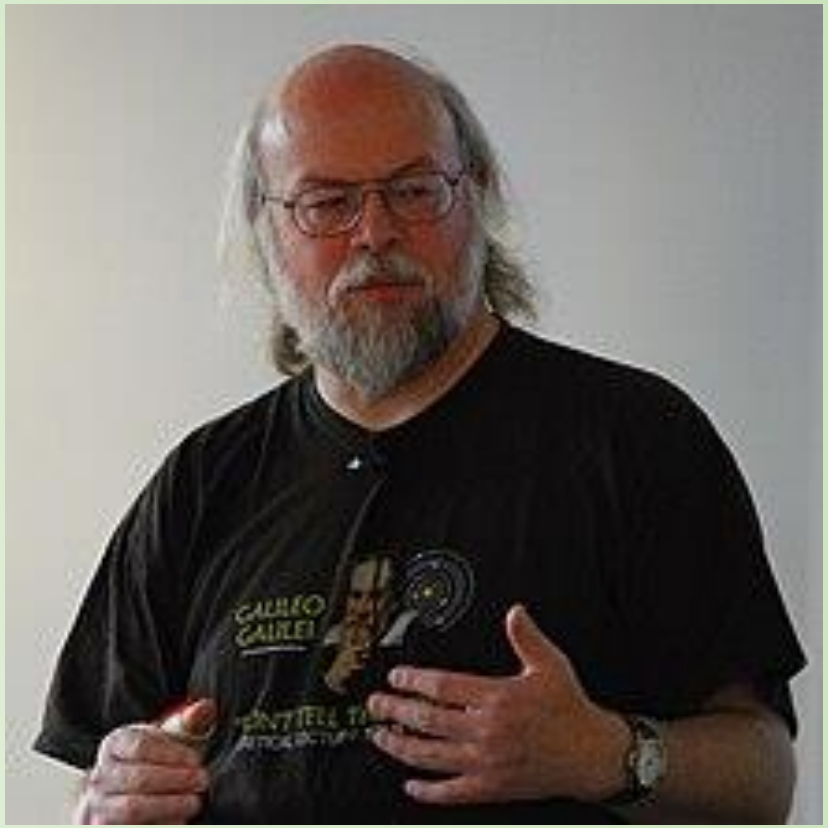
Java Fundamentals

Dr. Susovan Jana
Associate Professor & Programme In-Charge
Department of Computer Science & Engineering (IoT)
Institute of Engineering & Management, Kolkata, INDIA
Email (O): susovan.jana@iem.edu.in
Email (P): jana.susovan2@gmail.com

Father of Java

- James Gosling
  - DOB: 19 May 1955
  - Canadian Computer Scientist
- The Journey started in 1991 by a team of Sun Microsystems
  - Greentalk
  - Oak
  - Java, 1995



Brief Version History

- JDK 1.0 was released on January 23, 1996
- Oracle owned Java from Sun Microsystems on January 27, 2010
- Java SE 21 was released in September 19, 2023

Editions

- Java SE (Java Standard Edition)
- Java EE (Java Enterprise Edition)
- Java ME (Java Micro Edition)

Attributes of Java

- Familiar, Simple, Small
- Compiled and Interpreted
- Platform-Independent and Portable
- Object-Oriented
- Robust and Secure
- Distributed
- Multithreaded and Interactive
- High Performance
- Dynamic and Extensible

Java Download Link

- https://www.oracle.com/java/technologies/downloads/

Installation & Path Setting in Windows

- Windows
  - This PC > Properties > Advance System Setting
  - Environment Variables > System Variables > Select Path & Edit > Add installation path upto bin

Installation & Path Setting in Linux

- Linux
  - https://www.youtube.com/watch?v=vVrIDJ--GOA

First Java Program

```
class First
{
public static void main(String args[])
{
System.out.println("Welcome to JAVA");
}
}
```

Compile: javac First.java

After successful compilation a First.class file will be generated

Execution: java First

Java Program Execution Stages

Source Code
.java file

Compiler

Byte Code
.class file

OS wise JVM

Interpreter (Windows)

Interpreter (Linux)

Interpreter (Solaris)

Machine Code - Operating System

Java Comments

- Single-line Comments
  - Single-line comments start with two forward slashes (//).
  - Any text between // and the end of the line is ignored by Java (will not be executed).
  - // This is a comment
    System.out.println("Hello World");
- Java Multi-line Comments
  - Multi-line comments start with /* and ends with */.
  - Any text between /* and */ will be ignored by Java.
  - /* The code below will print the words Hello World
    to the screen, and it is amazing */
    System.out.println("Hello World");

Keywords

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

* not used

** added in 1.2

*** added in 1.4

**** added in 5.0

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

Primitive Data Types

- **byte**
  - The byte data type is an 8-bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive).
  - The byte data type can be useful for saving memory in large arrays, where the memory savings actually matters.
- **short**
  - The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive).
  - As with byte, the same guidelines apply: you can use a short to save memory in large arrays, in situations where the memory savings actually matters.
- **int**
  - By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -2^31 and a maximum value of 2^31-1.
- **long**
  - The long data type is a 64-bit two's complement integer. The signed long has a minimum value of -263 and a maximum value of 263-1.

Primitive Data Types

- **float**
  - The float data type is a single-precision 32-bit IEEE 754 floating point.
- **double**
  - The double data type is a double-precision 64-bit IEEE 754 floating point.
- **boolean**
  - The boolean data type has only two possible values: true and false. Use this data type for simple flags that track true/false conditions.
- **char**
  - The char data type is a single 16-bit Unicode character.

Data Types and Default Values

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Type Casting in Java

- Widening Casting (automatically)
  - converting a smaller type to a larger type size
  - byte -> short -> char -> int -> long -> float -> double
- Narrowing Casting (manually)
  - converting a larger type to a smaller size type
  - double -> float -> long -> int -> char -> short -> byte

Widening Casting

```
public class Main {
  public static void main(String[] args) {
    int myInt = 9;
    double myDouble = myInt; // Automatic casting: int to double
    System.out.println(myInt);      // Outputs 9
    System.out.println(myDouble);   // Outputs 9.0
  }
}
```

Narrowing Casting

```
public class Main {
  public static void main(String[] args) {
    double myDouble = 9.78d;
    int myInt = (int) myDouble; // Manual casting: double to int
    System.out.println(myDouble);   // Outputs 9.78
    System.out.println(myInt);      // Outputs 9
  }
}
```

Operators in Java

- Simple Assignment Operator
  - = Simple assignment operator
- Arithmetic Operators
  - + Additive operator (also used for String concatenation)
  - - Subtraction operator
  - * Multiplication operator
  - / Division operator
  - % Remainder operator

Operators in Java

- Unary Operators

  \+ Unary plus operator; indicates positive value (numbers are positive without this, however)

  \- Unary minus operator; negates an expression

  ++ Increment operator; increments a value by 1

  -- Decrement operator; decrements a value by 1

  ! Logical complement operator; inverts the value of a boolean

Operators in Java

- Equality and Relational Operators
  - == Equal to
  - != Not equal to
  - > Greater than
  - >= Greater than or equal to
  - < Less than
  - <= Less than or equal to

Operators in Java

- Conditional Operators

  && Conditional-AND

  || Conditional-OR

  ?: Ternary (shorthand for if-then-else statement)

- Type Comparison Operator

  instanceof Compares an object to a specified type

Operators in Java

- Bitwise and Bit Shift Operators
  - ~ Unary bitwise complement
  - << Signed left shift
  - >> Signed right shift
  - >>> Unsigned right shift
  - & Bitwise AND
  - ^ Bitwise exclusive OR
  - | Bitwise inclusive OR

Decision Making: *if-else*

```
class IfElseDemo {
    public static void main(String[] args) {
            int age=25;
                    if (age>=21)
                            {
                            System.out.println("He is Adult.");
                            }
                    else
                            {
                            System.out.println("He is not Adult.");
                            }
                    }
            }
```

Decision Making: **nested *if-else***

```
public class Demonested {
   public static void main(String[] args) {

      int n1 = 150, n2 = 180, n3 = 170;

      if (n1 >= n2) {
         if (n1 >= n3)
            System.out.println("Student with height: " + n1 + " is the tallest.");
         else
            System.out.println("Student with height: " + n3 + " is the tallest.");
      }
      else {
            if (n2 >= n3)
               System.out.println("Student with height: " + n2 + " is the tallest.");
            else
               System.out.println("Student with height: " + n3 + " is the tallest.");
      }
      System.out.println("\n");
   }
}
```

Decision Making: ***else-if lader***

```
class IfElseDemo {
    public static void main(String[] args)
{
        int testscore = 76;
        char grade;
        if (testscore >= 90) {
            grade = 'A';
        } else if (testscore >= 80) {
            grade = 'B';
}
else if (testscore >= 70) {
            grade = 'C';
        } else if (testscore >= 60) {
            grade = 'D';
        } else {
            grade = 'F';
        }
        System.out.println("Grade = " +
grade);
    }
}
```

Decision Making: ***switch* statement**

```
public class SwitchDemo {
   public static void main(String[] args) {
      int month = 8;
      String monthString;
      switch (month) {
         case 1:  monthString = "January";
                  break;
         case 2:  monthString = "February";
                  break;
         case 3:  monthString = "March";
                  break;
         case 4:  monthString = "April";
                  break;
         case 5:  monthString = "May";
                  break;
         case 6:  monthString = "June";
                  break;
         case 7:  monthString = "July";
                  break;
         case 8:  monthString = "August";
                  break;
         case 9:  monthString = "September";
                  break;
         case 10: monthString = "October";
                  break;
         case 11: monthString = "November";
                  break;
         case 12: monthString = "December";
                  break;
         default: monthString = "Invalid month";
                  break;
      }
      System.out.println(monthString);
   }
}
```

Iteration- *for* loop

Syntax:

```
for (initialization;  termination condition;  increment/decrement)
{
    statement(s)
}
```

Example:

```
class ForDemo {
    public static void main(String[] args){
        for(int i=1; i<11; i++){
            System.out.println("Count is: " + i);
        }
    }
}
```

Iteration- *while* loop

Syntax:

```
while (expression) {
    statement(s)
}
```

Example:

```
class WhileDemo {
   public static void main(String[] args){
      int count = 1;
      while (count < 11) {
         System.out.println("Count is: " + count);
         count++;
      }}}
```

Iteration- *do-while* loop

Syntax:

```
do {
    statement(s)
} while (expression);
```

Example:

```
class DoWhileDemo {
   public static void main(String[] args){
      int count = 1;
      do {
         System.out.println("Count is: " + count);
         count++;
      } while (count < 11);
   }}
```

Jumping Statement: ***continue***

```
public class Test {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      if (i == 2) {
        continue; //skip the particular iteration
      }
      System.out.print(i+"\t");
    }
  }
}
```

Output: 0 1 3 4

Jumping Statement: ***break***

```
public class Test {
  public static void main(String[] args) {
    for (int i = 0; i < 5; i++) {
      if (i == 2) {
        break; //it terminates the iteration
      }
      System.out.print(i+"\t");
    }
  }
}
```

Output: 0 1

Array in Java

- Java array is an object which contains elements of a similar data type.
- Additionally, The elements of an array are stored in a contiguous memory location.
- Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.
- Advantage
  - Random access
- Disadvantage
  - Storage limitation

1D Array in Java

- Declaration
  - dataType[] arr;

    (or)
  - dataType []arr;

    (or)
  - dataType arr[];
- Instantiation
  - arrayRefVar=new datatype[size];
- Declaration, instantiation, and initialization
  - int a[]={33,3,4,5};

1D Array in Java

```
class Testarray{
public static void main(String args[]){
int a[]=new int[5];//declaration and instantiation
a[0]=10;//initialization
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;
//traversing array
for(int i=0; i<a.length; i++)//length is the property of array
System.out.println(a[i]);
}}
```

1D Array with for Loop

```
class Testarrayloop{
public static void main(String args[]){
int arr[]={3,7,4,5};
//printing array using for-each loop
for(int i:arr)
System.out.println(i);
}}
```

Output: 3
7
4
5

Multidimensional Array in Java

- Declaration
  - dataType[][] arrayRefVar;
    (or)
  - dataType [][]arrayRefVar;
    (or)
  - dataType arrayRefVar[][];
    (or)
  - dataType []arrayRefVar[];
- Instantiation
  - arrayRefVar=new datatype[size][size];
- Declaration, instantiation, and initialization
  - int arr[][]={{1,2,3},{2,4,5},{4,4,5}};

Multidimensional Array in Java

```
class Testarraymultidementional{
public static void main(String args[]){
//declaring and initializing 2D array
int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
//printing 2D array
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
   System.out.print(arr[i][j]+" ");
 }
 System.out.println();
}
}}
```

Copying an Array in Java

```
class TestArrayCopyDemo {
    public static void main(String[] args) {
        //declaring a source array
        char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e', 'i', 'n', 'a', 't', 'e', 'd' };
        //declaring a destination array
        char[] copyTo = new char[7];
        //copying array using System.arraycopy() method
        System.arraycopy(copyFrom, 2, copyTo, 0, 7);
        //printing the destination array
        System.out.println(String.valueOf(copyTo));
    }
}
```

Output: caffein

Cloning an Array in Java

```
class Testarray{
public static void main(String args[]){
        int arr[]={33,3,4,5};
        System.out.println("Printing original array:");
        for(int i:arr)
            System.out.println(i);
        System.out.println("Printing clone of the array:");
        int carr[]=arr.clone();
        for(int i:carr)
            System.out.println(i);
        System.out.println("Are both equal?");
        System.out.println(arr==carr);
}}
```

Command Line Argument

```
class CommandLineExample{
public static void main(String args[]){
System.out.println("Your first argument is: "+args[0]);
}
}
```

Compile: javac CommandLineExample.java

Run: java CommandLineExample **IEM**

Output: Your first argument is IEM

String

- Java String is basically an object that represents sequence of char values. An array of characters works same as Java string.
  - char[] ch={'s','u','s','o','v','a','n'};
    String s=new String(ch);

    or
  - String s="susovan";

Methods of String Class

```
class Demostring {
public static void main(String args[]) {
String s="Welcome to Java";
System.out.println(s); //output: Welcome to Java
int l=s.length();
System.out.println(l);//output:15
char c=s.charAt(5);
System.out.println(c);//output:m Note: It returns the charcter at the index 5
System.out.println(s.indexOf("m"));//output:5
System.out.println(s.indexOf("o"));//output:4
System.out.println(s.indexOf("o",5));//output:9
} }
```

Methods of String Class

```
class Demostring {
public static void main(String args[]) {
String s="Welcome to Java"; System.out.println(s); //output: Welcome to Java
String s1=s.substring(8);
System.out.println(s1); //output: to Java
String s2=s.substring(8,13);
System.out.println(s2); //output: to Ja
String s3=s1.concat(s2);
System.out.println(s3); //output: to Javato Ja
String s4="   Hello     ";
System.out.println(s4); //output:   Hello
System.out.println(s4.trim()); //output: Hello
System.out.println(s.toLowerCase());//output: welcome to java
System.out.println(s.toUpperCase());//output:WELCOME TO JAVA
}}
```

Methods of String Class

```
class Demostring {
public static void main(String args[]) {
String s="Welcome to Java";
System.out.println(s); //output: Welcome to Java
String t="welcome to java";
System.out.println(s.equals(t));//false
System.out.println(s.equalsIgnoreCase(t));//true
System.out.println(s.replace('o', 'X'));//WelcXme tX Java
String arr[]=s.split(" ", 3);
for (String a : arr)
System.out.print(a); // Welcome to Java
} }
```