



LAB - 1

Java Programming Lab

Topics Covered

Basic concept of Java programming, Compilation and Execution Process,
Data Types, operators, Reading user input, Strings

Syeda Reeha Quasar

14114802719

4C7

What is Java?

Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was Oak. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

Application

According to Sun, 3 billion devices run Java. There are many devices where Java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, javatpoint.com, etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.

Types of Java Applications

There are mainly 4 types of applications that can be created using Java programming:

1) Standalone Application

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine.

Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

2) Web Application

An application that runs on the server side and creates a dynamic page is called a web application. Currently, Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.

3) Enterprise Application

An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.

4) Mobile Application

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

Java Platforms / Editions

There are 4 platforms or editions of Java:

1) Java SE (Java Standard Edition)

It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

2) Java EE (Java Enterprise Edition)

It is an enterprise platform that is mainly used to develop web and enterprise applications. It is built on top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.

3) Java ME (Java Micro Edition)

It is a micro platform that is dedicated to mobile applications.

4) JavaFX

It is used to develop rich internet applications. It uses a lightweight user interface API.

EXPERIMENT – 1.1

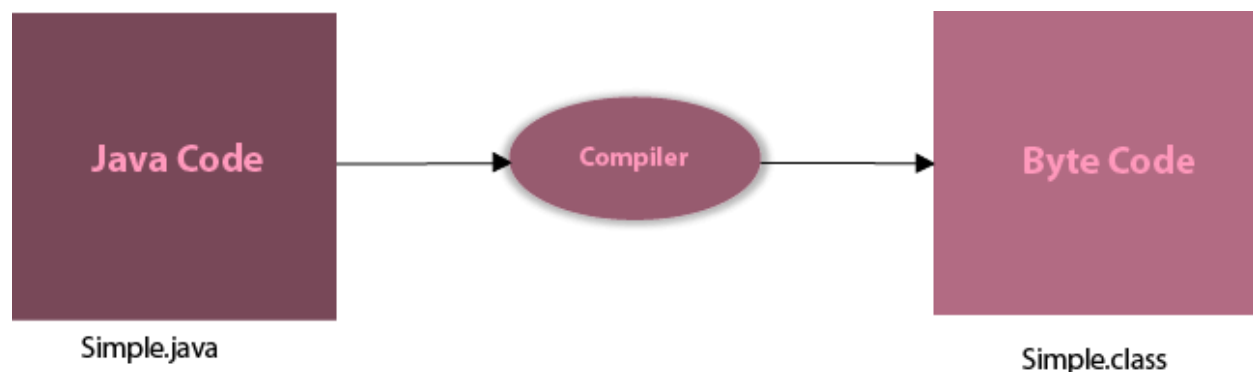
Aim:

Write a program to print “Hello World” on the screen.

Theory:

For executing any Java program, the following software or application must be properly installed.

- Install the JDK if you don't have installed it, [download the JDK](#) and install it.
- Set path of the jdk/bin directory. <http://www.javatpoint.com/how-to-set-path-in-java>
- Create the Java program
- Compile and run the 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.

Concepts:

- **class** keyword is used to declare a class in Java.

- **public** keyword is an access modifier that represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The `main()` method is executed by the JVM, so it doesn't require creating an object to invoke the `main()` method. So, it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **`String[] args`** or **`String args[]`** is used for command line argument. We will discuss it in coming section.
- **`System.out.println()`** is used to print statement. Here, `System` is a class, `out` is an object of the `PrintStream` class, `println()` is a method of the `PrintStream` class. We will discuss the internal working of `System.out.println()` statement in the coming section.

Source Code:

```
public class FirstProgram {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

Output:

```
PS E:\sem 5\java> cd "e:\sem 5\java\" ; if ($?) { javac FirstProgram.java } ; if ($?) { java FirstProgram }  
Hello World!
```



EXPERIMENT – 1.2

Aim:

WAP that convert string to character using toString() and valueOf() .

Theory:

Given a string, the task is to convert this string into a character array in Java.

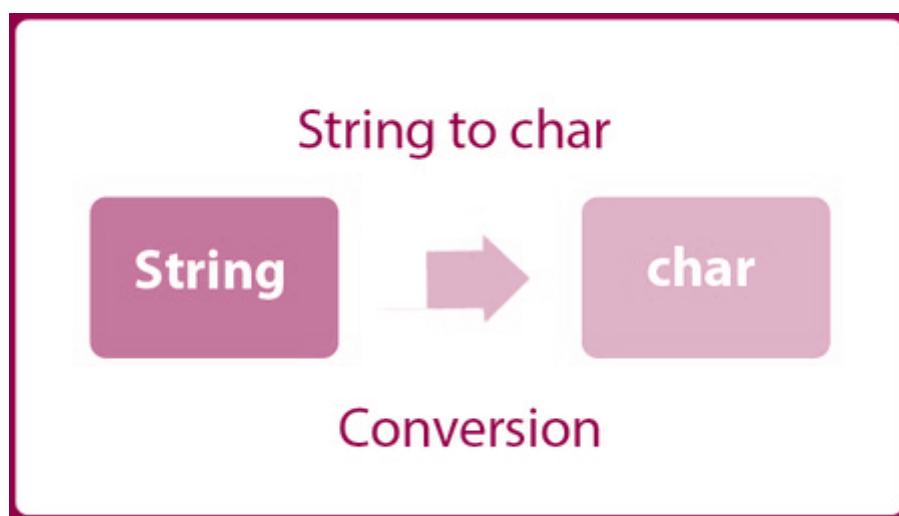
Examples:

Input: Hello World

Output: [H, e, l, l, o, , W, o, r, l, d]

Method : Naïve Approach

- Step 1: Get the string.
- Step 2: Create a character array of the same length as of string.
- Step 3: Traverse over the string to copy character at the i'th index of string to i'th index in the array.
- Step 4: Return or perform the operation on the character array.



Concepts:

- **class** keyword is used to declare a class in Java.
- **public** keyword is an access modifier that represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main() method is executed by the JVM, so it doesn't require creating an object to invoke the main() method. So, it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **String[] args** or **String args[]** is used for command line argument. We will discuss it in coming section.
- **System.out.println()** is used to print statement. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class. We will discuss the internal working of System.out.println() statement in the coming section.
- **charAt()** method returns a single character of specified index. The signature of charAt() method is given below:
public char charAt(int index)

Source Code:

```
public class stringToChar {  
    public static void main(String args[]) {  
        String s = "hello World!";  
        for (int i = 0; i < s.length(); i++) {  
            char c = s.charAt(i);  
            System.out.println("char at " + i + " index is: " + c);  
        }  
    }  
}
```

Output:

```
PS E:\sem 5\java> cd "e:\sem 5\java\" ; if ($?) { javac stringToChar.java } ; if ($?) { java stringToChar }
char at 0 index is: h
char at 1 index is: e
char at 2 index is: l
char at 3 index is: l
char at 4 index is: o
char at 5 index is:
char at 6 index is: W
char at 7 index is: o
char at 8 index is: r
char at 9 index is: l
char at 10 index is: d
char at 11 index is: !
```

Source Code:

```
public class stringToChar1 {
    public static void main(String args[]) {
        String s1 = "hello";
        char[] ch = s1.toCharArray();
        for (int i = 0; i < ch.length; i++) {
            System.out.println("char at " + i + " index is: " + ch[i]);
        }
    }
}
```

Output:

```
PS E:\sem 5\java> cd "e:\sem 5\java\" ; if ($?) { javac stringToChar1.java } ; if ($?) { java stringToChar1 }
char at 0 index is: h
char at 1 index is: e
char at 2 index is: l
char at 3 index is: l
char at 4 index is: o
```


EXPERIMENT – 1.3

Aim:

WAP that convert Char to string.

Theory:

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character “\0”. A character array can be converted to a string and vice versa.

Methods:

1. Using copyOf() method of Arrays class
2. Using StringBuilder class
3. Using valueOf() method of String class
4. Using copyValueOf() method of String class
5. Using Collectors in Streams

Using copyOf() method of Array class

The given character can be passed into the String constructor. By default, the character array contents are copied using the Arrays.copyOf() method present in the Arrays class.

Using StringBuilder class

StringBuilder is a mutable class, therefore, the idea is to iterate through the character array and append each character at the end of the string. Finally, the string contains the string form of the characters.

Using valueOf() method of String class

This method inherently converts the character array to a format where the entire value of the characters present in the array is displayed. This method generally converts int, float, double, char, boolean, and even object to a string. Here we will achieve the goal by converting our character array to string.

Using copyValueOf() method of String class

The contents from the character array are copied subsequently modified without affecting the string to be returned, hence this method also enables us to convert the character array to string.

Using Collectors in Streams

With the introduction of streams in java8, we straight away use Collectors in streams to modify our character input array elements and later uses joining() method and return a single string and print it.

Source Code:

```
public class charToString {  
    public static void main(String args[]) {  
        char ch = 'a';  
        String str1 = Character.toString(ch);  
        System.out.println("String after conversion is: " + str1);  
    }  
}
```

Output:

```
PS E:\sem 5\java> cd "e:\sem 5\java\" ; if ($?) { javac charToString.java } ; if ($?) { java charToString }  
String after conversion is: a
```

```
String after conversion is: a
```

Source Code:

```
public class charToString1 {  
    public static void main(String args[]) {  
        char ch = 'a';  
        String str = String.valueOf(ch);  
        System.out.println("String after conversion is: " + str);  
    }  
}
```

Output:A screenshot of a Windows command prompt window. The prompt is 'PS E:\sem 5\java>'. The command 'java charToString1' has been entered and executed. The output 'String after conversion is: a' is displayed on the next line.

```
PS E:\sem 5\java> java charToString1  
String after conversion is: a
```

A screenshot of a terminal window showing the output of the program. The text 'String after conversion is: a' is displayed in a monospaced font on a dark background.

```
String after conversion is: a
```

EXPERIMENT – 1.4

Aim:

Program to find ASCII code of a character.

Theory:

ASCII acronym for American Standard Code for Information Interchange. It is a 7-bit character set contains 128 (0 to 127) characters. It represents the numerical value of a character. For example, the ASCII value of A is 65.

There are two ways to print ASCII value in Java:

- Assigning a Variable to the int Variable
- Using Type-Casting

Assigning a Variable to the int Variable

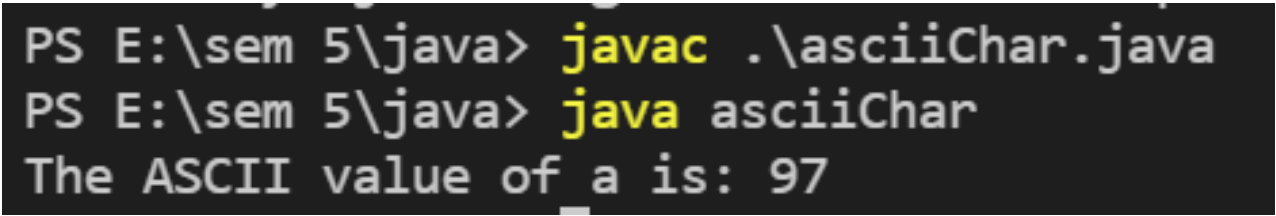
To print the ASCII value of a character, we need not use any method or class. Java internally converts the character value to an ASCII value.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Source Code:

```
public class asciiChar {  
    public static void main(String[] args) {  
        char ch1 = 'a';  
        int asciivalue = ch1; //type cast to int  
        System.out.println("The ASCII value of " + ch1 + " is: " + asciivalue);  
    }  
}
```

Output:

```
PS E:\sem 5\java> javac .\asciiChar.java  
PS E:\sem 5\java> java asciiChar  
The ASCII value of a is: 97
```



```
The ASCII value of a is: 97
```

Source Code:

```
public class asciiChar1 {  
    public static void main(String[] String) {  
        int ch1 = 'a';  
        System.out.println("The ASCII value of a is: " + ch1);  
    }  
}
```

Output:

```
PS E:\sem 5\java> javac .\asciiChar1.java
PS E:\sem 5\java> java asciiChar1
The ASCII value of a is: 97
```

```
The ASCII value of a is: 97
```

EXPERIMENT – 1.5

Aim:

Swapping two numbers using bitwise operators.

Theory:

In Java, there are many ways to swap two numbers. Generally, we use either `swap()` method of the `Math` class or use a third (temporary) variable to swap two numbers. Except these two ways, we can also swap two numbers using the bitwise operator (XOR) and using division and multiplication.

Using Bitwise Operator

Bitwise Operator: Bitwise XOR operator is used to swap two numbers. It is represented by the symbol (\wedge). It compares bits of two operands and returns false or 0 if they are equal and returns true or 1 if they are not equal. The truth table of XOR operator is as follows:

X	Y	$X \wedge Y$
0	0	0
0	1	1
1	0	1
1	1	0

We can use the bitwise XOR operator to swap two numbers without using the `swap()` method and third variable. We must follow the steps given below:

- Find the binary equivalent of given variables, say X and Y.
- Find $X \wedge Y$ and store it in x, i.e. $X = X \wedge Y$.
- Again, find $X \wedge Y$ and store it in Y, i.e. $Y = X \wedge Y$.
- Find $X \wedge Y$ and store it in X, i.e. $X = X \wedge Y$.
- The numbers are swapped.

Now implement the above steps in an example and understand the swapping.

Example:

Swap the variables $X = 5$ and $Y = 9$ using the bitwise operator.

Solution:

Step 1: Binary equivalent of the variables X and Y are:

$X = 5 = 0101$ and $Y = 9 = 1001$

Step 2: Find $X = X \wedge Y$.

X	0	1	0	1
Y	1	0	0	1
$X = X \wedge Y$	1	1	0	0

Step 2: Find $Y = X \wedge Y$.

X	1	1	0	0
Y	1	0	0	1
$Y = X \wedge Y$	0	1	0	1

Step 3: Find $X = X \wedge Y$.

X	1	1	0	0
Y	0	1	0	1
$X = X \wedge Y$	1	0	0	1

We see that the variable X contains 1001 which is equivalent to 9 and Y contains 0101 which is equivalent to 5. Therefore, the variables X and Y are swapped.

$X = 9$ and $Y = 5$

Source Code:

```
public class swapViaBitwise {  
    static void swapNumbers(int x, int y) {  
        System.out.println("Before swapping");  
        System.out.println("x= " + x + ", y= " + y);  
        x = x ^ y;  
        y = x ^ y;  
        x = x ^ y;  
        System.out.println("After swapping");  
        System.out.println("x= " + x + ", y= " + y);  
    }  
  
    public static void main(String[] args) {  
        int x = 12;  
        int y = 34;  
        swapNumbers(x, y);  
    }  
}
```

Output:

```
PS E:\sem 5\java> javac .\swapViaBitwise.java  
PS E:\sem 5\java> java swapViaBitwise  
Before swapping  
x= 12, y= 34  
After swapping  
x= 34, y= 12
```

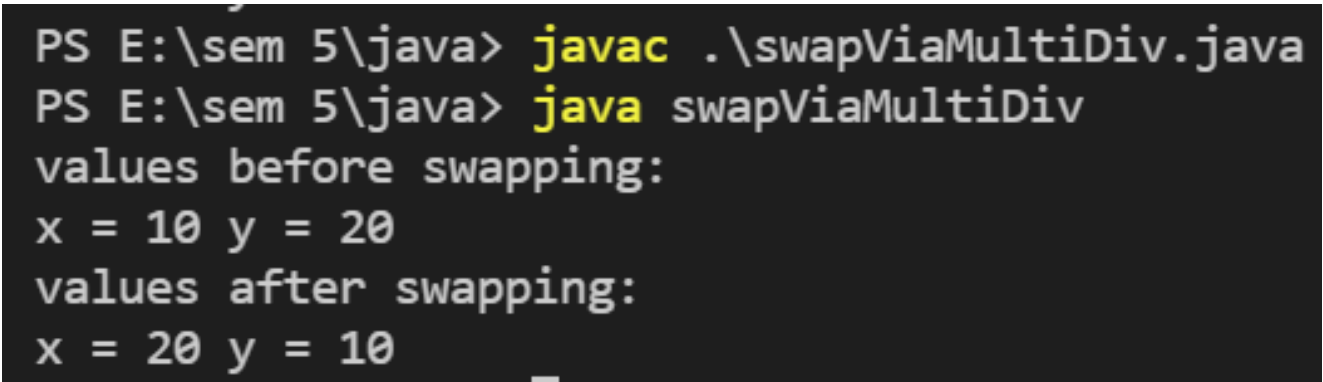
```
Before swapping  
x= 12, y= 34  
After swapping  
x= 34, y= 12
```

Swap – Multiplication Division

Source Code:

```
public class swapViaMultiDiv {  
    public static void main(String args[]) {  
        int x = 10;  
        int y = 20;  
        System.out.println("values before swapping:");  
        System.out.println("x = " + x + " y = " + y);  
        x = x * y;  
        y = x / y;  
        x = x / y;  
        System.out.println("values after swapping:");  
        System.out.println("x = " + x + " y = " + y);  
    }  
}
```

Output:



```
PS E:\sem 5\java> javac .\swapViaMultiDiv.java  
PS E:\sem 5\java> java swapViaMultiDiv  
values before swapping:  
x = 10 y = 20  
values after swapping:  
x = 20 y = 10
```

Viva Questions

1. What is the most important feature of Java?

Ans.

Java is a platform independent language.

2. What do you mean by platform independence?

Ans.

Platform independence means that we can write and compile the java code in one platform (e.g. Windows) and can execute the class in any other supported platform e.g. (Linux, Solaris, etc.).

3. What is JVM?

Ans.

JVM is Java Virtual Machine which is a run time environment for the compiled java class files.

4. Are JVM's platform independent?

Ans.

JVM's are not platform independent. JVM's are platform specific run time implementation provided by the vendor.

5. What is the difference between a JDK and a JVM?

Ans.

JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.