MODULE 1: PROGRAMMING IN JAVA

Week 5 - 6

Most Essential Learning Competencies: *At the end of the course, you must be able to:*

1. Distinguish between instance and local variables
2. Recognize assignment compatibility and required casts in fundamental types
3. Apply if, switch, for, while, and do constructions and the labeled forms of break and continue as flow control structures in a program

Reading Activity

Java Programming Language Coding Conventions

- **Packages**

  com.example.domain;

- **Classes, interfaces, and enum types**

  SavingsAccount

- **Methods**

  getAccount()

- **Variables**

  currentCustomer

- **Constants**

  HEAD_COUNT

- **Control structures**

```
if ( condition) {
statement1;
} else {
statement2;
}
```

- **Spacing**

  Use one statement per line.
  Use two or four spaces for indentation.
  Comments:
  Use // to comment inline code.

Use /** documentation */ for class members.

Expressions and Flow Control

Variables and Scope

• Local variables are:

- *Variables that are defined inside a method and are called local, automatic, temporary, or stack variables*

• Variables that are created when the method is executed are destroyed when the method is exitedVariable initialization comprises the following:

- *Local variables require explicit initialization*
- *Instance variables are initialized automatically*

```
public class ScopeExample {
        private int i = 1;
        public void firstMethod() {
        int i = 4, j = 5;
        this.i = i + j;
        secondMethod(7);
}
        public void secondMethod(int i) {
        Int j = 8;
                this.i = i + j;
        }
}
        public class TestScoping {
        public static void main(String[] args) {
        ScopeExample scope = new ScopeExample();
        Scope.firstMethod();
}
```

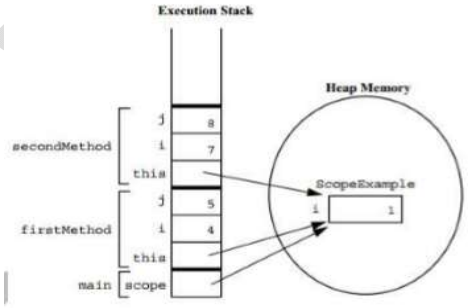

Figure 5-6.1 Variable Scope Example

}

Variable	Value
byte	0
short	0
int	0
long	0L
float	0.0F
double	0.0D
char	'\u0000'
boolean	false
All reference types	null

Figure 5-6.2 Variable Initialization

Initialization Before Use Principle

The compiler will verify that local variables have been initialized before used.

```
public void doComputation() {
int x = (int)(Math.random() * 100);
int y;
int z;
if (x > 50) {
y = 9;
}
z = y + x;  // Possible use before initialization
}
```

javac TestInitBeforeUse.java

```
TestInitBeforeUse.java:10: variable y might not have been initialized
z = y + x; // Possible use before initialization
^
```

1 error

Operators	Associative
++ -- + *unary* - *unary* ~ ! (<data_type>)	R to L
* / %	L to R
+ -	L to R
<< >> >>>	L to R
< > <= >= instanceof	L to R
== !=	L to R
&	L to R
^	L to R
|	L to R
&&	L to R
||	L to R
<boolean_expr> ? <expr1> : <expr2>	R to L
= *= /= %= += -= <<= >>= >>>= &= ^= |=	R to L

Figure 5-6.3 Operator Precedence

Logical Operators

• The boolean operators are:

! – NOT & – AND

| – OR ^ – XOR

• The short-circuit boolean operators are:

&& – AND || – OR

• You can use these operators as follows:

```
MyDate d = reservation.getDepartureDate();
if ( (d != null) && (d.day> 31) {
//do something with d
}
```

Bitwise Logical Operators

• The integer bitwise operators are:

~ – Complement & – AND

^ – XOR | – OR

• Byte-sized examples include:

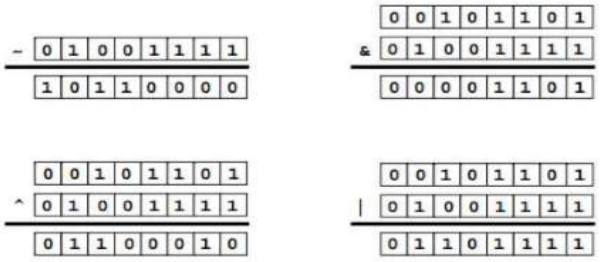

Figure 5-6.4

Right-Shift Operators >> and >>>

Arithmetic or signed right shift (>>) operator:

Examples are:

128 >> 1 returns 128/21 = 64

256 >> 4 returns 256/24 = 16

-256 >> 4 returns -256/24 = -16

- *The sign bit is copied during the shift*
- *Logical or unsigned right-shift (>>>) operator*
- *This operator is used for bit patterns*
- *The sign bit is not copied during the shift*

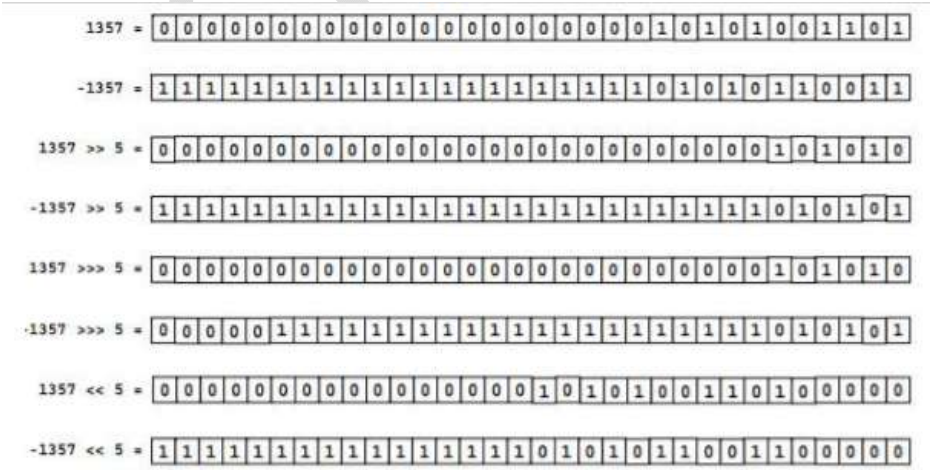

Figure 5-6.5 Shift Operator Examples

String Concatenation With +

The + operator works as follows:

- *Performs String concatenation*
- *Produces a new String:*

```
String salutation = "Dr.";
String name = "Pete" + " " + "Seymour";
String title = salutation + " " + name;
```

- *One argument must be a String object*
- *Non-strings are converted to String objects automatically*

Casting

- *If information might be lost in an assignment, the programmer must confirm the assignment with a cast.*
- *The assignment between long and int requires an explicit cast.*

```
longbigValue = 99L;
int squashed = bigValue; // Wrong, needs a cast
int squashed = (int) bigValue; // OK
int squashed = 99L; // Wrong, needs a cast
int squashed = (int) 99L; // OK, but...
int squashed = 99; // default integer literal
```

Promotion and Casting of Expressions

- *Variables are promoted automatically to a longer form (such as int to long).*
- *Expression is assignment-compatible if the variable typeis at least as large (the same number of bits) as the expression type.*

```
longbigval = 6; // 6 is an int type, OK
intsmallval = 99L; // 99L is a long, illegal
double z = 12.414F; // 12.414F is float, OK
float z1 = 12.414; // 12.414 is double, illegal
```

Simple if, else Statements

The if statement syntax:

```
if (<boolean_expression>)
<statement_or_block>
```

Example:

```
if ( x< 10 )
System.out.println("Are you finished yet?");
```

or (recommended):

```
if ( x < 10 ) {
System.out.println("Are you finished yet?");
}
```


Complex if, else Statements

The if-else statement syntax:

```
if ( <boolean_expression>)
<statement_or_block>
else if ( <boolean_expression>)
<statement_or_block>
```

Example:

```
int count = getCount(); // a method defined in the class
if (count < 0) {
System.out.println("Error: count value is negative.");
} else if (count >getMaxCount()) {
System.out.println("Error: count value is too big.");
} else {
```

```
System.out.println("There will be " + count +
" people for lunch today.");
}
```

Switch Statements

The switch statement syntax

```
switch ( <expression>) {
case <constant1>:
<statement_or_block>*
[break;]
case <constant2>:
<statement_or_block>*
[break;]
default:
<statement_or_block>*
[break;]
}
```

A switch statement example:

```
switch ( carModel ) {
        case DELUXE:
        addAirConditioning();
        addRadio();
        addWheels();
        addEngine();
         break;
        case STANDARD:
        addRadio();
        addWheels();
        addEngine();
        break;
        default:
```

```
        addWheels();
        addEngine();
}
```

This switch statement is equivalent to the previous example:

```
switch ( carModel ) {
case DELUXE:
addAirConditioning();
case STANDARD:
addRadio();
default:
addWheels();
addEngine();
}
```

Without the break statements, the execution falls through each subsequent case clause.

Looping Statements

- **The for loop**

for (<init_expr>; <test_expr>; <alter_expr>)

<statement_or_block>

Example:

```
        for ( inti = 0; i< 10; i++ )
        System.out.println(i + " squared is " + (i*i));
        or (recommended):
        for ( inti = 0; i< 10; i++ ) {
        System.out.println(i + " squared is " + (i*i));
}
```

• **The while loop**

while (<test_expr>)

<statement_or_block>

Example:

```
inti = 0;
while ( i< 10 ) {
System.out.println(i + " squared is " + (i*i));
i++;
}
```

• The do/while loop

do <statement_or_block>
while (<test_expr>);

Example:

```
Int i = 0;
do {
System.out.println(i + " squared is " + (i*i));
 i++;
} while ( i< 10 );
```

Special Loop Flow Control

• The break [<label>]; command

• Thecontinue [<label>]; command

• The <label>: <statement>command, where<statement>should be a loop

The break Statement

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

The continue Statement

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

Using break Statements with Labels

```
outer:
do {
statement1;
```

```
do {
statement2;
if ( condition) {
break outer;
}
statement3;
} while ( test_expr);
statement4;
} while ( test_expr);
```

Using continue Statements with Labels

```
 test:
do {
statement1;
do {
statement2;
if ( condition) {
continue test;
}
statement3;
} while ( test_expr);
statement4;
} while ( test_expr);
```

Read Additional Resources

1. **Using NetBeans™ to Compile and Run Java Programs (PDF)**
   https://users.drew.edu/bburd/BeginProg2/NetBeans.pdf
2. **Java Loop Control (PDF)**
   https://www.tutorialspoint.com/java/pdf/java_loop_control.pdf

Watch Video Resources

1. **Conditional Logic (If / Else) in Java**
   https://youtu.be/UAZ5A7gG5vk
2. **Loops in Java (For / While)**
   https://youtu.be/vdtowtOTGng

Insights

Insights 5-6.1

Instructions: *Write your answer on the Insights Sheet (IS) provided in this module.*

We wanted to know how deep your understandings about the topics presented in the module. Your answer will be graded accordingly using the prescribed rubrics. In your own words (at least 200 words):

1. Write down the importance of Loops in the Java Programming Language. *(10-points)*

2. Is it beneficial and powerful to implement conditional logic in the aspect of programming? Why? Justify your answer. *(10-points)*

Laboratory Activity

Activity 5-6.1

1. Creating a pyramid using **for loop**

Filename: **pyramid.java**

```
/*
 * To change this license header, choose License Headers in Project Prope
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package pyramid;


public class Pyramid {


    public static void main(String[] args) {
      int rows = 10, coef = 1;
    for(int i = 1; i <= rows; i++) {
        for(int space = 1; space < rows - i; ++space) {
            System.out.print(" ");
        }
        for(int j = 0; j <= i - 1; j++){
            System.out.printf("%4d", i);
        }
        System.out.println();
      }
      System.out.println("Lastname, Firstname");

    }

}
```

Activity 5-6.2

Machine Problem: Write a program that will assign given data to variables. Compute and display using the format shown below. Save the program as salary.java, compile and execute the program.

Given:

Gross Pay = Day Worked * Daily Rate

Net Pay= Gros sPay – (Tax + SSS + Philhealth + Pagibig)

Output Format:

```
Employee Name: Juan Dela Cruz
Employee ID: 201701
Employment Status: Permanent
Department: Finance
Days Worked: 22 Days
Daily Rate: 463.00
Gross Pay: xxxxx

Less Tax: 270.00
SSS: 500.00
Philhealth: 100.00
Pagibig: 50.00

Net Pay: xxxx.xx
```

Internet References

1. https://www.youtube.com
2. https://www.youtube.com/c/devfactor/about
3. https://users.drew.edu
4. https://www.tutorialspoint.com
5. Phoenix Publishing House Inc.