Practices for Lesson 4

Practices Overview

In these practices, you create several Java classes that declare, initialize and manipulate field variables. Solutions for these practices can be found in `Lesson04`.

Practice 4-1: Declaring Field Variables in a Class

Overview

In this practice, you create a class containing several fields. You declare the fields, initialize them, and then test the class by running the CustomerTest program.

Assumptions

This practice assumes that the CustomerTest Java source file appears in the practice folder for this lesson: `Lesson04`

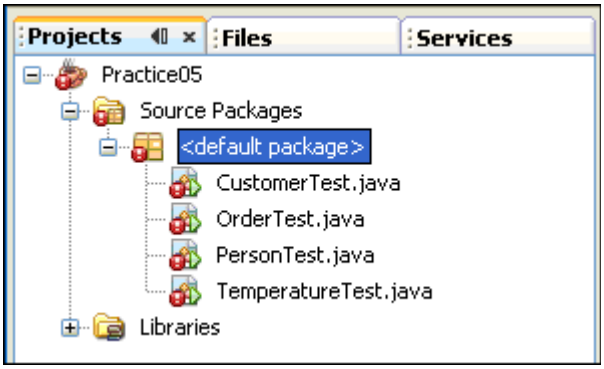
Tasks

1. Close any open project in NetBeans. In the Projects window, right-click the project name and select **Close** from the context menu.
2. Create a new project from existing Java source, using the values in the table below when you complete the New Project wizard.

Step	Window/Page Description	Choices or Values
a.	Choose Project step	Category: Java Project: Java Project with Existing Sources
b.	Name and Location step	Project Name: Practice05
c.	Existing Sources step	**Add Folder**: `Lesson4`
d.	Project Properties window	Set the **Source/Binary Format** property to **JDK 7**

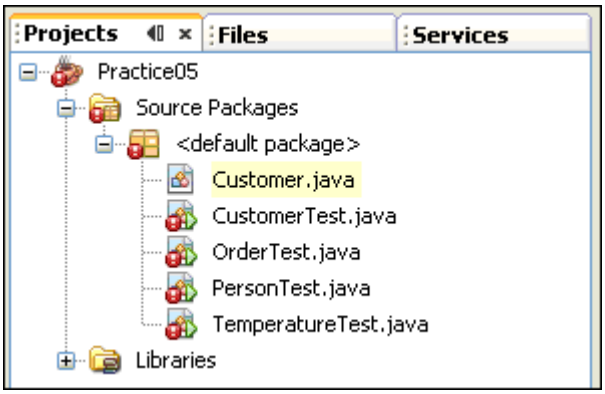
Note: If you need a more detailed reminder of how to create a new project, refer to Practice 1-2, steps 3 and 4.

Solution: The Projects window should show four Java source files beneath the <default package> node.



3. Create a new Java class. The table below provides the high level steps. If you need more assistance, refer to Practice 2-2, step 1.

Step	Window/Page Description	Choices or Values
a.	Menu	**File > New File**
b.	New File window | Choose File Type step	Category: Java File Types: Java class **Next**
c.	New Java Class window | Name and Location step	Class Name: Customer **Finish**



4. With Customer.java open for editing in the Editor pane, declare and initialize the fields described in the table below. If you need more assistance, more detailed steps are provided following the table.

Field Name	Data Type	Default Value
`customerID`	`int`	<your choice>
`status`	`char`	<your choice> 'N' for new, 'O' for old
`totalPurchases`	`double`	0.0

a. The syntax of a variable declaration and initialization is:

```
modifier type variable = <value>;
```

b. Assume that all fields are `public`.

c. Include a comment at the end of each line describing the field.

Solution: This shows one possible solution for the `customerID` declaration and initialization. The others are similar.

```
public int customerID = 0;   // Default ID for a customer
```

5. Add a method within the Customer class called `displayCustomerInfo`. This method uses the `System.out.println` method to print each field to the screen with a corresponding label (such as "Purchases are: ").

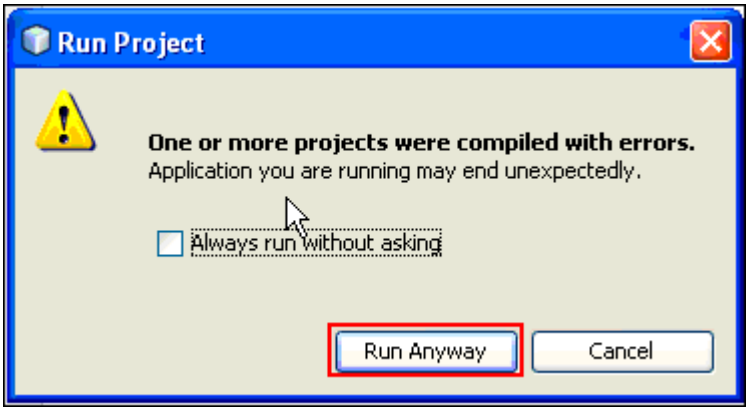
Solution:

```
public void displayCustomerInfo () {
      System.out.println("Customer ID: " + customerID);
            // continue in a similar fashion for all other fields
}
```

6. Click Save to compile the class.

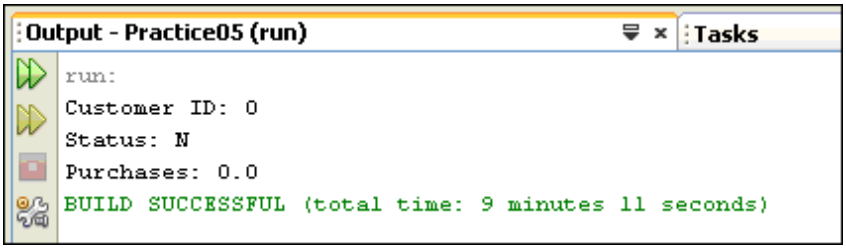
   Note: You will notice that the red error indicator next to the CustomerTest class in the Projects window disappears after saving the Customer class. The reason is that the CustomerTest class references the `displayCustomerInfo` method, which did not exist before you saved the file. NetBeans recognized a potential compilation error in the CustomerTest class, due to the missing method.

7. Run the CustomerTest class to test your code. If you are prompted with a warning indicating that there are compilation errors within the project, click **Run Anyway**.



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

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



Practice 4-2: Using Operators and Performing Type Casting to Prevent Data Loss

Overview

In this practice, you use operators and type casting. This exercise has three sections. In each section you create one Java class, compile it, and test it.

Assumptions

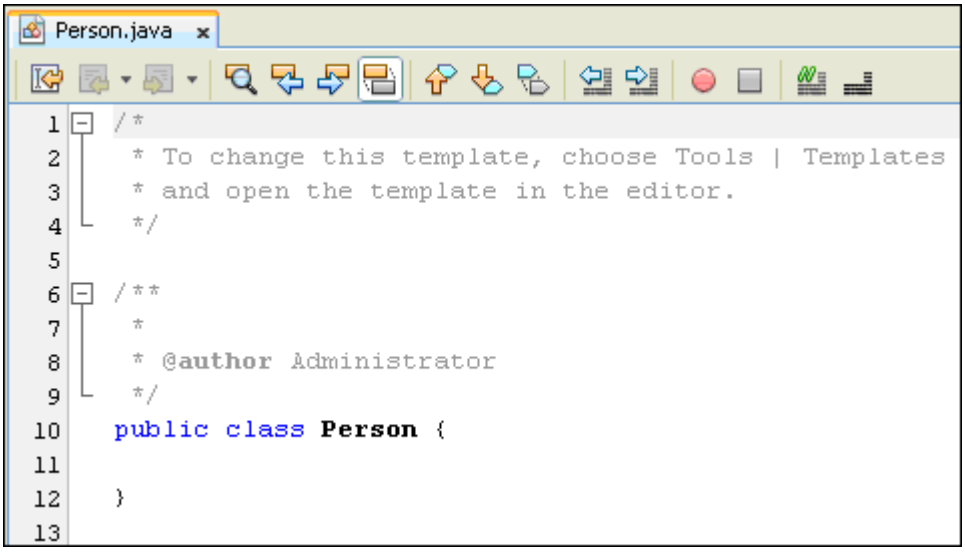
The following Java source files appear in the practice folder for this lesson: `Lesson4`

- PersonTest.java
- OrderTest.java
- TemperatureTest.java

Calculating Age Using Operators

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

1. Select File > New File from the menu to create a new Java class called Person.



2. Using the editor, add the following fields to store age in years, days, minutes, seconds, and milliseconds. Provide meaningful names for all the fields. The table below provides more detailed information:

Year Part	Data Type	Additional Info
Years	`int`	Initialize to 1
Days	`int`	Do not initialize
Minutes	`long`	Do not initialize
Seconds	`long`	Do not initialize
Milliseconds	`long`	Do not initialize

Hint: You can declare multiple variables of the same type in one line by separating the variables by a comma. Be sure to end the line with a semicolon, just as you would any other line of code.

3. Create a new public method in this class called `calculateAge`.
   a. The method should calculate age in days, minutes, seconds, and milliseconds, assigning the value to the relevant field. The following table gives you the calculations:

Year Part	Calculated By:
Days	Year * 365
Seconds	Days * 24 * 60 * 60
Minutes	Seconds / 60
Milliseconds	Seconds * 1000

   b. Print out all the ages in various units, each in a separate line with an appropriate message. For example “You are 3156000 seconds old.”

   Solution:

```
public void calculateAge () {
      ageDays = ageYears * 365;
      ageSeconds = ageDays * 24 * 60 * 60;
      ageMinutes = ageSeconds / 60;
      ageMilliseconds = ageSeconds * 1000;

      System.out.println ("You are " + ageDays + " days old.");
      System.out.println("You are " + ageMinutes +
        " minutes old.");
      System.out.println("You are " + ageSeconds +
        " seconds old.");
      System.out.println("You are " + ageMilliseconds +
        " milliseconds old.");
}
```

4. Save to compile the class and then run the `PersonTest.java` file.
5. Perform several tests, by setting the value of age as 1, 24, and 80 in the `Person` class.

   Solution:

   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.

Using Casting to Prevent Data Loss

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

6. Create a new Java class called `Order`

7. Add three fields to the `Order` class as follows:

Field Name	Data Type	Initialized Value
`orderValue`	`long`	0L (zero L)
`itemQuantity`	`int`	10_000_000
`itemPrice`	`int`	555_500

Note: The underscores used to initialize the `int` values improve the readability of your code. They have no effect on the actual numeric value of the field. The compiler strips them out. This is one of the new language features of Java 7.

8. Create a `calculateTotal` method that calculates the total order value (`itemQuantity` * `itemPrice`) and print it. Be sure to type cast either `itemQuantity` or `itemPrice` to a `long` so that the temp storage used to hold the outcome of the multiplication is large enough to contain a `long` value.

Solution:

```
public void calculateTotal(){
     orderValue = (long)itemQuantity * itemPrice;
     System.out.println("Order total: "+ orderValue);
}
```

9. Save `Order.java` and then test it by running `OrderTest.java`. Verify the result by using a calculator.

Solution: Result should be 5555000000000

10. Edit the `Order.java` file to remove the type casting done in the `calculateTotal` method.
11. Compile and run OrderTest again to see the resulting data loss that occurs without type casting.

Creating a Temperature Program

In this section, you write a program to convert temperature from Fahrenheit to Celsius.

12. Create a new Java class called Temperature. Add a member field to the Temperature class that stores the temperature in Fahrenheit. Declare the field variable with an appropriate data type, such as `int`, `float`, or `double`.
13. Create a `calculateCelsius` method. Convert the Fahrenheit temperature to Celsius by subtracting 32, multiplying by 5, and dividing by 9. Be sure to observe the rules of precedence when typing this expression.

Hint: The rules of precedence are listed here for your convenience.

- Operators within a pair of parentheses
- Increment and decrement operators
- Multiplication and division operators, evaluated left to right
- Addition and subtraction operators, evaluated left to right

Solution: This is one possible solution.

```
public class Temperature {
     public float fahrenheitTemp = 78.9F;

     public void calculateCelsius() {
          System.out.println ((fahrenheitTemp – 32) * 5 / 9);
     }
}
```

14. Compile the Temperature class and test it using the TemperatureTest class. Confirm that you get the same result running the program as you do when doing this calculation using a calculator.
15. Test the program using several values of temperature.
16. When you have finished experimenting with different values, close the Practice05 project.