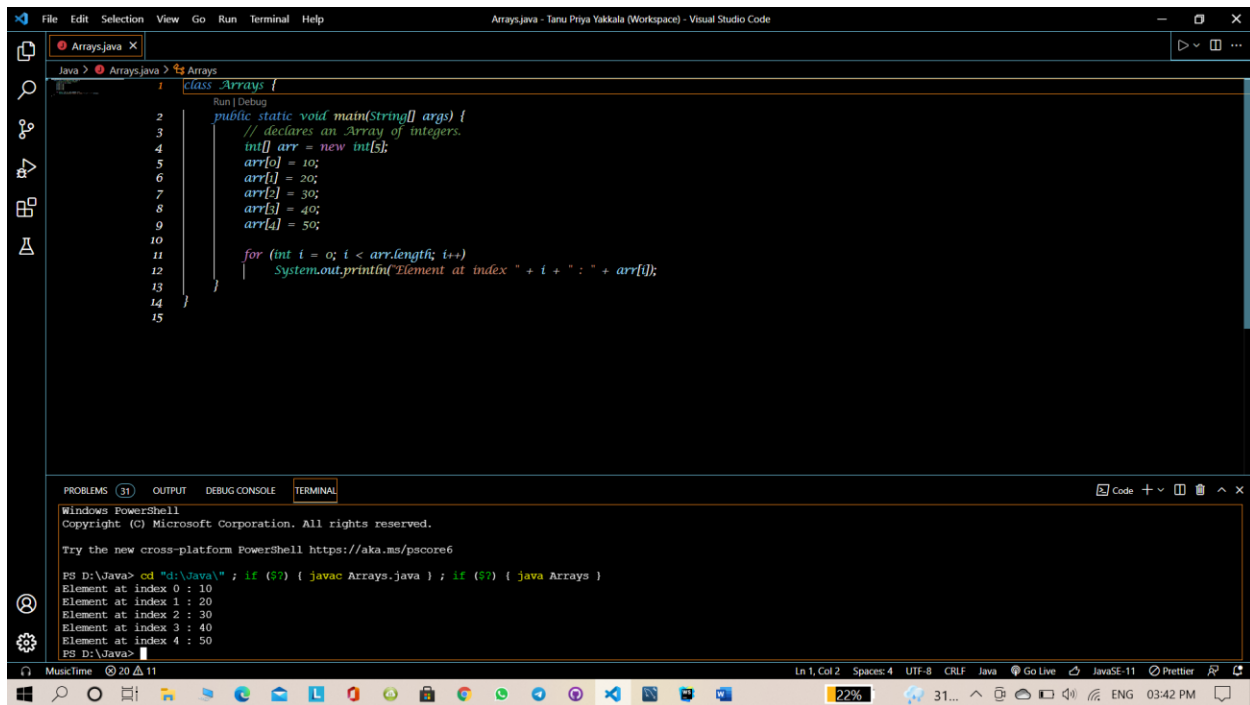
<- Array Indices

Array Length = 9

First Index = 0

Last Index = 8

Introduction to Java, Basics of Java Programming and OOPs



```
File Edit Selection View Go Run Terminal Help
Arrays.java - Tanu Priya Yakkala (Workspace) - Visual Studio Code

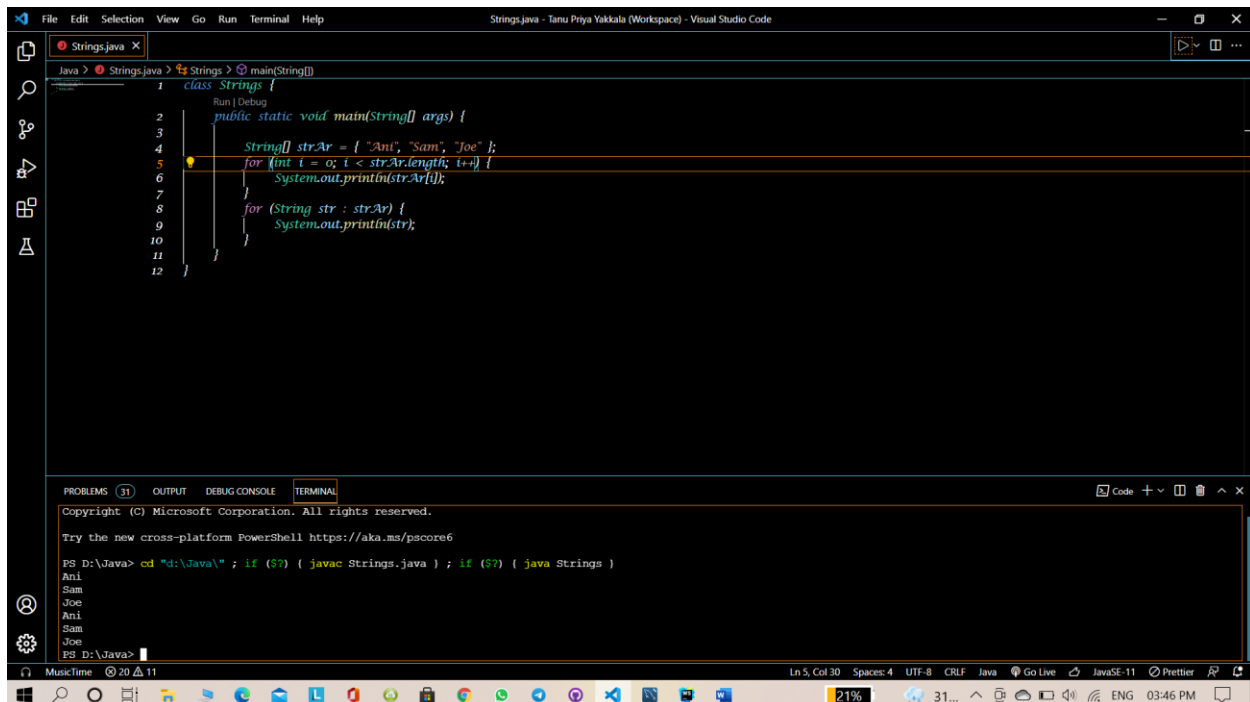
Java > Arrays.java > Arrays
1 class Arrays {
2     public static void main(String[] args) {
3         // declares an Array of integers.
4         int[] arr = new int[5];
5         arr[0] = 10;
6         arr[1] = 20;
7         arr[2] = 30;
8         arr[3] = 40;
9         arr[4] = 50;
10
11         for (int i = 0; i < arr.length; i++)
12             System.out.println("Element at index " + i + " : " + arr[i]);
13     }
14 }
15

PROBLEMS (31) OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS D:\Java> cd "d:\Java\"; if ($?) { javac Arrays.java }; if ($?) { java Arrays }
Element at index 0 : 10
Element at index 1 : 20
Element at index 2 : 30
Element at index 3 : 40
Element at index 4 : 50
PS D:\Java>
```

Strings - A String is a sequence of characters. Generally, a string is an immutable object, which means the value of the string cannot be changed. The String Array works similarly to other data types of Arrays.



```
File Edit Selection View Go Run Terminal Help
Strings.java - Tanu Priya Yakkala (Workspace) - Visual Studio Code

Java > Strings.java > Strings > main(String[])
1 class Strings {
2     public static void main(String[] args) {
3         String[] strAr = { "Ani", "Sam", "Joe" };
4         for (int i = 0; i < strAr.length; i++) {
5             System.out.println(strAr[i]);
6         }
7         for (String str : strAr) {
8             System.out.println(str);
9         }
10     }
11 }
12 }

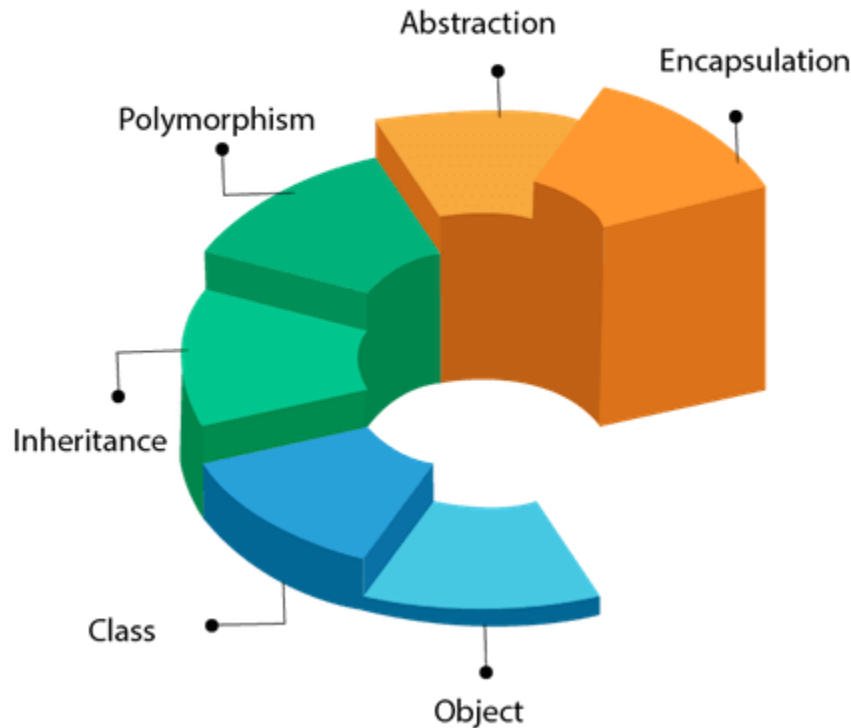
PROBLEMS (31) OUTPUT DEBUG CONSOLE TERMINAL
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS D:\Java> cd "d:\Java\"; if ($?) { javac Strings.java }; if ($?) { java Strings }
Ani
Sam
Joe
Ani
Sam
Joe
PS D:\Java>
```

OOPs (Object Oriented Programming System)

OOPs (Object-Oriented Programming System)



1. **Classes and Objects:** Java is an object-oriented programming language. This means Java programs use Objects and Classes. An object is anything that really exists in the world and can be distinguished from others. A group of objects exhibiting same behavior will come under the same group called a class.
2. **Encapsulation:** Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines. A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.
3. **Abstraction:** Hiding internal details and showing functionality is known as abstraction. For example, phone call, we don't know the internal processing.
4. **Inheritance:** When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
5. **Polymorphism:** If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc. In Java, we use method overloading and method overriding to achieve polymorphism.