Basic Java

Unit 2 – Language Fundamentals

Pratian Technologies (India) Pvt. Ltd.

www.pratian.com

PRATIAN TECHNOLOGIES

Topics

- Keywords
- Identifiers
- Data Types
- Literals
- Type Conversion
- Formatted Output
- Operators
- Comments
- Conditional Constructs
- Looping Constructs
- Break and Continue
- Processing Arrays
- Recursion

Keywords

Keywords are standard words that constitute Java.

- Have pre-defined meaning and cannot be redefined.
- Are always in lowercase.

There are about 50 keywords in Java.

Identifiers

Identifiers are user-defined names that are given to variables, functions, arrays, classes, etc.

Naming Rules

Data Types

Data type determines the values that a variable can contain and the operations that can be performed.

Two types:

- Primitive Types
- Reference Types

Primitive Types

- There are eight primitive types in Java.
- The size of these data types are independent of the platform.

Primitive type	Size
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

Literals

Literals are values assigned to identifiers.

Examples of literals:

- 3L
  Long literals take constant L.
- 100
  Integer literal
- 98.6 or 98.6D
  Double literals optionally take double constant D.
- 98.6f
  Float literals take floating point constant f.
- ‘A’
  Character literals are enclosed in ‘ ’.
- “This is a test”
  String literals are enclosed in “ ”.

Declaring Variables

General format of variable declaration: `type var1, var2,…… varN;`

Few examples:

- `int num1, num2, sum;`
- `char ch;`
- `double x, y;`

You can also initialize the variables at the time of declaration:

- `int a = 10 , b = 20;`
- `char ch = ‘A’;`

Sample Code : PrimitiveDemo.java

PRATIAN TECHNOLOGIES

Knowledge Check

Question

Write a program to swap the values of two numbers. Display the numbers before swapping and display them again after values are swapped.

Type Casting of Primitives

A primitive of one data type can be cast to another data type in Java.

- Casting is possible, if the two data types are compatible.
- Casting is implicit, if destination type is larger than source type.
- Casting needs to be explicit, if the destination type is smaller than source type. This may lead to loss of data.

Sample Code : PrimitiveTypeCast.java

PRATIAN TECHNOLOGIES

Knowledge Check

Question

Write a program to extract the whole and decimal parts of a fractional number by developing a 'DecimalSplitter' class with methods:

- `getWhole(double d)`: Contains logic of extracting the whole part of d
- `getFraction(double d)`: Contains logic of extracting the fractional part of d

The ‘String’ data type

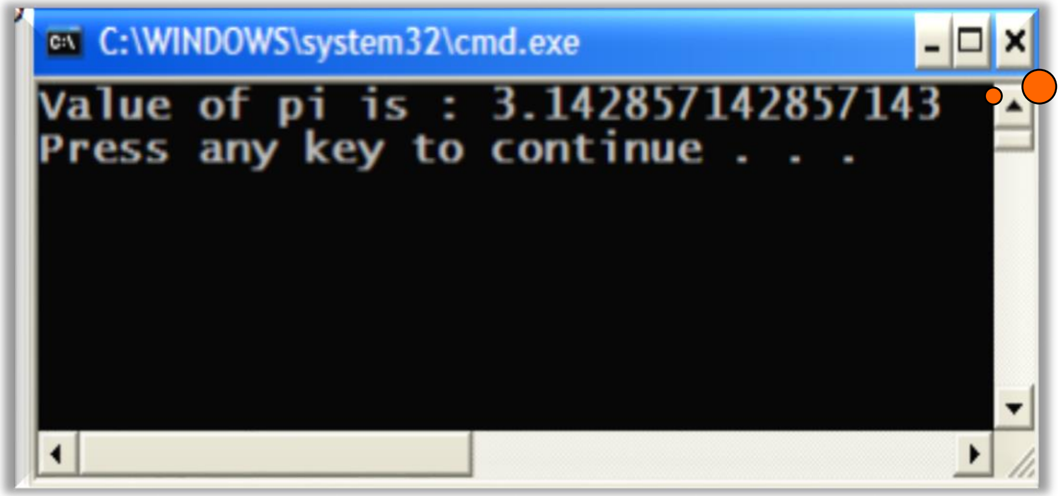
- Management of data containing multiple characters can be done through a String object
- The built-in library class ‘String’ provides support for representing string of characters and performing basic operations on them.
- Concatenation of strings is done by using the ‘+’ operator.

Sample Code : StringDemo.java

Formatted Output

Consider the below code and its output:

```
class Unformatted
{
          public static void main(String[] args)
          {
         double pi = 22.0/7;
         System.out.println ("Value of pi is : " + pi);
          }
}
```



```
C:\WINDOWS\system32\cmd.exe
Value of pi is : 3.142857142857143
Press any key to continue . . .
```

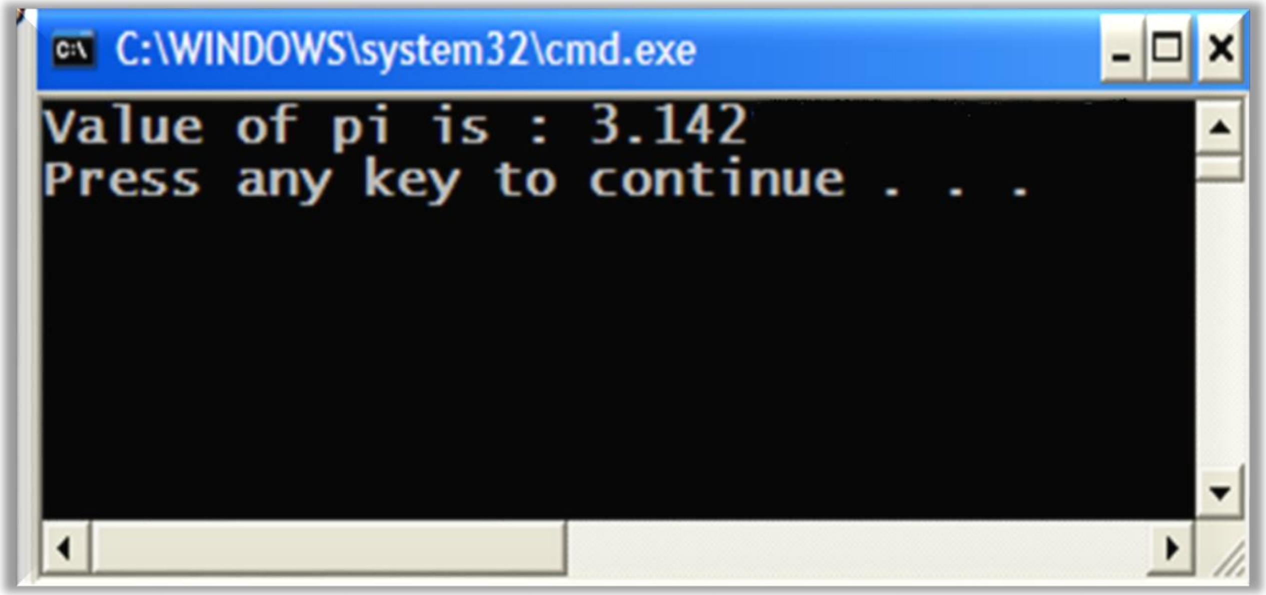
What if you want to display the value as 3.143?

Formatted Output

- Java 1.5 introduces the C-style `printf()` method for making formatted output.
- `System.out.printf()` method lets you format a string, in accordance with a format specifier, before making an output on the console.
- Each argument to be formatted is described by using a string that begins with % and ends with the formatted object's type.
- Simplest overloaded form of this method:
  - **`printf (String format, Object... args)`**
  - `printf (“%d”,i)`   // where i is an integer
- Most of C’s string formats are available in Java.

Formatted Output

```
class Unformatted
{
        public static void main(String[] args)
        {
        double pi = 22.0/7;
        System.out.printf ("Value of pi is : %.3f" , pi);
        }
}
```



```
Value of pi is : 3.142
Press any key to continue . . .
```

Formatted Output

The table below lists some important format specifiers:

Specifier	Description
%n	Outputs the line separator for the platform
%d	▪ Formats the value as a base-10 integer ▪ Arguments must be Byte, Short, Integer, and Long
%f	▪ Formats the value as a floating-point number in base-10 without exponential notation ▪ Arguments must be Float or Double
%e	▪ Formats the value as a base-10 floating-point number by using exponential notation ▪ Arguments must be Float or Double
%%	Escape sequence to allow printing of % in a String
%c	▪ Formats the value supplied as a single character ▪ Supplied value must be a Byte, Short, Character, or Integer

Formatted Output

- `System.out.format()` method performs the same function as `System.out.printf()` and they can be used interchangeably.
- For a complete list of all format specifiers, please refer to java docs for class Formatter.
- Sample Code : PrintfDemo.java

Java Language Operators

Operators are used to perform a function on variables.

Types of Operators	Description	Examples
Unary	Requires only one operand	num++;
Binary	Requires two operands	num1 = num2;
Ternary	Requires three operands	num1>0 ? num1=100 : num1=200;
Arithmetic	Is used for all floating-point and integer numbers	+ (addition), - (subtraction), * (multiplication), / (division), and % (modulo)
Relational and Conditional	Relational operator compares two values and determines the relationship between them.	Relational: >, <, <=, >=, ==, != Conditional: &&, ||
Bitwise and Logical	Bitwise operator allows you to perform bit manipulation on data;	Bitwise: <<, >>, |, &, ^, ~
Assignment	Is used to assign one value to another	=

Operator Precedence

- The operators in this table are listed in precedence order: the higher an operator appears, the higher its precedence.
- Operators with higher precedence are evaluated before operators with a relatively lower precedence.
- Operators on the same line have equal precedence.

Operators	Examples
postfix	[ ] . (params) expr++ expr-
unary	++expr –expr +expr –expr ~ !
creation or cast	New (type)expr
multiplicative	* / %
additive	+ -
shift	<< >>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
conditional	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>=

PRATIAN TECHNOLOGIES

Knowledge Check

<u>Question</u>

Evaluate the following expression: a + b – c * d / e, wherein a = 2, b = 2, c = 1, d = 3, and e = 10.

Knowledge Check

Question 2

Write a program to accept a number and display whether a number is odd or even using the ternary operator.

Question 2

Find the largest of three numbers using ternary operator:

- Develop a ‘LargestFinder’ class with the following method:

```
public int getLargest (int num1, int num2, int num3)
```

  - Also, write the logic of finding the largest of three numbers using the ternary operator.

Comments

Three kinds of comments:

- Double slashes
    - // Single line comment
- C-style
    - /* C-style comment

      Is used to comment out multiple

      lines */
- Javadoc comments
    - Used to generate documentation

      /** This class displays a text string at

      * the console.

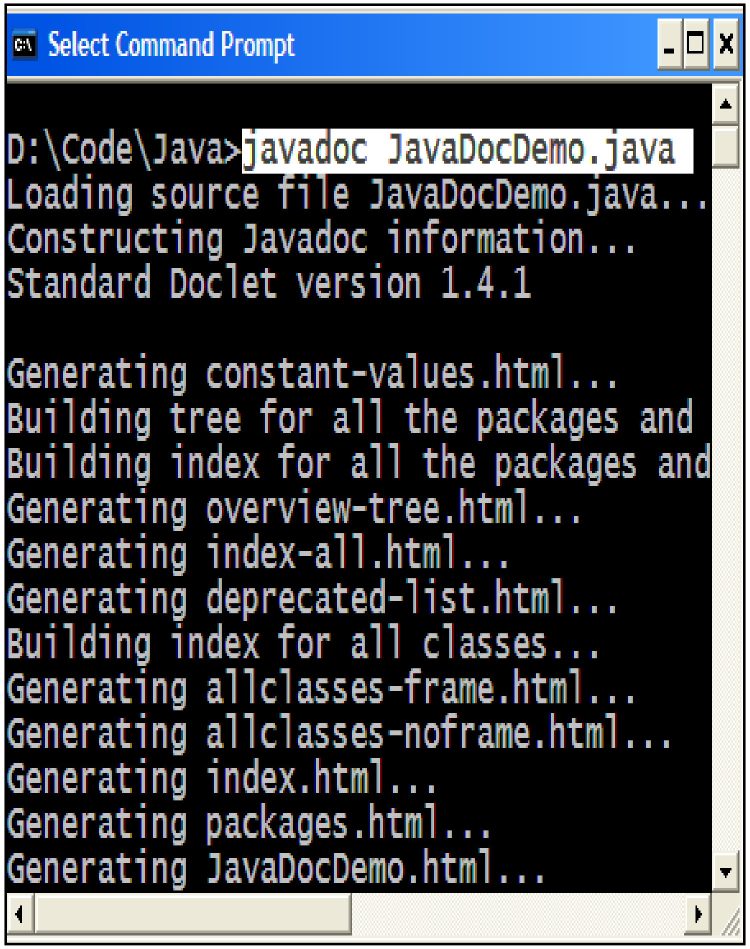
      */

Javadoc Comments

Sample Javadoc comments:

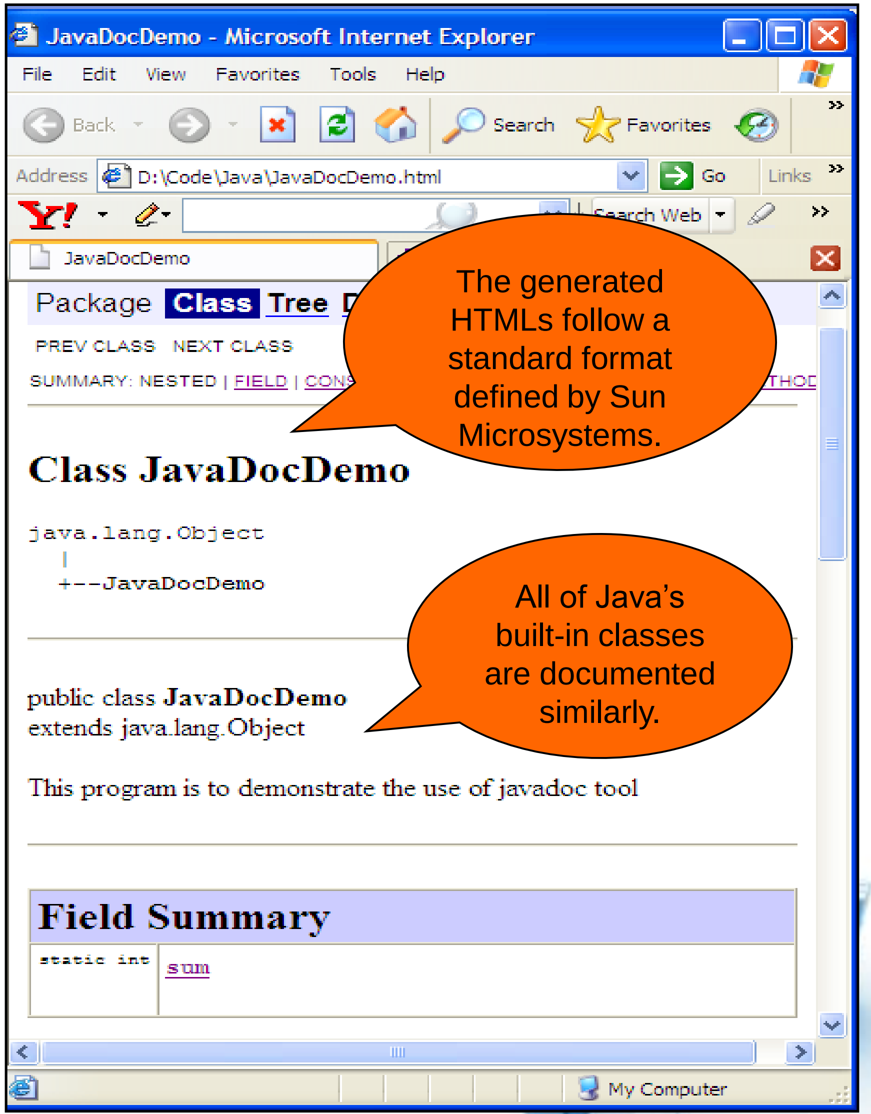
```
/** This program is to demonstrate the use of javadoc tool
     @author Chandra
    */
public class  JavaDocDemo
{
    /** This variable holds the value of the computed sum
    */
    public static int sum = 0;
    /** This is the main() method
       This is the point at which execution starts */
    public static void main(String[] args)
    {
         for(int i=1;i<=5;i++)
                sum = sum + i;
         System.out.println(sum);
    }
}
```

Generating Java Docs


Select Command Prompt

```
D:\Code\Java>javadoc JavaDocDemo.java
Loading source file JavaDocDemo.java...
Constructing Javadoc information...
Standard Doclet version 1.4.1

Generating constant-values.html...
Building tree for all the packages and
Building index for all the packages and
Generating overview-tree.html...
Generating index-all.html...
Generating deprecated-list.html...
Building index for all classes...
Generating allclasses-frame.html...
Generating allclasses-noframe.html...
Generating index.html...
Generating packages.html...
Generating JavaDocDemo.html...
```


JavaDocDemo - Microsoft Internet Explorer

File Edit View Favorites Tools Help
Back Search Favorites
Address D:\Code\Java\JavaDocDemo.html Go Links
JavaDocDemo
Package Class Tree
PREV CLASS NEXT CLASS
SUMMARY: NESTED | FIELD | CONS
My Computer

Class JavaDocDemo

```
java.lang.Object
  |
  +--JavaDocDemo
```

public class **JavaDocDemo**
extends java.lang.Object

This program is to demonstrate the use of javadoc tool

Field Summary

static int	sum

The generated HTMLs follow a standard format defined by Sun Microsystems.

All of Java's built-in classes are documented similarly.

Conditional Statements

- Some statements are executed only if certain conditions are met
  - A condition is represented by a logical (Boolean) expression that has a value of either true or false.
  - A condition is met, if it evaluates to true.

- For decision making:
  - if – else
  - switch – case

if-else Statement

- **if – else statement**

```
if (condition)
{
     statements
}
else
{
    statements
  }
```

- **Nested if–else statement**

```
if (condition)
{
     statements
}
else if (condition)
{
    statements
   }
   else
   {
    statements
   }
```

PRATIAN TECHNOLOGIES

Knowledge Check

Question

Display a student’s result by:

- developing a class ‘TestResult’ with the following method

```
public String getResult( int marks, int marks2, int marks3).
```

  - Write the logic of returning the result as ‘First Class’, ‘Second Class’, ‘Pass Class’, or ‘Fails’ based on the average marks secured.

switch –case Statement

- A switch statement is a multi-way decision maker that tests the value of an expression against a list of values.
- When a match is found, the statements associated with that value are executed.
- **Syntax**:

```
          switch(variable)
          {
              case value-1 : statements
break;
              case value-2 : statements
break;
              default : statements
          }
```

Note:
Variable can only be of type int, short, byte, char or enum.

Structured Loops

- A loop maybe defined as a set of statements that are repeatedly executed.
- A loop permits the repeated execution of a sequence of statements while some condition is true.
- There are three types of loops:
  - while loop
  - do…while loop
  - for loop

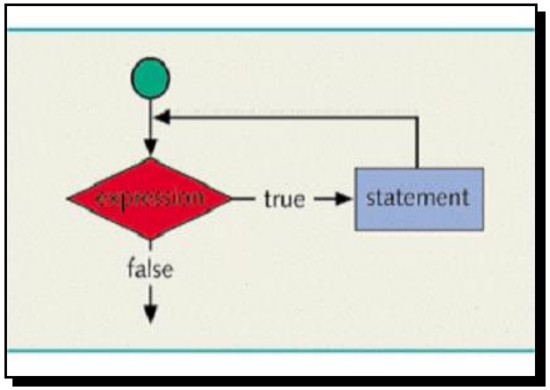
while Loop

- The general form of the while statement is:

```
while(expression)
{
    statement….
}
```

- The loop is entered when the expression is true.
- The loop is terminated when the expression is false.
- This is a pre-tested loop.
- This loop need not be compulsorily executed once.

```
int i =1;
while(i<=10)
{
    System.out.print(i+"\t");
    i++;
}
```



Knowledge Check

Question

Display a number in words by:

- developing a class ‘NumToWordsConverter’ with the following method

`public String numToWords(int number).`

  - Write the logic of constructing a string that represents the number in words.

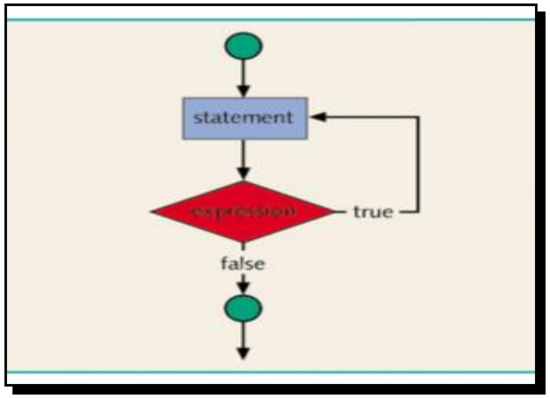
do… while Loop

- The general form of the do…while statement is:

```
do
{
    statement….
} while(expression);
```

- This loop will be executed at least once.
- The loop is re-entered when the expression is true.
- The loop is terminated when the expression is false.
- This is a post-tested loop.

```
int i =1;
do
{
System.out.println(i);
i++;
} while(i<=10);
```



PRATIAN TECHNOLOGIES

Knowledge Check

Question

Accept user’s choice and perform mathematical computation.

for Loop

- The general form of the for statement is:

  for (init counter; testcondition; re-valuation counter)

  {

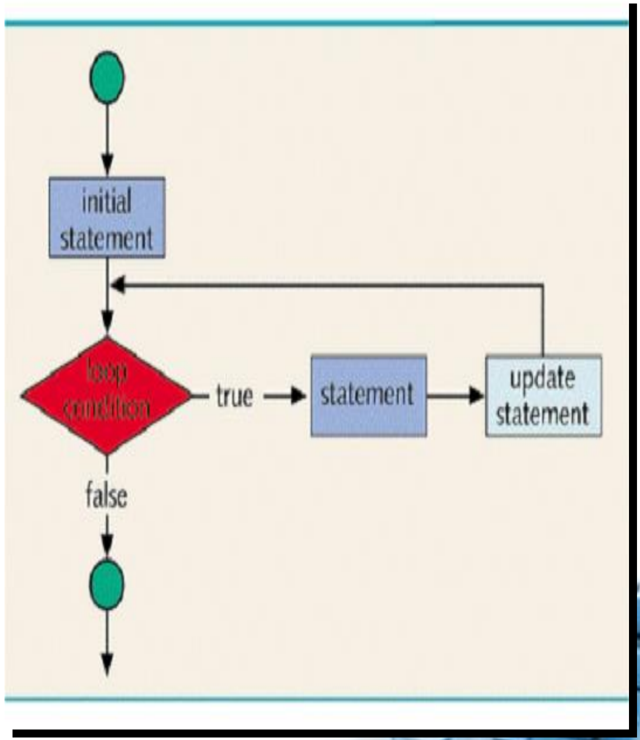
  statement…..

  }

Steps involved in execution:

- The initial statement executes.
- The loop condition is evaluated.
- If the loop condition evaluates to true:
  - execute the for loop statement.
  - execute the update statement (the third expression in the parentheses).
- Repeat the previous step until the loop condition evaluates to false.

```
for(int i=0;i <=10;i++)
{
  System.out.println(i);
}
```



Knowledge Check

Question

- Display the following pattern

```
*
*   *
*   *   *
*   *   *  *
```

break Statement

- The break statement, when executed in a switch structure, provides an immediate exit from the switch structure.
- When the break statement executes in a repetition structure, it immediately exits from these structures.

The break statement is typically used for two purposes:

- To exit early from a loop
- To skip the remainder of the switch structure

- After the break statement executes, the program continues to execute with the first statement after the structure.
- The use of a break statement in a loop can eliminate the use of certain (flag) variables.

Syntax:

```
    while(condition)
{
    statement;
    if(condition)
            break;
    statement;
}
statement;
```

Knowledge Check

Question

- Write a program to find the sum of all the prime numbers in the range n to m.

Display each prime number and also the final sum.

continue Statement

- The continue statement is used in while, for, and do… while structures.
- In a while and do… while structure, the expression (loop-continue test) is evaluated immediately after the continue statement.
- In a for structure, the update statement is executed after the continue statement, and then the loop condition executes.

When the continue statement is executed in a loop, it skips the remaining statements and proceeds with the next iteration of the loop.

Syntax:

```
    while(condition)
{
    statement;
    if(condition)
            continue;
    statement;
}
```

Knowledge Check

Question

- Write a program to display the 1st, 2nd, and 4th multiple of 7, which gives the remainder 1 when divided by 2, 3, 4, 5, and 6.

Arrays

- Array is a collection of a fixed number of components wherein all of the components are of the same data type.
  - It is a homogeneous data type.

- Whenever an array is used, the subscript or the index is involved. The value to be stored or retrieved from the array has to be specified by using both the name of the array and the subscript.

	0	1	2	3	4	5	6	7	8	9
arr =	5	10	15	20	25	30	35	40	45	50

```
arr[0] = 5;
System.out.println(arr[0]);
```

In an array of size ‘n’, the first subscript is always 0 and the last subscript is n – 1.

PRATIAN TECHNOLOGIES

Declaring an Array

- To declare an array that can hold integer

```
int[ ] arrayOfInts;                                ed

int arrayOfInts[ ];                                ed

int[ ] arrayOfInts= new int[10]; //              ted
```

NOTE
Without use of 'new' keyword, the array is not allocated memory

- **Note**: Without use of 'new' keyword, the array is not allocated memory.
- Arrays can contain any legal Java data type including reference types such as objects or other arrays.
- For example, the following declares an array that can contain ten Customer objects:
- `Customer[ ] arrayOfCust = new Customer[10];`

Initializing an Array

- We can store values into an array by using the index to specify position:

```
int a[ ] = new int[5];
a[0] = 10;
a[1] = 20;
a[2] = 30;
a[3] = 40;
a[4] = 50;
```

- An array can also be initialized at the time of declaration:

```
int iArray[ ] = { 2, 3, 5, 7, 11, 13 };
```

Accessing Elements of an Array

- Elements of an array can be accessed by looping through the array, and accessing the element stored at a particular index:

```
int[ ] arr = new int[5];
for(int i=0;i<arr.length;i++)
      System.out.print(arr[i] + "\t");
```

- Sample Code : ArrayDemo.java

PRATIAN TECHNOLOGIES

Knowledge Check

Question

- Write a program to store N elements in an array of integer. Display the elements. Accept a number to be searched. Display whether the number is found or not in the array (LINEAR SEARCH).

Multi-Dimensional Arrays

- An array where elements can be accessed by using more than one subscript is known as multi-dimensional array.

- A two-dimensional array can be logically visualized as a collection of rows and columns, like in a matrix.

$$\begin{array}{c|c|c|c|}
 & [0] & [1] & [2] \\ \hline
a[0] & 100 & 200 & 300 \\ \hline
a[1] & 400 & 500 & 600 \\ \hline
a[2] & 700 & 800 & 900 \\ \hline
\end{array} \quad \nearrow \text{a[1][2]}$$

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

Logically, the array can be visualized as a 3x3 matrix.

A two-dimensional array requires two subscript for accessing its elements.

Knowledge Check

Question

Check if a given square matrix is identical and / or symmetric by:

- Developing a MatrixWizard class with the following methods:
- `public boolean isIdentity(int[ ][ ] matrix)`

  (Check if the matrix is identity or not and return a boolean value indicating it
- `public boolean isSymmetric(int[ ][ ] matrix)`

  (Check if the matrix is symmetric or not and return a boolean value indicating it)

Recursion

- Recursion is an algorithmic technique where a function/ method, in order to accomplish a task, calls itself with some part of the task.

Example:

```
public void recurse (int n)
{
System.out.println(n);
recurse(n-1);
}
```

- Method call should terminate at some point, else the recursive method will be called infinitely.

```
public void recurse(int n)
{
 System.out.println(n);
 recurse(n-1);
 if(n == 0)
    return;
}
```

Recursion

- Consider finding the sum of elements in an array from a start index to the last element in the array.
- This can be recursively performed.
- Sample Code : RecursionDemo.java

Knowledge Check

Question

Find the factorial of a number using recursion by:

- Develop a class FactorialGenerator with the following method

```
public int getFactorial(int num)
```

- Write the logic of finding the factorial by using the technique of recursion.

Question time

Please try to limit the questions to the topics discussed during the session. Thank you.

Keywords

Keywords are standard words that constitute Java.

- Have pre-defined meaning and cannot be redefined.
- Are always in lowercase.

There are about 50 keywords in Java.

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

Identifiers

Identifiers are user-defined names that are given to variables, functions, arrays, classes, etc.

Naming Rules

- Only numeric and alphabetic characters are permitted
- Special characters permitted are '_' (underscore) and '$' (dollar)
- Should not start with a digit
- Case-sensitive; uppercase and lowercase letters are distinct
- Keyword cannot be used as a variable name

```
int age;
class Employee
{
//…
}
void setName(Stringname)
int [ ] scores = new int [10];
```

Knowledge Check

Knowledge Check

Question

Identify legal and illegal identifiers from the following (state your reasons):

- ❑ first
- ❑ Employee Salary
- ❑ Conversion
- ❑ Hello!
- ❑ One+two
- ❑ $test
- ❑ 2nd
- ❑ _myName
- ❑ Employee

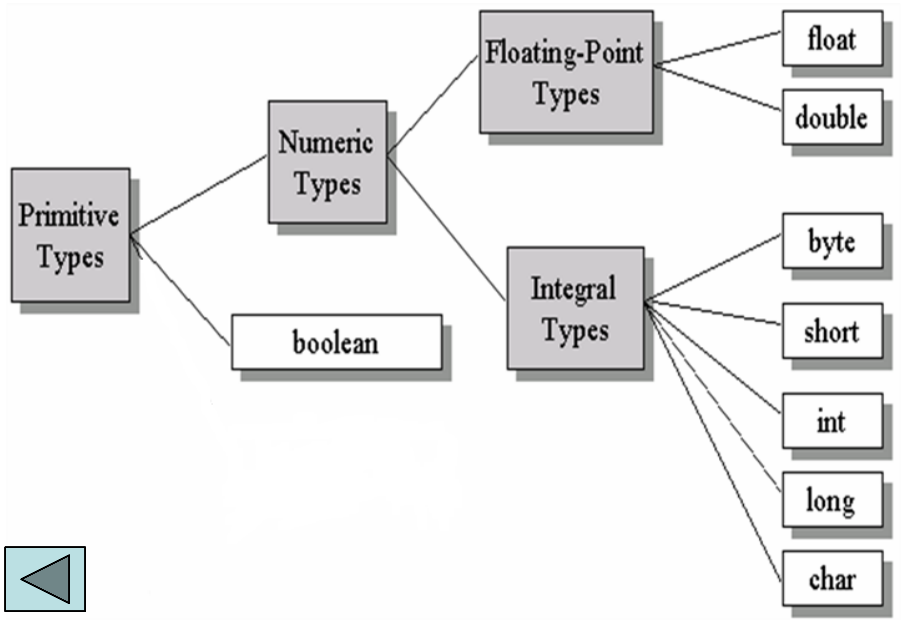
Data Types

Data type determines the values that a variable can contain and the operations that can be performed.

Two types:

- Primitive Types
- Reference Types

Classification



Examples

- 100
- 23.667
- 'c'
- true

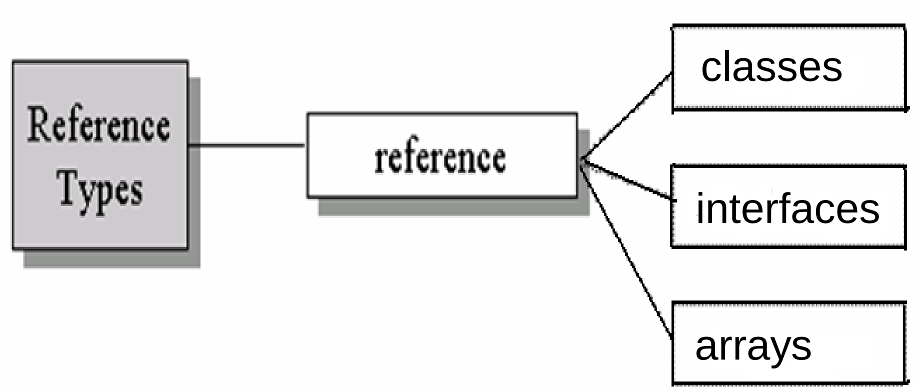
Data Types

Data type determines the values that a variable can contain and the operations that can be performed.

Two types:

- Primitive Types
- Reference Types

Classification



Examples

- customer
- intArr[ ]
- empArr[ ]

PRATIAN TECHNOLOGIES

Primitive Types

- There are eight primitive types in Java.
- The size of these data types are independent of the platform.

Primitive type	**Size**
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

boolean
- Can take values true or false.
- Cannot be used interchangeably with 1 and 0

Primitive Types

- There are eight primitive types in Java.
- The size of these data types are independent of the platform.

Primitive type	Size
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

char
• Unicode character set

PRATIAN TECHNOLOGIES

Primitive Types

- There are eight primitive types in Java.
- The size of these data types are independent of the platform.

Primitive type	Size
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

byte
• Takes values in the range -128 to 127

Primitive Types

- There are eight primitive types in Java.
- The size of these data types are independent of the platform.

Primitive type	Size
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

short
- Takes values in the range -32768 to 32767

Primitive Types

- There are eight primitive types in Java.

- The size of these data types are independent of the platform.

Primitive type	Size
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

int
- All integer values are signed int
- Takes values in the range -2,147,483,648 to 2,147,483,647

Primitive Types

- There are eight primitive types in Java.
- The size of these data types are independent of the platform.

Primitive type	Size
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

long

- Takes values in the range -9223372036854775808 to 9223372036854775807

Primitive Types

- There are eight primitive types in Java.
- The size of these data types are independent of the platform.

Primitive type	Size
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

float

- Takes floating point constant f
- Takes values in the range 1.40239846e-45 to 3.40282347e+38

Primitive Types

- There are eight primitive types in Java.
- The size of these data types are independent of the platform.

Primitive type	Size
boolean	1-bit
char	16-bit
byte	8-bit
short	16-bit
int	32-bit
long	64-bit
float	32-bit
double	64-bit

double
• Takes floating point constant d
•Takes values in the range 4.94065645841246544e-324 to 1.79769313486231570e+308

PRATIAN TECHNOLOGIES

Type Casting of Primitives

A primitive of one data type can be cast to another data type in Java.

- Casting is possible, if the two data types are compatible.
- Casting is implicit, if destination type is larger than source type.
- Casting needs to be explicit, if the destination type is smaller than source type. This may lead to loss of data.

- All numeric types are compatible with each other.
- Integers are compatible with characters.
- Boolean is not compatible with any of the data type.

Sample Code : PrimitiveTypeCast.java

Type Casting of Primitives

A primitive of one data type can be cast to another data type in Java.

- Casting is possible, if the two data types are compatible.
- Casting is implicit, if destination type is larger than source type.
- Casting needs to be explicit, if the destination type is smaller than source type. This may lead to loss of data.

For example: `int` to `double`, `int` to `long`, and `short` to `int`

Sample Code : PrimitiveTypeCast.java

Type Casting of Primitives

A primitive of one data type can be cast to another data type in Java.

- Casting is possible, if the two data types are compatible.
- Casting is implicit, if destination type is larger than source type.
- Casting needs to be explicit, if the destination type is smaller than source type. This may lead to loss of data.

For example: `double` to `int` and `long` to `int`

Sample Code : PrimitiveTypeCast.java