Data Types, Variables and Arrays (Chapter 3 of Schilit)

Object Oriented Programming BS (CS/SE) II

By

Abdul Haseeb Shaikh

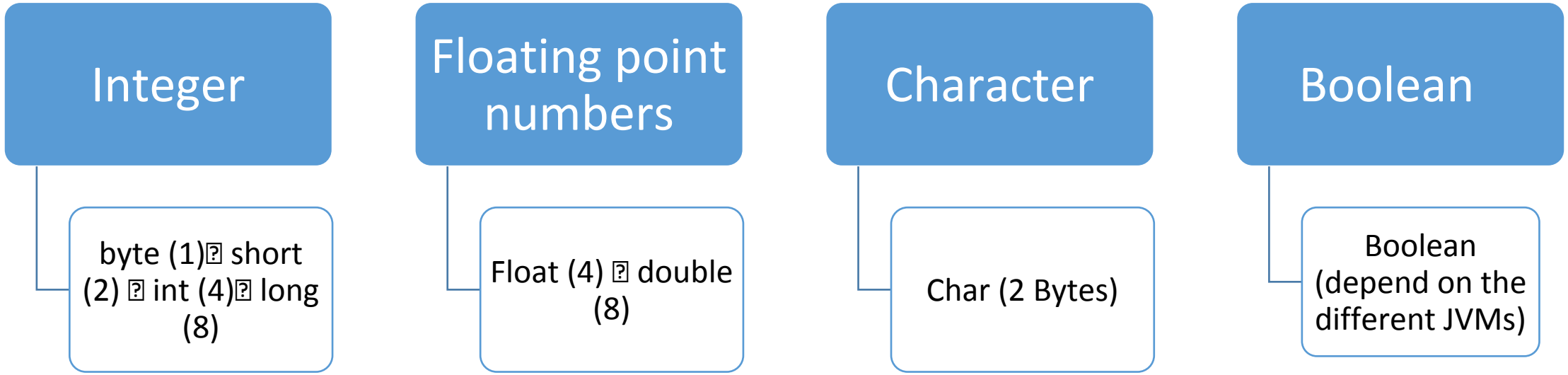
Java is Strongly Typed Language

- Compiler is the Boss
  - Every Declaration must have a data type

Primitive Data Types

- A set of basic data types from which all other data types are constructed.

Primitive Types



Integer

Name	Width	Range
long	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	–2,147,483,648 to 2,147,483,647
short	16	–32,768 to 32,767
byte	8	–128 to 127

Floating Point Type

Name	Width in Bits	Approximate Range
double	64	4.9e–324 to 1.8e+308
float	32	1.4e–045 to 3.4e+038

Taking Input

Package:

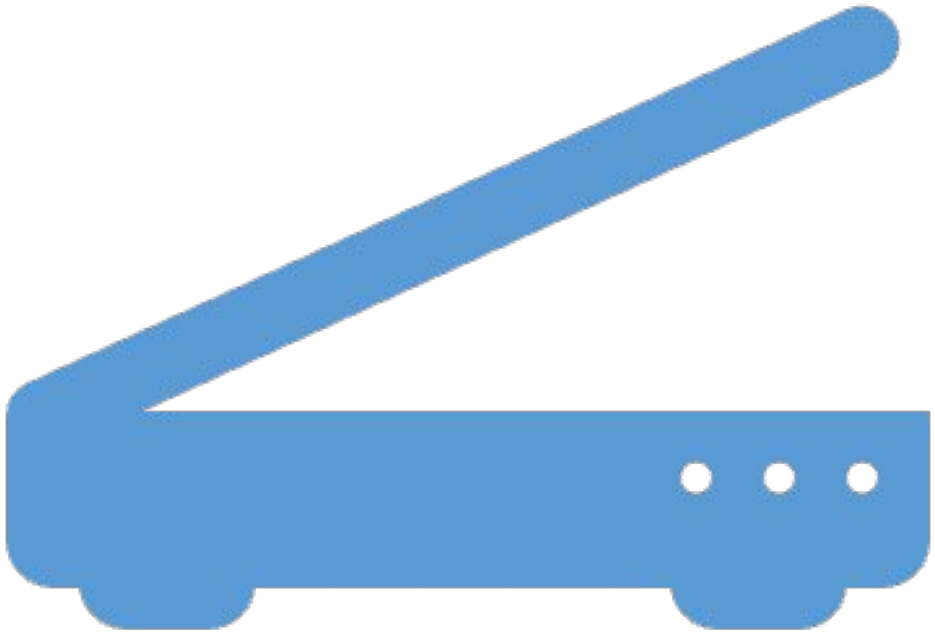
java.util

Class:

Scanner

Syntax:

import java.util.Scanner;



Program to take Input of different data types

```
import java.util.Scanner;
class FirstProgram{

public static void main(String args[]){

Scanner scan=new Scanner(System.in);


  //int types
       byte b1=scan.nextByte();
       int i1=scan.nextInt();
       short s1=scan.nextShort();
       long l1=scan.nextLong();

       //float and double
       float f1=scan.nextFloat();
       double d1=scan.nextDouble();

       //char
       char c1=scan.next().charAt(0);

       //String
       String str1=scan.nextLine();

}


}
```

Character

char var1=‘G’;

char var2=‘O’;

char var3=‘O’;

char var4=‘D’;

Print them using single print statement

Concatenation?

Storing inside a string variable?

One possible Solution

```
str = String.valueOf(var1)+String.valueOf(b)+String.valueOf(c);
```


Try it with your name

Boolean

boolean b1=true/false;

Take an integer number as input from user

If the number is multiple of 2, set the flag variable to True

Now If the flag is True print Multiple of 2 otherwise print not a multiple of 2

Memory

- All the primitive types get memory from stack

Declaring Variables

- Declaring a variable is for:
  - Setting the identifier
  - Type of value
  - Initial value that it takes
  - Exp: int a; , char c; , boolean b;

Declare a variable and print it without assigning any values

Dynamic Initialization

- Use of Math class
- Values are not always assigned as a constant, there could be a method call etc

```
// Demonstrate dynamic initialization.
class DynInit {
  public static void main(String args[]) {
    double a = 3.0, b = 4.0;

    // c is dynamically initialized
    double c = Math.sqrt(a * a + b * b);

    System.out.println("Hypotenuse is " + c);
  }
}
```

Scope and lifetime of a variable

- Scope defines the visibility of your variable along with its lifetime
- A block defines a new scope
- Method’s scope is within curly braces:
  - Defining a variable inside method limits its scope to outside world
  - Concept of Local Variable

Scope

```
// Demonstrate block scope.
class Scope {
  public static void main(String args[]) {
    int x; // known to all code within main

    x = 10;
    if(x == 10) { // start new scope
      int y = 20; // known only to this block

      // x and y both known here.
      System.out.println("x and y: " + x + " " + y);
      x = y * 2;
    }
    // y = 100; // Error! y not known here

    // x is still known here.
    System.out.println("x is " + x);
  }
}
```

Lifetime

```
// Demonstrate lifetime of a variable.
class LifeTime {
  public static void main(String args[]) {
    int x;

    for(x = 0; x < 3; x++) {
      int y = -1; // y is initialized each time block is entered
      System.out.println("y is: " + y); // this always prints -1
      y = 100;
      System.out.println("y is now: " + y);
    }
  }
}
```

Same name issue

```
// This program will not compile
class ScopeErr {
  public static void main(String args[]) {
    int bar = 1;
    {                    // creates a new scope
      int bar = 2;  // Compile-time error - bar already defined!
    }
  }
}
```

Command Line Arguments

```
import java.util.Scanner;
class DataTypes{

public static void main(String var[]){

for(int i=0; i<var.length; i++){
System.out.println(var[i]);
}


}


}
```

```
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\92306\Desktop\Aror Uni\JAVA>javac DataTypes.java

C:\Users\92306\Desktop\Aror Uni\JAVA>java DataTypes Argument1 0 1 Argument4
Argument1
0
1
Argument4

C:\Users\92306\Desktop\Aror Uni\JAVA>
```

Arrays

- Grouping of related(homogenous) data
- Each element is accessed:
    - Via Index (starting from zero)

Array Declaration

type var-name[ ];

```
int month_days[];
```

Allocation of memory with new

array-var = new *type* [*size*];

```
month_days = new int[12];
```

Access without assigning values to array elements

- Numeric data types with a zero value
- Boolean with false
- Reference types with null values

Assigning and printing values

```
month_days[1] = 28;
```

The next line displays the value stored at index 3:

```
System.out.println(month_days[3]);
```

```
// Demonstrate a one-dimensional array.
class Array {
  public static void main(String args[]) {
    int month_days[];
    month_days = new int[12];
    month_days[0] = 31;
    month_days[1] = 28;
    month_days[2] = 31;
    month_days[3] = 30;
    month_days[4] = 31;
    month_days[5] = 30;
    month_days[6] = 31;
    month_days[7] = 31;
    month_days[8] = 30;
    month_days[9] = 31;
    month_days[10] = 30;
    month_days[11] = 31;
    System.out.println("April has " + month_days[3] + " days.");
  }
}
```

Putting it all to gather

Combine declaration and allocation

```
int month_days[] = new int[12];
```

```
// An improved version of the previous program.
class AutoArray {
  public static void main(String args[]) {

    int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
                         30, 31 };
    System.out.println("April has " + month_days[3] + " days.");
  }
}
```

Array Initializer

- List of comma separated values, surrounded by curly braces
- Array size auto decided, according to number of elements

Write array program to store months in year

Write Average Program using array of 5 elements, and new keyword

Arrays Task

- Write a program using arrays:
  - Create student_names array which holds names of any 5 students, the names will be input by the user
  - Create student_marks array which holds marks of those 5 students, again input by the user
  - Print it in following format
    - Name Marks
    - Ali 50
    - Ahmed 60

Multidimensional Arrays

- Array of Arrays
- Normally we will stick to 2D Array

```
int twoD[][] = new int[4][5];
```

Code Demonstration

```
// Demonstrate a two-dimensional array.
class TwoDArray {
  public static void main(String args[]) {
    int twoD[][]= new int[4][5];
    int i, j, k = 0;

    for(i=0; i<4; i++)
      for(j=0; j<5; j++) {
        twoD[i][j] = k;
        k++;

      }

    for(i=0; i<4; i++) {
      for(j=0; j<5; j++)
        System.out.print(twoD[i][j] + " ");
      System.out.println();
    }
  }
}
```

Allocate second Dimension Manually

```
int twoD[][] = new int[4][];
twoD[0] = new int[5];
twoD[1] = new int[5];
twoD[2] = new int[5];
twoD[3] = new int[5];
```

Alternatives

```
int a1[] = new int[3];
int[] a2 = new int[3];
```

The following declarations are also equivalent:

```
char twod1[][] = new char[3][4];
char[][] twod2 = new char[3][4];
```

This alternative declaration form offers convenience when declaring several arrays at the same time. For example,

```
int[] nums, nums2, nums3; // create three arrays
```

creates three array variables of type **int**. It is the same as writing

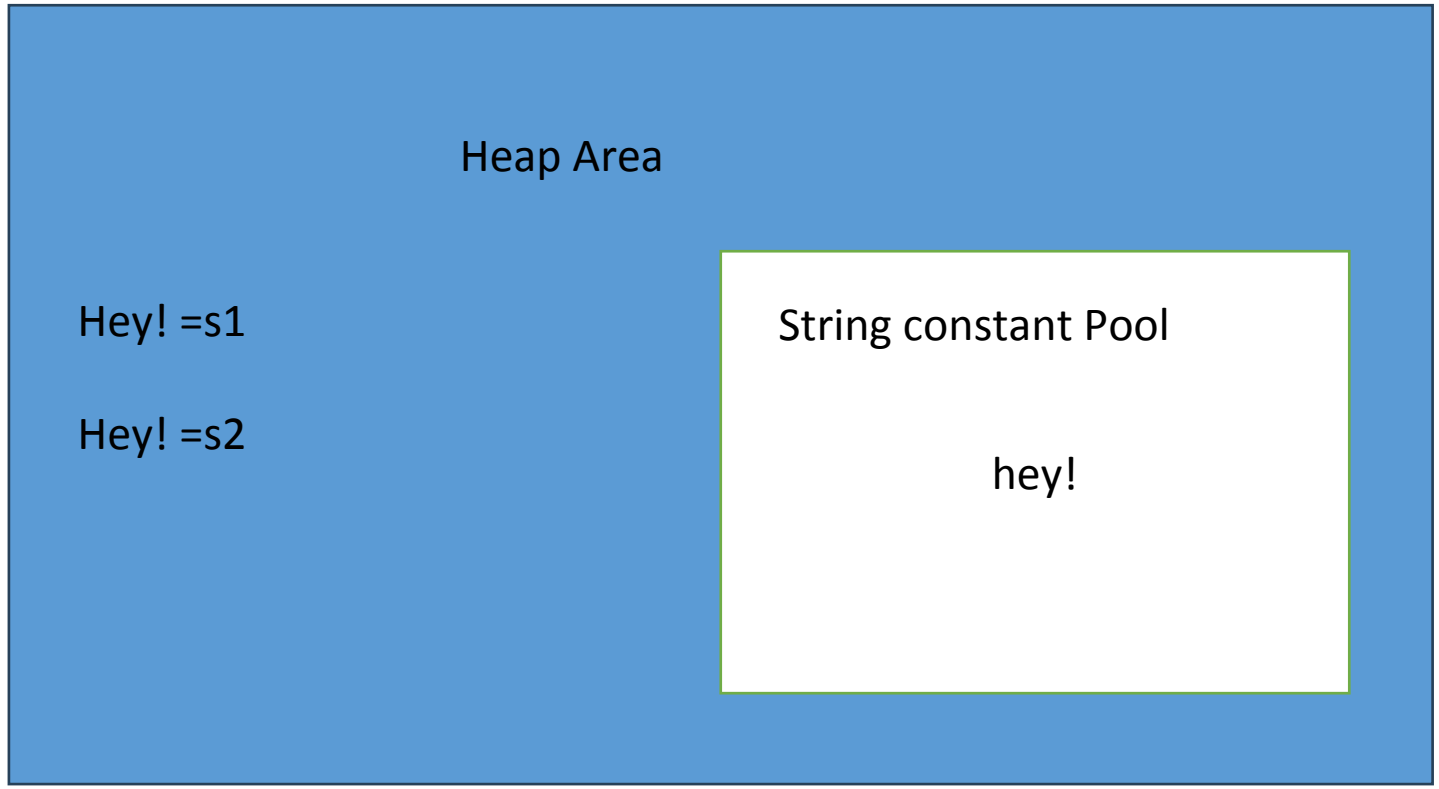
```
int nums[], nums2[], nums3[]; // create three arrays
```

Arrays

- Declaration
- Initialization
- Multi Dimensional Arrays

Strings

- Not a primitive type
- Rather it is an object in java


Heap Area
Hey! =s1
Hey! =s2
String constant Pool
hey!

Copying Arrays

- Copying One Array to Other
- = operator
- Loop to copy

Type conversion

- You assign a value of one data type to another:
  - Two types might not be compatible or might be
- If Data types are compatible:
  - Java will perform the conversion automatically known as Automatic Type Conversion
- If not then they need to be cast or converted explicitly.
  - For example, assigning a long value to an int variable.

Datatype	Bits Acquired In Memory
boolean	1
byte	8 (1 byte)
char	16 (2 bytes)
short	16(2 bytes)
int	32 (4 bytes)
long	64 (8 bytes)
float	32 (4 bytes)
double	64 (8 bytes)

Widening or Automatic Type Conversion

- Automatically done by Java
- When:
  - Two Data Types are compatible
    - Like numeric types
    - Numeric to boolean is incompatible
  - Assign the value of smaller dtype to bigger dtype

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

```java
// Main class
class GFG {

    // Main driver method
    public static void main(String[] args)
    {
        int i = 100;

        // Automatic type conversion
        // Integer to long type
        long l = i;

        // Automatic type conversion
        // long to float type
        float f = l;

        // Print and display commands
        System.out.println("Int value " + i);
        System.out.println("Long value " + l);
        System.out.println("Float value " + f);
    }
}
```

Narrowing or Explicit conversion

- Larger data type to Smaller Data type:
  - Useful for incompatible types

Error (int 4 bytes, char 2 bytes)

```java
// Java program to illustrate Incompatible data Type
// for Explicit Type Conversion

// Main class
public class GFG {

    // Main driver method
    public static void main(String[] argv)
    {

        // Declaring character variable
        char ch = 'c';
        // Declaringinteger variable
        int num = 88;
        // Trying to insert integer to character
        ch = num;
    }
}
```

```java
// Main class
public class GFG {

    // Main driver method
    public static void main(String[] args)
    {

        // Double datatype
        double d = 100.04;

        // Explicit type casting by forcefully getting
        // data from long datatype to integer type
        long l = (long)d;

        // Explicit type casting
        int i = (int)l;
```