VIPS
Technical Campus
योग: कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

SCHOOL OF ENGINEERING AND TECHNOLOGY

Course : OBJECT ORIENTED PROGRAMMING

Paper Code: AIML-202,IIOT-202

Faculty : Dr. Shivanka

Assistant Professor

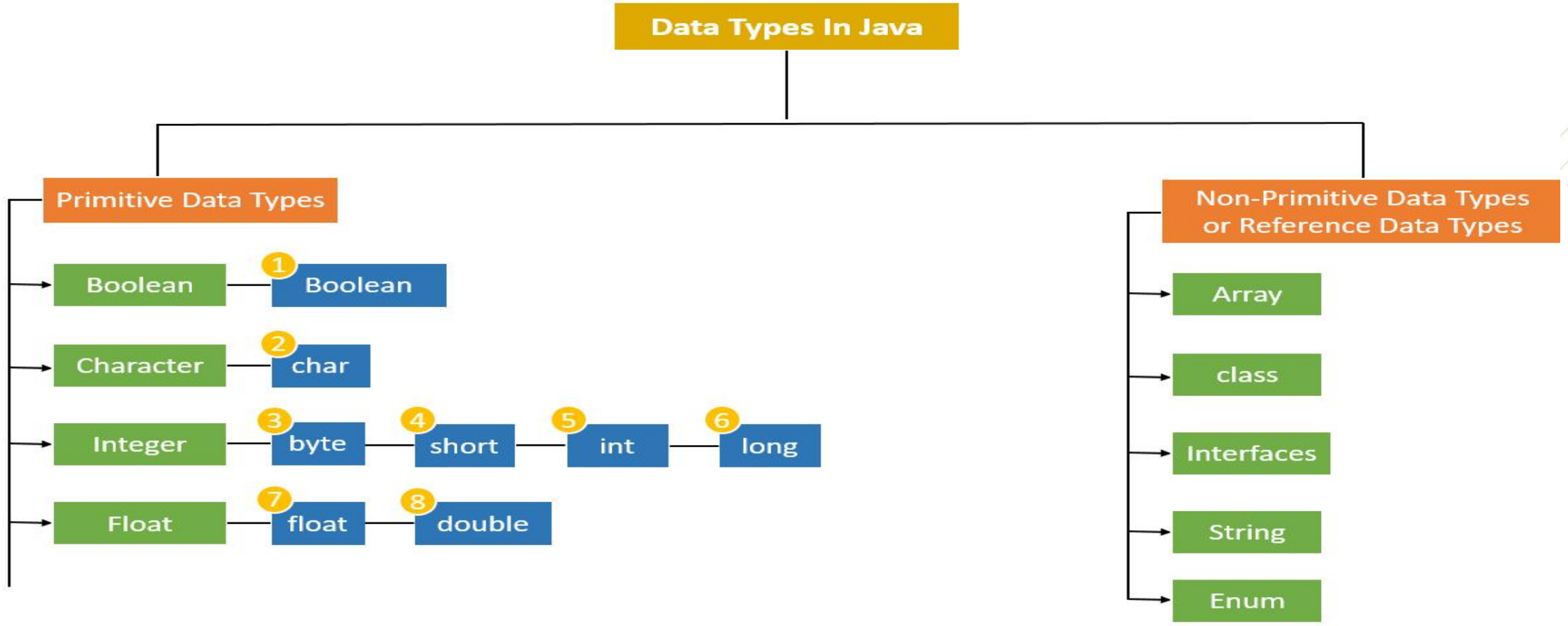
VIPS

Data Types in Java

- Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

- Primitive data types: The primitive data types include boolean, char, byte, short, int, long, float and double.
- Non-primitive data types: The non-primitive data types include Classes, Interfaces, and Arrays.

Data Types in Java



S.No	Data Type	Description/Nature	Storage Value	Size
1.	Byte	Arithmetic Type	Integer	1 Byte
2.	Short	Arithmetic Type	Integer	2 Byte
3.	Int	Arithmetic Type	Integer	4 Byte
4.	Long	Arithmetic Type	Integer	8 Byte
5.	Float	Arithmetic Type	Float/Decimal	4 Byte
6.	Double	Arithmetic Type	Float/Decimal	8 Byte
7.	Boolean	Boolean Type(0/1)	True or False	1 Bit
8.	Char	Character Type	Character	2 Byte

Boolean Data Type

- 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 Data Type

- The byte data type is an example of primitive data type. It isan 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.
- The byte data type is used to save memory in large arrays where the memory savings is most required. It saves space because a byte is 4 times smaller than an integer. It can also be used in place of "int" data type.
- Example:
- byte a = 10, byte b = -20

Short Data Type

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

- The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

short s = 10000, short r = -5000

Int Data Type

- 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 default value is 0.
- The int data type is generally used as a default data type for integral values unless if there is no problem about memory.

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

Float Data Type

- The float data type is a single-precision 32-bit.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

Long Data Type

- 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 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

Double Data type

The double data type is a double-precision 64-bit IEEE 754 floating point. 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 Data Type

- 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 letterA = 'A'

Q Why char uses 2 byte in java and what is \u0000 ?

Ans:-It is because java uses Unicode system not ASCII code system. The \u0000 is the lowest range of Unicode system.

Unicode System

- Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages.
- Why java uses Unicode System?
- Before Unicode, there were many language standards:
- ASCII (American Standard Code for Information Interchange) for the United States.
- ISO 8859-1 for Western European Language.
- KOI-8 for Russian.
- GB18030 and BIG-5 for chinese, and so on.

Problem

This caused two problems:

- A particular code value corresponds to different letters in the various language standards.
- The encodings for languages with large character sets have variable length.Some common characters are encoded as single bytes, other require two or more byte.

Solution

- To solve these problems, a new language standard was developed i.e. Unicode System.
- In unicode, character holds 2 byte, so java also uses 2 byte for characters.
- **lowest value:\u0000**
- **highest value:\uFFFF**

Example of Datatype


```
public class Datatype {
  public static void main(String[] args) {
    int myNum = 10;              // integer (whole number)
    float myFloatNum = 6.99f;    // floating point number
    char myLetter = 'G';         // character
    boolean myBool = true;       // boolean
    String myText = "OOPS CLASS";    // String
    System.out.println("Integer Value is: "+ myNum);
    System.out.println("Float value is: " + myFloatNum);
    System.out.println("Char value is: " +myLetter);
    System.out.println("Boolean value  is: " +myBool);
    System.out.println("String value is: "+ myText);
  }
}
```

Output:-Integer Value is: 10

Float value is: 6.99

Char value is: G

Boolean value is: true

String value is: OOPS CLASS

Example of float Data types:

A floating point number can also be a scientific number with an "e" to indicate the power of 10:

```
public class Main {
  public static void main(String[] args) {
    float f1 = 35e3f;
    double d1 = 12E4d;
    System.out.println(f1);
    System.out.println(d1);
  }
}
```

Output:
35000.0
120000.0

Non-Primitive Data Types

- Non-primitive data types are called reference types because they refer to objects.
- The **main difference between primitive and non-primitive data types are:**

- Primitive types are **predefined (already defined)** in Java. Non-primitive types are created by the programmer and is not defined by Java (except for String).
- Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
- A primitive type has always **a value,** while non-primitive types can be **null.**
- A primitive type starts with a **lowercase letter**, while non-primitive types starts with an **uppercase letter**.
- The size of a primitive type depends on the data type, while non-primitive types have all the **same size**.
- Examples of non-primitive types are **Strings, Arrays, Classes, Interface**, etc.

Java Strings

Strings are used for storing text.A String variable contains a collection of characters surrounded by double quotes:

String a1= “OOPS Java Class”;

Example :

```
public class Main {
  public static void main(String[] args) {
    String a1= “OOPS Java Class”;
    System.out.println(g);
  }
}
```

Output: OOPS Java Class

String Length

- A String in Java is actually an object, which contain methods that can perform certain operations on strings.
- For example, the length of a string can be found with the length() method:

Example of String Length:-

```
public class Main {
  public static void main(String[] args) {
    String txt = "IIOT and AIML OOPS class";
    System.out.println("The length of the txt string is: " + txt.length());
  }}
```

Output:The length of the txt string is: 24

String Concatenation

- The + operator can be used between strings to combine them. This is called concatenation:

- Example

```
public class stringConcat {
  public static void main(String args[]) {
    String firstName = "Shivam";
    String lastName = "roy";
    System.out.println(firstName + " " + lastName);
  }
} Output:Shivam roy
```

Adding number and String

- Java uses the + operator for both addition and concatenation.
- Numbers are added. Strings are concatenated.

If you add two numbers, the result will be a number:example

```
int a = 40;
int b = 30;
int c = a + b;  // c will be 70 (an integer/number)
```

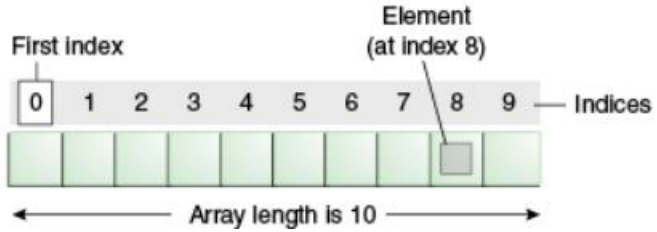
Example of String Concatenation in Number

```
public class Main {
 public static void main(String[] args)
{
    String a = "40";
    String b = "30";
    String c = a + b;
    System.out.println(c);
  }}
```

Output 4030

Arrays

- An array is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed. You have seen an example of arrays already, in the main method of the "Hello World!" application.



An array of 10 elements.

- Illustration of an array as 10 boxes numbered 0 through 9; an index of 0 indicates the first element in the array.

- Each item in an array is called an element, and each element is accessed by its numerical index. As shown in the preceding illustration, numbering begins with 0. The 9th element, for example, would therefore be accessed at index 8.

Array Example


```
class ArrayDemo {
   public static void main(String[] args) {  // declares an array of integers
      int[] anArray;
      anArray = new int[10];   // allocates memory for 10 integers
anArray[0] = 100;    // initialize first element

      anArray[1] = 200;  // initialize second elemen
      anArray[2] = 300;
      anArray[3] = 400; // and so forth
      anArray[4] = 500;
      anArray[5] = 600;
      anArray[6] = 700;
      anArray[7] = 800;
      anArray[8] = 900;
anArray[9] = 1000;
   System.out.println("Element at index 0: " + anArray[0]);
    System.out.println("Element at index 1: " + anArray[1]);
    System.out.println("Element at index 2: "  + anArray[2]);
    System.out.println("Element at index 3: "  + anArray[3]);
    System.out.println("Element at index 4: "    + anArray[4]);
    System.out.println("Element at index 5: "     + anArray[5]);
    System.out.println("Element at index 6: " + anArray[6]);
    System.out.println("Element at index 7: " + anArray[7]);
    System.out.println("Element at index 8: "+ anArray[8]);
    System.out.println("Element at index 9: "+ anArray[9]);   } }
```

Output

```
Element at index 0: 100
Element at index 1: 200
Element at index 2: 300
Element at index 3: 400
Element at index 4: 500
Element at index 5: 600
Element at index 6: 700
Element at index 7: 800
Element at index 8: 900
Element at index 9: 1000
```

Example :Sum of array N element


```
package array;
import java.util.Scanner;
public class Arraysum {
    public static void main(String[] args)
    { int n,sum=0;
    Scanner ob=new Scanner(System.in);
    System.out.println("Enter the size of array : ");
    n=ob.nextInt();
    int a[]=new int[n];
    System.out.println("Enter array Elements : ");
    for(int i=0;i<n;i++)
    {
    a[i]=ob.nextInt();
    sum=sum+a[i];
    }System.out.println("Sum of array N element is : "+sum);
    }}
```

Output: -

```
Enter the size of array :
10
Enter array Elements :
2
4
6
8
10
12
14
16
18
20
Sum of array N element is : 110
```

Declares an array of integers

// declares an array of integers

- int[] anArray;
- Like declarations for variables of other types, an array declaration has two components: the array's type and the array's name.

- Similarly, you can declare arrays of other types:
- byte[] anArrayOfBytes;
- short[] anArrayOfShorts;
- long[] anArrayOfLongs;
- float[] anArrayOfFloats;
- double[] anArrayOfDoubles;
- boolean[] anArrayOfBooleans;
- char[] anArrayOfChars;
- String[] anArrayOfStrings;
- You can also place the brackets after the array's name:

- // this form is discouraged
- float anArrayOfFloats[];

Creating, Initializing, and Accessing an Array

- One way to create an array is with the new operator. The next statement in the ArrayDemo program allocates an array with enough memory for 10 integer elements and assigns the array to the anArray variable.

- // create an array of integers
- anArray = new int[10];

Alternatively, you can use the shortcut syntax to create and initialize an array:

```
int[] anArray = {
    100, 200, 300,
    400, 500, 600,
    700, 800, 900, 1000
};
```

Thank You