- Chapter 3

Java fundamentals and control structures

Review

- Java was introduced by Sun Microsystems in 1995.
- Java is a programming language popularly used to build programs that can work on the Net.
- Its primary features are: it is object-oriented and a cross platform language.
- Swing, Drag and Drop, Java 2D API, Java Sound and RMI are some of the features added to the existing version of Java.
- A Java applet is designed to work in a pre-defined “sandbox” only. This makes it safe to be used on the Internet.
- Java bytecodes are machine language instructions understood by the Java Virtual Machine and usually generated as a result of compiling Java language source code.

Review Contd…

- Java programs can be divided into following categories-applets, applications, GUI applications, servlets and database applications.
- Java visual development tools help the programmer to develop Java applications and applets more quickly and efficiently.
- The JDK contains the software and tools needed to compile, debug and execute applets and applications written in the Java language. It’s basically a set of command-line tools.
- Enhancement in Swing, AWT, a new I/O class and so on has been added in the latest version of Java 1.4.2.
- The future will use a lot of Java related programs for consumer gadgets with embedded technologies.

Objectives

- *Interpret the Java Program*
- *Understand the basics of Java Language*
- *Identify the Data Types*
- *Understand arrays*
- *Identify the Operators*
- *Format output using Escape Sequence*

A Sample Java program

```
// This is a simple program called First.java
class First
{
    public static void main (String [] args)
    {
        System.out.println ("My first program in Java ");
    }
}
```

Analyzing the Java Program

- The symbol // stands for commented line.
- The line `class First` declares a new class called **First**.
- `public static void main (String [] args)`
  - This is the main method from where the program begins its execution.
- `System.out.println ("My first program in java");`
  - *This line displays the string **My first program in java** on the screen.*

Compiling and executing the Java program

C:\WINNT\system32\cmd.exe

```
E:\Java\JavaExamples>javac First.java

E:\Java\JavaExamples>
```

The java compiler creates a file called '**First.class**' that contains the byte codes

To actually run the program, a java interpreter called `java` is required to execute the code.

C:\WINNT\system32\cmd.exe

```
E:\Java\JavaExamples>java First
My first program in Java

E:\Java\JavaExamples>
```

Passing Command Line Arguments

```
    class CommLineArg
    {
        public static void main (String [] pargs)
        {
                System.out.println("These are the arguments passed to the main
method.");
                System.out.println(pargs [0]);
                System.out.println(pargs [1]);
                System.out.println(pargs [2]);
        }
    }
```

Passing Command Line Arguments

Output

```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac CommLineArg.java

E:\Java\JavaExamples>java CommLineArg We Change Lives
These are the arguments passed to the main method
We
Change
Lives
```

Basics of the Java Language

- Classes & Methods
- Data types
- Variables
- Operators
- Control structures

Classes in Java

- Class declaration Syntax

```
class Classname
{
        var_datatype variablename;
        :

        met_datatype methodname(parameter_list)
        :
}
```

Sample class

Customer
CustomerName
Address
Email
Phone
Accept Customer Details
Display Customer Details

Data Types

Primitive Data Types

- byte
- char
- boolean
- short
- int
- long
- float
- double

Reference data types

- Array
- Class
- Interface

Type Casting

- In type casting, a data type is converted into another data type.
- Example

```
float c = 34.89675f;
int b = (int)c + 10;
```

Automatic type and Casting

- There are two type of data conversion: automatic type conversion and casting.
- When one type of data is assigned to a variable of another type then automatic type conversion takes place provided it meets the conditions specified:
  - The two types are compatible
  - The destination type is larger than the source type.
- Casting is used for explicit type conversion. It loses information above the magnitude of the value being converted.

Type Promotion Rules

- All `byte` and `short` values are promoted to `int` type.
- If one operand is `long`, the whole expression is promoted to `long`.
- If one operand is `float` then the whole expression is promoted to `float`.
- If one operand is `double` then the whole expression is promoted to `double`.

Variables

- Three components of a variable declaration are:
  - Data type
  - Name
  - Initial value to be assigned (optional)
- Syntax

  datatype identifier [=value][, identifier[=value]...];

Example

```
class DynVar
{
    public static void main(String [] args)
    {
            double len = 5.0, wide = 7.0;
            double num = Math.sqrt(len * len + wide * wide);
            System.out.println("Value of num after dynamic initialization is " + num);
    }
}
```

Output

```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac DynVar.java

E:\Java\JavaExamples>java DynVar
Value of num after dynamic initialization is 8.602325267042627
```

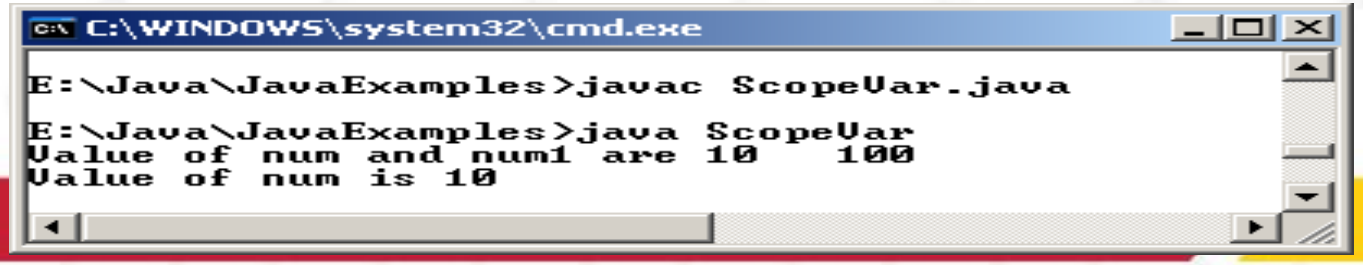
Scope and Lifetime of Variables

- Variables can be declared inside a block.
- The block begins with an opening curly brace and ends with a closing curly brace.
- A block defines a scope.
- A new scope is created every time a new block is created.
- Scope specifies what objects are visible to other parts of the program.
- It also determines the life of an object.

Example

```
class ScopeVar
{
  public static void main(String [] args)
  {
    int num = 10;
    if ( num == 10)
    {
            // num is available in inner scope
            int num1 = num * num;
            System.out.println("Value of num and num1 are " + num + "   " + num1);
    }
    //num1 = 10;  ERROR ! num1 is not known
    System.out.println("Value of num is " + num);
  }
}
```

Output



Array Declarations

- Three ways to declare an array are:
  - **datatype identifier [ ];**
  - **datatype identifier [ ] = new datatype[size];**
  - **datatype identifier [ ] = {value1,value2,….valueN};**

Example – One Dimensional Array

```
class ArrDemo
{
  public static void main(String [] arg)
  {
    double nums[] = {10.1, 11.3, 12.5,13.7, 14.9};
    System.out.println(" The value at location 3 is : " + nums[3]);
  }
}
```

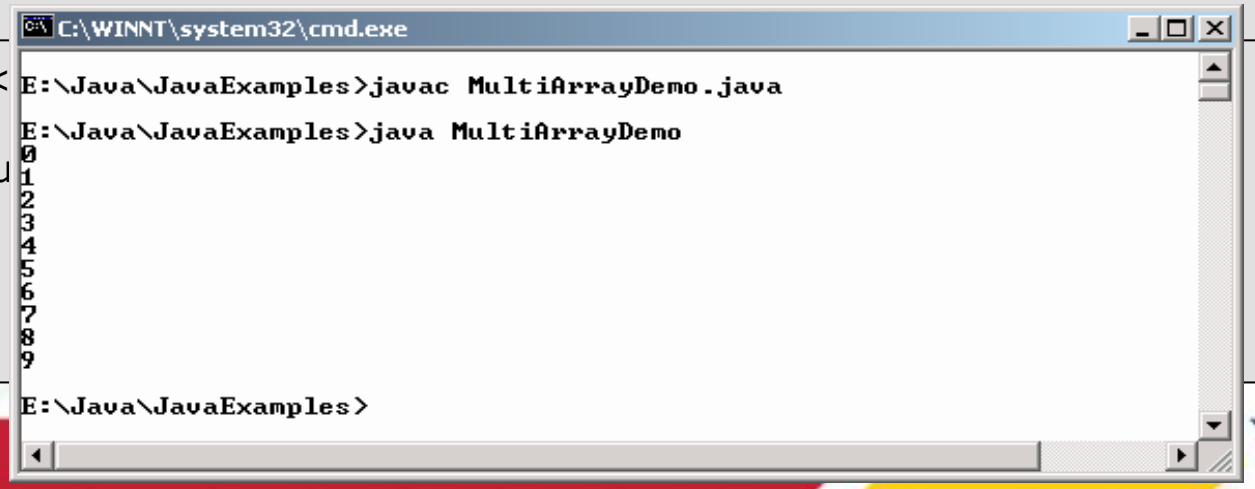
Output

```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac ArrDemo.java

E:\Java\JavaExamples>java ArrDemo
 The value at location 3 is : 13.7
```

Example – Multi Dimensional Array

```
class  MultiArrayDemo
        for (int count = 0; count < 4; count++)
        {
                for (int ctr = 0; ctr < count+1; ctr++)
                {
                        System.out.print(multi[count][ctr] + " ");
                        System.out.println();
                }
            }
        }
    }
        for (int ctr = 0; ctr <
        {
                multi[cou
                num++;
        }
    }
```

Output



Operators

- Arithmetic Operators
- Bitwise Operators
- Relational Operators
- Logical Operators
- Conditional Operators
- Assignment operators

Arithmetic Operators

- Operands of the arithmetic operators must be of numeric type.
- Boolean operands cannot be used, but character operands are allowed.
- These operators are used in mathematical expressions.

Example

```
class  ArithmeticOp
{
        public static void ma                    arg)
        {
                int num = 5, num1 = 12, num2 = 20, result;
                resul
                Syste
                resul
                Syste
                resul
                Syste
                Syste
                num
                Syste
                doub
                res =
                Syste
                res -
                Syste
                Syste
                res--
                Syste
        }
}
```

Output

C:\WINNT\system32\cmd.exe

```
E:\Java\JavaExamples>javac ArithmeticOp.java

E:\Java\JavaExamples>java ArithmeticOp
Sum of num and num1 is : (num + num1) 17
Modulus of num and num1 is : (num % num1) 5
Product  of result and num2 is :  (result *= num2) 100
Value of num before the operation is : 5
Value of num after ++ operation is : 6
num3 -- num4 is : 11.5
res -= 2.50 9.0
Value of res before  -- operation is : 9.0
Value of res after -- operation is : 8.0

E:\Java\JavaExamples>
```

Bitwise Operators

- A bitwise operator allows manipulation of individual bits in an integral primitive data type.
- These operators act upon the individual bits of their operands.
- Bitwise operators perform Boolean algebra on the corresponding bits in the two arguments to produce the result.

Relational Operators

- Relational operators test the relation between two operands.
- The result of an expression in which relational operators are used, is boolean (either true or false).
- Relational operators are used in control structures.

Example

```
class RelOp
{
    public static void main(String [] args)
    {
        float num = 10.0F;
        double num1 = 10.0;
        if (num == num1)
                System.out.println ("num is equal to num1");
        else
                System.out.println ("num is not equal to num1");
    }
}
```

Output

```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac RelOp.java

E:\Java\JavaExamples>java RelOp
num is equal to num1
```

Logical Operators

- Logical operators work with boolean operands.
- Some operators are
  - &&
  - ||
  - ^
  - !

Conditional Operators

- The conditional operator is unique, because it is a ternary or triadic operator that has three operands to the expression.
- It can replace certain types of if-then-else statements.
- The code below checks whether a commuter's age is greater than 65 and print the message.

```
CommuterCategory = (CommuterAge > 65)? "Senior Citizen" : "Regular";
```

Assignment Operators

- The assignment operator is a single equal sign, =, and assigns a value to a variable.
- Assigning values to more than one variable can be done at a time.
- In other words, it allows us to create a chain of assignments.

Operator Precedence

- Parentheses: ( ) and [ ]
- Unary Operators: +, -, ++, --, ~, !
- Arithmetic and Shift operators: *, /, %, +, -, >>, <<
- Relational Operators: >, >=, <, <=, ==, !=
- Logical and Bitwise Operators: &, ^, |, &&, ||,
- Conditional and Assignment Operators: ?=, =, *=, /=, +=, -=
- Parentheses are used to change the order in which an expression is evaluated. Any part of an expression enclosed in parentheses is evaluated first.

Formatting output with Escape Sequences

- Whenever an output is to be displayed on the screen, it needs to be formatted.
- The formatting can be done with the help of escape sequences that Java provides.
- **System.out.println ("Happy \tBirthday");**
  - Output: **Happy Birthday**

Control Flow

- All application development environments provide a decision making process called control flow statements that direct the application execution.
- Flow control enables a developer to create an application that can examine the existing conditions, and decide a suitable course of action.
- Loops or iteration are an important programming construct that can be used to repeatedly execute a set of actions.
- Jump statements allow the program to execute in a non-linear fashion.

Control Flow Structures in Java

- **Decision-making**
  - if-else statement
  - switch-case statement
- **Loops**
  - while loop
  - do-while loop
  - for loop

if-else statement

- The `if-else` statement tests the result of a condition, and performs appropriate actions based on the result.
- It can be used to route program execution through two different paths.
- The format of an `if-else` statement is very simple and is given below:

```
if (condition)
{
    action1;
}
else
{
    action2;
}
```

Example

```
class CheckNum
{
  public static void main(String [] args)
  {
    int num = 10;
    if (num % 2 == 0)
      System.out.println(num + " is an even number");
    else
      System.out.println(num + " is an odd number");
  }
}
```

Output

```
C:\WINNT\system32\cmd.exe

E:\Java\JavaExamples>javac CheckNum.java

E:\Java\JavaExamples>java CheckNum
10 is an even number

E:\Java\JavaExamples>
```

switch – case statement

- The switch – case statement can be used in place of `if-else-if` statement.
- It is used in situations where the expression being evaluated results in multiple values.
- The use of the `switch-case` statement leads to simpler code, and better performance.

Example

```
class SwitchDemo
        case 5:
                str = "Friday";
                        break;
        case 6:
                str = "Saturday";
                        break;
        default:
                str = "Invalid day";
            }
        System.out.println(str);
        }
    }
```

Output

```
= "Monday";
        break;
```

```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac SwitchDemo.java

E:\Java\JavaExamples>java SwitchDemo
Thursday
```

```
        break;
```

while Loop

- `while` loops are used for situations when a loop has to be executed as long as certain condition is True.
- The number of times a loop is to be executed is not pre-determined, but depends on the condition.
- The syntax is:

```
while (condition)
{
action statements;
.
.
.}
```

Example

```
class FactDemo
{
  public static void main(String [] args)
  {
    int num = 5, fact = 1;
    while (num >= 1)
    {
          fact *= num;
          num--;
    }
    System.out.println("The
  }
}
```

Output

```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac FactDemo.java

E:\Java\JavaExamples>java FactDemo
The factorial of 5 is : 120
```

do – while Loop

- The `do-while` loop executes certain statements till the specified condition is True.
- These loops are similar to the `while` loops, except that a `do-while` loop executes at least once, even if the specified condition is False. The syntax is:

```
do
{
  action statements;
.
.
} while (condition);
```

Example

```
class DoWhileDemo
{
  public static void main(String [] args)
  {
    int count = 1, sum = 0;
    do
    {
          sum += count;
          count++;
    }while (count <= 100);
    System.out.println("The sum of first 100 numbers is : " + sum);
  }
}
```

Output

The sum of first 100 numbers is : 5050

for Loop

- All loops have some common features: a counter variable that is initialized before the loop begins, a condition that tests the counter variable and a statement that modifies the value of the counter variable.
- The `for` loop provides a compact format for incorporating these features.

Syntax:

```
for (initialization statements;  condition; increment / decrement statements)
{
   action statements;
.
.
}
```

Example

```
class ForDemo
{
	public static void main(String [] args)
	{
		int count = 1, sum = 0;
		for (count = 1; count <= 10; count += 2)
		{
			sum += count;
		}
		System.out.println("The sum of first 5 odd numbers is : " + sum);
	}
}
```

Output

The sum of first 5 odd numbers is : 25

Jump Statements

- Three jump statements are:
  - break
  - continue
  - return
- The three uses of `break` statements are:
  - It terminates a statement sequence in a `switch` statement.
  - It can be used to exit a loop.
  - It is another form of `goto`.

Example

```
class BrDemoAppl
{
   public static void main(String [] args)
   {
      for (int count = 1; count <= 100; count++)
      {
            if (count == 10)
               break;
            System.out.println("The value of num  is : " + count);
      }
System.out.println("The loop is over");
   }
}
```

Output

```
The value of num  is : 1
The value of num  is : 2
The value of num  is : 3
The value of num  is : 4
The value of num  is : 5
The value of num  is : 6
The value of num  is : 7
The value of num  is : 8
The value of num  is : 9
The loop is over
```

Summary

- A Java program consists of a set of classes. A program may contain comments. The compiler ignores this commented lines.
- The Java program must have a `main()` method from where it begins its execution.
- Classes define a template for units that store data and code related to an entity.
- Variables defined in a class are called the instance variables.
- There are two types of casting:widening and narrowing casting.
- Variables are basic unit of storage.
- Each variable has a scope and lifetime.
- Arrays are used to store several items of same data type in consecutive memory locations.

Summary Contd…

- Java provides different types of operators. They include:
  - Arithmetic
  - Bitwise
  - Relational
  - Logical
  - Conditional
  - Assignment
- Java supports the following programming constructs:
  - `if-else`
  - `switch`
  - `for`
  - `while`
  - `do-while`
- The three jump statements-break,continue and return helps to transfer control to another part of the program.