Lab 1

Declaring, Initializing, and Using Variables

Objectives

Upon completion of this lab, you should be able to:

- Complete review questions
- Perform variable declarations in a class
- Perform type casting and use operators in a Java technology program

Lab Overview

In this lab, you complete review questions and two exercises.

- In the first exercise, you create a Java class containing variable declarations.
- In the second exercise, you write a Java technology program in which you use operators and perform type casting.

Completing Review Questions

Complete the following questions:

1. State whether the following statements are true or false:
   a. There are eight primitive types built into the Java programming language.
   b. In the Java programming language, the variable names `intgrade` and `intGrade` are the same.
   c. byte, short, int and long are the four integral primitive types in the Java programming language.
   d. A boolean type variable holds true, false and nil.
2. Which answer best describes the following variable declaration:
   `final double PI = 3.14159;`
   a. This is the value assignment for an integral primitive type.
   b. The name of the variable is a Java programming language keyword.
   c. The variable is a constant.
   d. The variable is incorrectly named, according to the naming guidelines.
3. Which of the following operations correctly illustrate operator precedence in the following calculation
   `X = 1 + 2 * 5 -8 +6 / 7 * 4;`

   ```
   a.  (1 + 2) * (5 - 8) + (6 / (7 * 4))
   b.  1 + (2 * 5) - 8 + ((6 / 7) * 4)
   c.  (1 + 2) * (5 - ((8 + 6) / (7 * 4)))
   d.  1 + (2 * (5 - 8)) + ((6 / 7) * 4)
   ```

4. Examine the following code, and select the correct output after execution:

```
int a = 5;
int b = ++a;
System.out.println("a = "+ a + "and b = "+ b);
```

a. `a = 5` and `b = 5`

b. `a = 5` and `b = 6`

c. `a = 6` and `b = 6`

d. `a = 6` and `b = 5`

Exercise 1: Using Primitive Type Variables in a Program

In this exercise, you create a class containing variable declarations.

Preparation

Ensure that `CustomerTest.java` files exists in the `SL110/exercises/04_variables/exercise1` directory.

Task – Creating a Class With Variables

In this task, you compile and execute a `Customer` class with several variables.

Complete the following steps to create your `Customer` class:.

1. Go to the `SL110/exercises/04_variables/exercise1` directory.
2. Open a text editor, and create a file in this directory called `Customer.java`.
3. In the `Customer.java` file, write a class called `Customer` that creates and assigns default values to the variables for the following information (use constants if appropriate):
   - A customer ID
   - The customer status of 'N' for new or 'O' for old
   - Total purchases for the year
4. Create a method within the class called `displayCustomerInfo` that uses the `System.out.println` method to print each value to the screen with a corresponding label (such as `"Purchases are: "`).
5. Compile and run the program using the provided `CustomerTest` class file.

Note – All of the examples and exercises in this course require a test class. In most situations, the test class is provided. However, in some situations, you will create the class.

6. Check the output to be sure that it contains the values you assigned

Exercise 2: Using Operators and Type Casting

The objective of this exercise is to give you practice using operators and type casting.

This exercise has three tasks. In each of the tasks you create one Java technology program, compile it, and test it.

This exercise consists of the following tasks:

- "Task 1 – Calculating Age Using Operators"
- "Task 2 – Using Casting to Prevent Data Loss"
- "Task 3 – Creating a Temperature Program"

Preparation

Ensure that the `PersonTest.java`, `OrderTest.java`, and `TemperatureTest.java` files exists in the `SL110/exercises/04_variables/exercise2` directory.

Task 1 – Calculating Age Using Operators

In this task, you use operators to calculate age in minutes and milliseconds.

Follow these steps to create a `Person` class that calculates various ages:

1. Open a terminal window and go to the `SL110/exercises/04_variables/exercise` directory.
2. Open a text editor and create a `Person` class in a file.
3. Add the following code to your `Person` class:
   a. Create member variables to store age in years, days, minutes, seconds, and milliseconds.
   b. Set an initial value of 1 to the variable which stores the age in years.

Note – Ensure that you assign the correct data types for each variable and provide meaningful names to all the variables.

c. Create a `calculateAge` method in the `Person` class.

d. In the method, calculate the age in days, minutes, seconds, and milliseconds and assign the new values to the corresponding member variables.

e. In the method, print out all the ages in various units, each in a separate line with an appropriate message. For example: "You are 31536000 seconds old".

f. Save and close the `Person.java` file.

g. Compile the `Person.java` and `PersonTest.java` files. Remove the compilation errors if any and recompile.

h. Test the program using the `PersonTest` class.

i. Perform tests, by setting the value of age as 1, 24, and 80 in the `Person` class.

Note – For one year, the results should be: You are 365 days old. You are 31536000 seconds old. You are 525600 minutes old. You are 31536000000 milliseconds old. For 24 years, the results should be: You are 8760 days old. You are 756864000 seconds old. You are 12614400 minutes old. You are 756864000000 milliseconds old.

Task 2 – Using Casting to Prevent Data Loss

In this task, you use casting to ensure that data loss does not occur in your programs.

Perform the following steps:

1. Create a class called Order that contains three member variables as follows:
   a. long orderValue=0
   b. int itemQuantity=10000000
   c. int itemPrice=555500
2. Create a `calculateTotal` method that will calculate the total order value and print it. Ensure that you type cast the result of the multiplication before storing it in order value.

3. Save and close the `Order.java` file.
4. Compile `Order.java` and `OrderTest.java` file.
5. Test the program by running `OrderTest.class` file. Verify the result of `orderValue` by performing the calculation in a calculator.
6. Open `Order.java` and remove and type casting done in the `calculateTotal` method. Save and close the file.
7. Recompile `Order.java` file and run the `OrderTest` class again.
8. Compare the current result of `orderValue` with the one obtained in Step 5.
9. Test the program by replacing the value of the integer member variables of `Order` class by:
   a. One-digit `int` types
   b. Five-digit `int` types
   c. Nine-digit `int` types
   Ensure that you get the same result with the program as you do when doing each calculation using a calculator.

Task 3 – Creating a Temperature Program

In this task, you write a program to convert temperature from fahrenheit to Celsius.

Write code to complete the following steps:

1. Create a `Temperature` class with a member variable which stores the temperature in fahrenheit. Declare the variable with an appropriate data type, such as int, float, or double.
2. Create a `calculateCelsius` method.
3. In the `calculateCelsius` method, use the following information to convert the temperature value from fahrenheit to Celsius.
   - To convert from Fahrenheit to Celsius, subtract 32, multiply by 5, and divide by 9.
4. Test the program using the `TemperatureTest` class.
5. Ensure that you get the same result with the program as you do when doing the calculation using a calculator.
6. Test the program using several values of temperature.

Exercise Summary

Take a few minutes to identify what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications