[[1]](#footnote-1)

1. LANGUAGE BASICS

Shailesh Kumar Thakur

TOPICS

1. Introduction
2. Identifiers
3. Reserved Words
4. Data Types
5. Literals
6. Arrays
7. Types of Variables
8. Un Initialiezed Array
9. Var-Args Method
10. Main method
11. Command Line Arguments
12. Java Coding Standards
13. Memory Architecture

# INTRODUCTION

## Sun Microsystems Inc in 1991.

## Later acquired by Oracle Corporation

## High-level object-oriented programming language

## **Features of Java-**

## **Object-Oriented-** programming structure which incorporates the concept of real-world objects contain both data and function.

## **Platform Independent-** Write your code once and run anywhere.

## **Simple-** Easy to learn

## **Robust-** Uses features like Memory management, lack of pointers that avoids security problems, automatic garbage collection, authentication techniques like public-key encryption makes JAVA robust.

## **Architecture-Neutral-** JAVA compilers make an object file which makes them executable anywhere on any system.

## **Portable-** Implementation independent and architecture-neutral makes it portable.

# IDENTIFIERS

## Name in the JAVA program.

## Class name, variable name, method name, label name.

## **Rules for identifiers:-**

* **Case Sensitive**
* **Allowed Characters**- [A-Z], [a-z], [0-9], underscore( \_ ) and dollar( $ ).
* **Starting with digit not allowed.**
* **Can’t use reserved words** as an identifier.
* **Pre-defined class names and interface** names used as an identifier but **not recommended** to use for good programmers.

# RESERVED WORDS

# DATA TYPES

* **Integral:**
* **byte:**

size: 1 byte

range: -128 to 127

* **short:**

size: 2 byte

range: -215 to 215-1

* **int:**

size: 4 byte

range: -231 to 231-1

* **long:**

size: 8 byte

range: -263 to 263-1

* **Character:**

Size: 2 byte

Range: 0 to 65535

* **Boolean:**

Size: not applicable

Range: not applicable

* **Floating Point:**
* **float:**

size: 4 byte

range: -3.4e38 to 3.4e38

precision: single

decimal: 5 to 6 decimal places

* **double:**

size: 8 byte

range: -1.7e308 to 1.7e308

precision:double

decimal: 14 to 15 decimal places

# LITERALS

Eg int x = 5;

x is an identifier

int is data type, and

5 is literals/data value.

|  |  |  |
| --- | --- | --- |
| ***Literals*** | ***Allowed Digit*** | ***Valid***  ***Examples*** |
| Integral Literals   1. Decimal 2. Octal 3. Hexa-Decimal |  |  |
| 0-9 | int x=10; |
| 0-7 | int x=**0**10 |
| 0-9, A-F | int x=**0x**Beef**;**  int x=**0X121F;** |
| Floating Point Literals   1. Float 2. Double |  |  |
| Explicitly define as float type by **F or f.** | float f=123.45**f** |
| Floating-point literals are by default double type. | double d=123.45 |
| |  |  |  | | --- | --- | --- | | Boolean | **true or false** | boolean b= true;  boolean b = false; | | Char Literals | Single char within single quotes.  OR  Integral literals either in decimal or octal or hexadecimal form but allowed value range is 0 to 65535 | char c = ‘c’;  char c = 97;  char c = 0x97;  char c =’\u0061’.  (Unicode representation) | | | |

# 

Figure 1Data type conversion

# ARRAYS

* **Def.:** Indexed collection of fixed numbers of the homogeneous data elements.
* **Adv:** We can represent multiple values with the same name to improve the readability of the code.
* **Cons:** Fixed in size. **We can resolve these problems with the use of collections**.
* **Declaration:**
* **1D Array:**

int[] a; or int []a; or int a[];

**Note:** At the time of the declaration, we can’t specify the size of array.

* **2D Array:**

int[][] a;

int[] []a;

int[] a[];

int [][]a;

int []a[];

int a[][];

* **3D Array:**

int[][][] a;

int[][] []a;

int[][] a[];

int[] [][]a;

int[] []a[];

int[] a[][];

int [][][]a;

int [][]a[];

int []a[][];

int a[][][];

**Note:** While the declaration of more than one variable in a line, we can’t specify dimensions for second variables onwards.

int []a, b;(valid)

int []a, []b;(invalid)

* **Construction:**
* Every array in java is an object. Hence we use the **new operator** to create an object of an array.
* **1D Array:**

int[] a= new int[3];

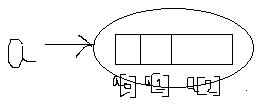
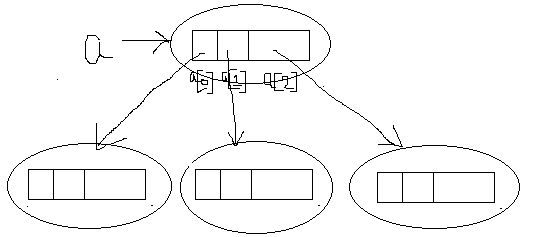


Figure 2 Memory Representation

* **2D Array:**

int[][] a= new int[3][3];



int [][]a= new int[3][];

a[0]=new int[3];

a[1]=new int[4];

…

int [][]a=new int[][3];**(invalid)**

* **Rules:**

1. Array size is required compulsory at the time of array construction. int a[] = new int[3];
2. Array size can be zero. int a[] = new int[0];
3. Can’t specify a negative value for array size.
4. Allow data types to specify array size are byte, short, char, and int, otherwise, we will get a compile-time error.
5. **The maximum allowed array size in java is max\_value of int size 231-1.**

* **Initialization:**

int[] a= new int[3];

**by default array is initialized with 0 by jvm.**

a[0]=5;

* We can perform declaration, construction and initialization in a single line.

int []a={10,20,30};

int [][]a={{10,20},{20, 30, 40}};

* **Anonymous arrays:**
* Arrays without name.
* Just for instant use.
* Eg. new int[]{10,20,30,40}

# JJ

# IMPORTANT NOTES:

**length vs length():**

**length:**

1. Final variable of Array class(applicable only for array).
2. Represent the size of the arrays.
3. Eg. int a[]=new int[3];

a.length;🡪 **get the size of the array i.e 3.**

int [][]a=new int[3][4];

a.length🡪 3

a[0].length🡪 4.

**length():**

1. Final method of String class(applicable only for String objects).
2. Returns no. of characters present in the String.
3. Eg. String s=”Shailesh”;

s.length;🡪 **8 number of characters**

1. Shailesh Kumar Thakur, Dhirendra Pratap Singh, Jaytrilok Choudhary are with Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal, MP-462003, India (e-mail: skt.nit.18@gmail.com; dpsingh.manit@gmail.com; jaytrilok@gmail.com). [↑](#footnote-ref-1)