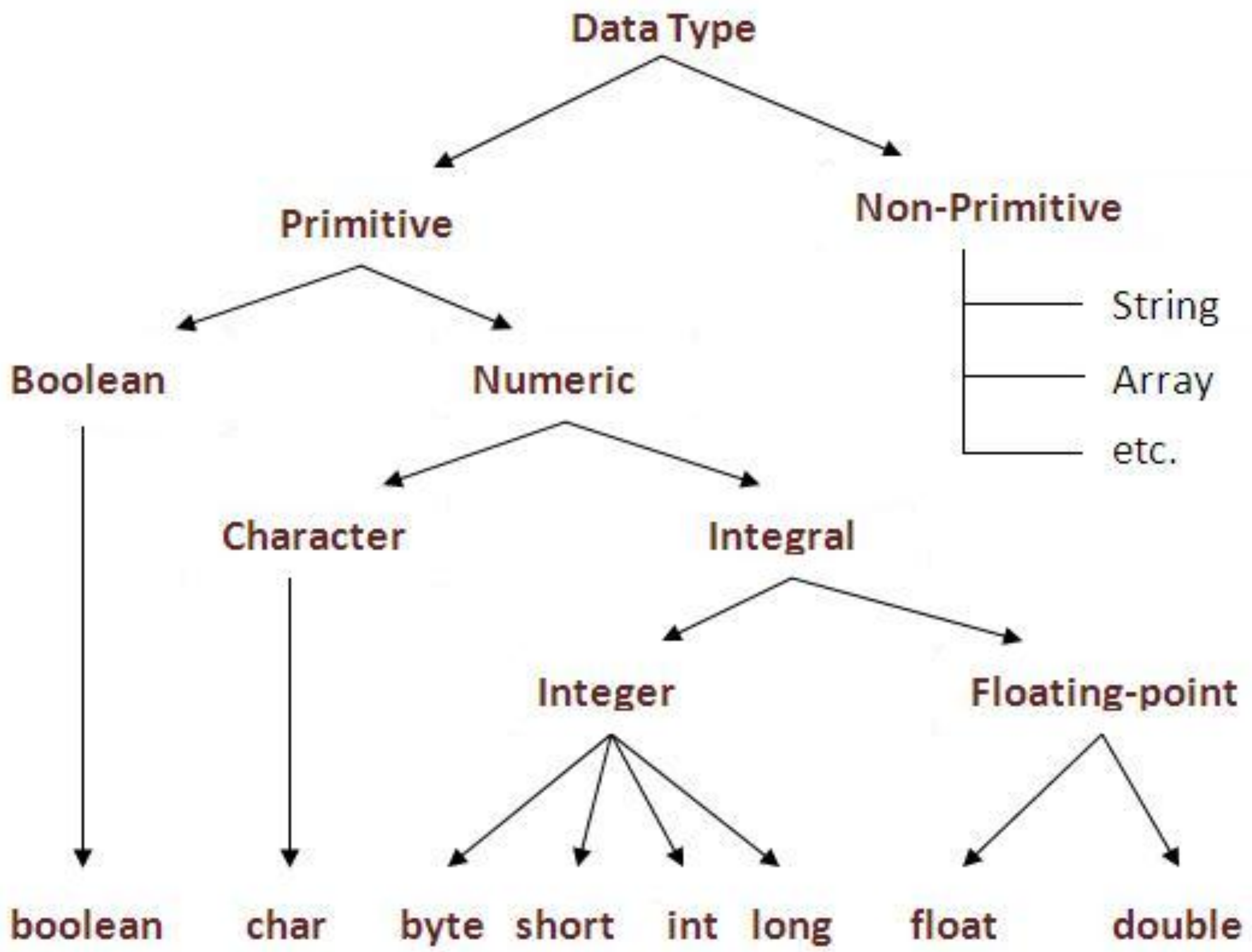
UNIT-II
Building Blocks of Language

Primitives Data Types

- Data types represent the different values to be stored in the variable. In java, there are two types of data types:
  - **Primitive data types**
  - **Non-primitive data types**


Data Type
Primitive
Non-Primitive
String
Array
etc.
Boolean
Numeric
Character
Integral
Integer
Floating-point
boolean
char
byte
short
int
long
float
double

Type	Description	Default	Size	Example Literals
boolean	true or false	false	1 bit	true, false
byte	twos complement integer	0	8 bits	(none)
char	Unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\'', '\n', 'ß'
short	twos complement integer	0	16 bits	(none)
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d

User Defined Data Type

- **User defined data types are those that user programmer defines.** For example, classes, objects, strings, interfaces.
- This is also called object or reference type.
- Employee **e1** = new Employee();

Derived Data Type

- **Derived data types** are those which used to store multiple values of same type.

- They are made by using any other data type for example, arrays.

```
int a[]=new int[5];
a[0]=10;
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;
```

Identifiers and Literals

- Literals means constant values like 1, 1234, -45, 3.14, “Hello”.
- Ex: Boolean Literals, integer , character literals etc.

Ex: Char c=“a”;

Int x=10;

Boolean value=true;

Identifiers and Literals

- In programming languages, identifiers are used for identification purpose. In Java an **identifier** can be a class name, method name, variable name or a label.

- For example :

  public static void main(String[] args)

- Here, string, args, main are identifiers.

Rules for Identifiers

- **There are certain rules for defining a valid java identifiers. These rules must be followed, otherwise we get compile-time error.** These rules are also valid for other languages like C,C++.

1) The only allowed characters for identifiers are all alphanumeric characters([**A-Z**],[**a-z**],[**0-9**]), ‘**$**‘(dollar sign) and ‘**_**‘ (underscore).
   - For example “week@” is not a valid java identifier as it contain ‘@’ – special character.
2) Identifiers should **not** start with digits(**[0-9]**).
   - For example “123area” is a not a valid java identifier.

Rules for Identifiers

3) Java identifiers are **case-sensitive**.

4) There is no limit on the length of the identifier but it is advisable to use an optimum length of 4 – 15 letters only.

5) **Reserved Words** can’t be used as an identifier.

   – For example “int while = 20;” is an invalid statement as while is a reserved word. There are **53** reserved words in Java.

Declaration of Constants

- Java does not directly support constants. However, a **static final variable is effectively work as a constant.**
- The **static** modifier causes the **variable to be available without loading an instance of the class** where it is defined.
- The **final** modifier causes the variable to be **unchangeable**.

Declaration of Constants

- Java constants are normally declared in ALL CAPS. Words in Java constants are normally separated by underscores.
- **Example:**

```
public class MaxUnits
{
    public static final int MAX_UNITS = 25;
}
```

Declaration of Variables

- **A variable is a container that holds values that are used in a program.** To be able to use a variable it needs to be declared.

- Java is a strongly typed programming language. This means that every variable must have a data type associated with it.

- For example, a variable could be declared to use one of the eight primitive data types: byte, short, int, long, float, double, char or boolean.

Declaration of Variables

- To declare a variable in Java, all that is needed is the data type followed by the variable name:
- **Example:**

```
int numberOfDays;
```

- **More examples:**

```
byte nextInStream;
short hour;
long totalNumberOfStars;
float reactionTime;
double itemPrice;
```

Initialization of Variables

- **Before a variable can be used it must be given an initial value. This is called initializing the variable.**
- To initialize a variable we use an assignment operator.
- **Example:**

```
int numberOfDays;

numberOfDays = 7;
```

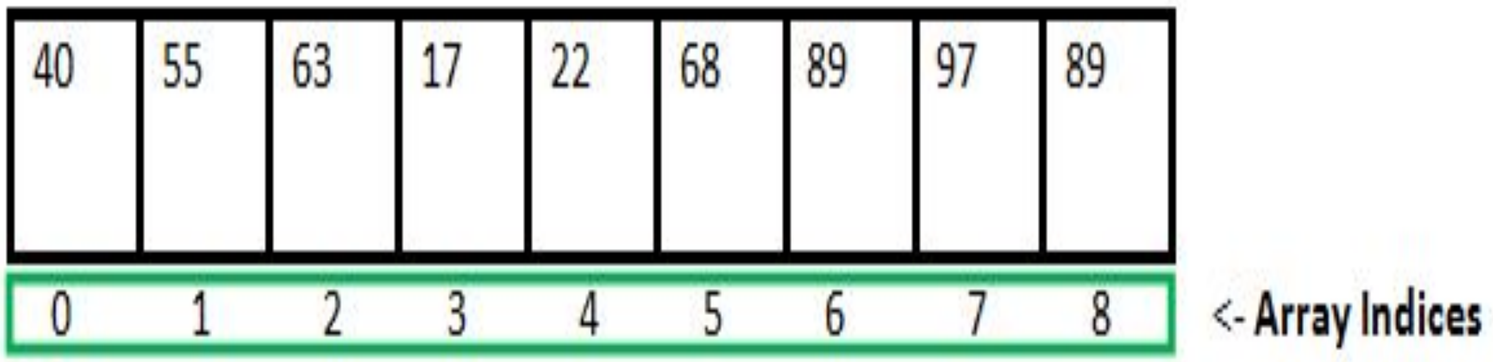
Arrays in JAVA

- Normally, **array is a collection of similar type of elements that have contiguous memory location.**

- **Java array** is an object the contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array.

- Array in java is index based, first element of the array is stored at 0 index.

Arrays in JAVA



Array Length = 9

First Index = 0

Last Index = 8

Arrays in JAVA

- **Advantage :**
  - **Code Optimization:** It makes the code optimized, we can retrieve or sort the data easily.
  - **Random access:** We can get any data located at any index position.
- **Disadvantage :**
  - **Size Limit:** We can store only fixed size of elements in the array. It doesn't grow its size at runtime.

Arrays in JAVA

- **Types of Array:**

- There are **two** types of array.

Single Dimensional Array

Multidimensional Array

Single Dimensional Arrays

- **Declaration Syntax:**

```
type var-name[];
    OR
type[] var-name;
```

- **Example:**

```
int intArray[];
    or
int[] intArray;
```

Single Dimensional Arrays

- **Instantiation** of an Array in java:

  arrayRefVar=**new** datatype[size];

- **Example:**

  int a[]=**new int**[5];

- **Declaration, Instantiation and Initialization**

  int a[]={1,2,3,4};

Multi Dimensional Arrays

- **Two Dimensional Array:**

```
int[][] intArray = new int[10][20];
```

- **Three Dimensional Array:**

```
int[][][] intArray = new int[10][20][10];
```

Multi Dimensional Arrays

```
class multiDimensional
{
    public static void main(String args[])
    {
        // declaring and initializing 2D array
        int arr[][] = { {2,7,9},{3,6,1},{7,4,2} };

        // 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();
        }
    }
}
```

Multi Dimensional Arrays

Output

```
2 7 9
3 6 1
7 4 2
```