Practices for Lesson 2: Java Syntax and Class Review

Chapter 2

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practices for Lesson 2: Overview

Practices Overview

In these practices, you will use the NetBeans IDE and create a project, create packages, and a Java main class, and then add classes. You will also run your project from within the IDE and learn how to pass command-line arguments to your main class.

Note: There are two levels of practice for most of the practices in this course. Practices that are marked “Detailed Level” provide more instructions and, as the name implies, at a more detailed level. Practices that are marked “Summary Level” provide less detail, and likely will require additional review of the student guide materials to complete. The end state of the “Detailed” and “Summary” level practices is the same, so you can also use the solution end state as a tool to guide your experience.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 2-1: Summary Level: Creating Java Classes

Overview

In this practice, using the NetBeans IDE, you will create an `Employee` class, create a class with a `main` method to test the `Employee` class, compile and run your application, and print the results to the command line output.

Tasks

1. Start the NetBeans IDE by using the icon from Desktop.
2. Create a new project `Employee` in the /`home/oracle/labs/02-Review/practices/practice1` directory with an `EmployeeTest` main class in the `com.example` package.
3. Set the Source/Binary format to JDK 8.
   a. Right-click the project and select Properties.
   b. Select JDK 8 from the drop-down list for Source/Binary Format.
   c. Click OK.
4. Create another package called `com.example.domain`.
5. Add a Java Class called `Employee` in the `com.example.domain` package.
6. Code the `Employee` class.
   a. Add the following data fields to the `Employee` class—use your judgment as to what you want to call these fields in the class. Refer to the lesson materials for ideas on the field names and the syntax if you are not sure. Use `public` as the access modifier.

Field use	Recommended field type
Employee id	`int`
Employee name	`String`
Employee Social Security Number	`String`
Employee salary	`double`

   b. Create a `no-arg` constructor for the `Employee` class.
   c. Add accessor/mutator methods for each of the fields.

   Note that NetBeans has a feature to create the getter and setter methods for you. Click in your class where you want the methods to go, then right-click and choose Insert Code (or press the Alt-Insert keys). Choose getters and setters from the Generate menu, and click the boxes next to the fields for which you want getter and setter methods generated.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

7. Write code in the `EmployeeTest` class to test your `Employee` class.
   a. Construct an instance of `Employee`.
   b. Use the setter methods to assign the following values to the instance:

Field	Value
Employee id	101
Employee name	Jane Smith
Employee Social Security Number	012-34-5678
Employee salary	120_345.27

   c. In the body of the main method, use the `System.out.println` method to write the value of the employee fields to the console output.
   d. Resolve any missing import statements.
   e. Save the `EmployeeTest` class.
8. Run the `Employee` project.
9. (Optional) Add some additional employee instances to your test class.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 2-1: Detailed Level: Creating Java Classes

Overview

In this practice, using the NetBeans IDE, you will create an `Employee` class, create a class with a `main` method to test the `Employee` class, compile and run your application, and print the results to the command-line output.

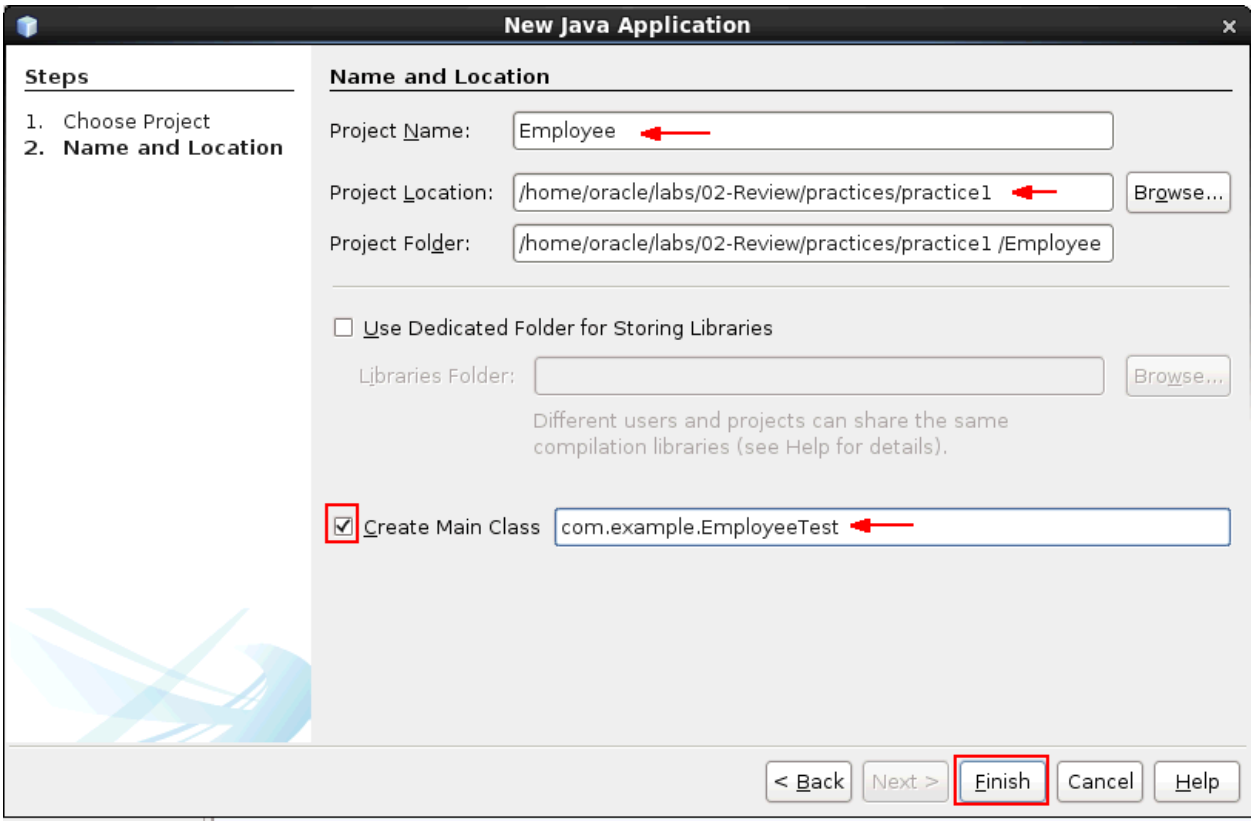
Tasks

1. Start the NetBeans IDE by using the icon from Desktop.
2. Create a new Project called `Employee` in NetBeans with an `EmployeeTest` class and a `main` method.
   a. Click File > New Project.
   b. Select Java from Categories, and Java Application from Projects.
   c. Click Next.
   d. On the New Application window, perform the following steps:

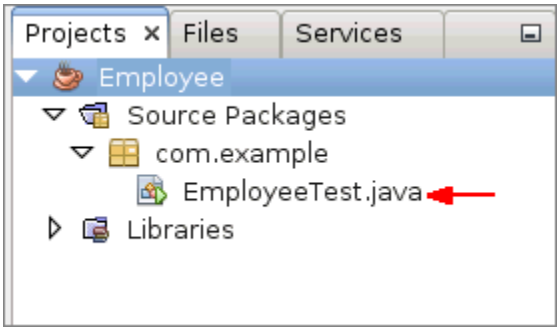
Window/Page Description	Choices or Values
Project Name:	`Employee`
Project Location	`/home/oracle/labs/02-Review/practices/practice1`
Use Dedicated Folder for Storing Libraries	Ensure this is **not** selected.
Create Main Class	Ensure this is selected. Change the name to `com.example.EmployeeTest` `com.example` is the package name.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

e. Click Finish.



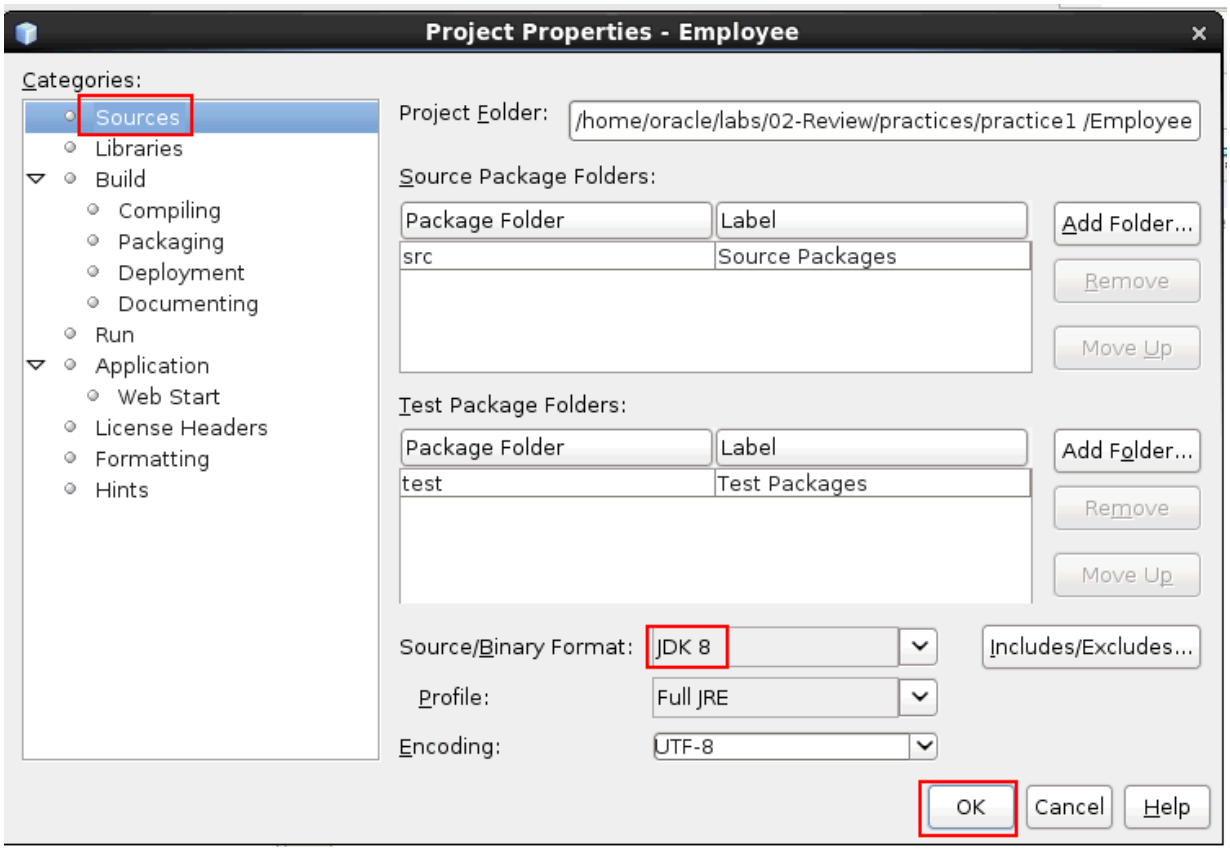
f. In the Projects tab, expand the `Employee` project, you will notice that NetBeans has created a class called `EmployeeTest`, including the package name of `com.example`, and skeleton of the `main` method is generated.



3. Set the Source/Binary format to JDK 8.
   a. Right-click the `Employee` project and select Properties.
   b. In the Project Properties window perform the following steps:
      1) Select JDK 8 from the drop-down list for Source/Binary Format.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

2) Click OK.



4. Create another package called `com.example.domain`.
   a. Right-click the current package `com.example` under `Source Packages`.
   b. Select **New > Java Package**.
   c. In the New Java Package window, perform the following steps:
      1) Enter `com.example.domain` in the Package Name field.
      2) Click Finish.

   You will notice that the icon beside the package name is gray in the Project—this is because the package has no classes in it yet.
5. Create a new Java Class called `Employee` in the `com.example.domain` package.
   a. Right-click the `com.example.domain` package and select **New > Java Class**.
   b. In the Class Name field, enter `Employee`.
   c. Click Finish to create the class.

   Notice that NetBeans has generated a class with the name `Employee` in the package `com.example.domain`.

   Note: You can format your code in NetNeans: right-click anywhere in the class and select Format, or press the Alt-Shift-F key combination.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

6. Code the `Employee` class.
   a. Add the following data fields to the `Employee` class.

Field use	Access	Recommended field type	Field name
Employee id	`public`	`int`	`empId`
Employee name	`public`	`String`	`name`
Employee Social Security Number	`public`	`String`	`ssn`
Employee salary	`public`	`double`	`salary`

   b. Add a constructor to the `Employee` class:
   ```
   public Employee() { }
   ```
   c. Create accesor/mutator (getter/setter) methods for each of the fields.
   Note that NetBeans has a feature to create the getter and setter methods for you.
      1) Click in your class where you want the methods to go, then right-click and choose Insert Code (or press the Alt-Insert keys).
      2) Select “Getter and Setter” from the Generate menu.
      3) Click the boxes next to the fields for which you want getter and setter methods generated. You can also click the class name (Employee) to select all fields.
      4) Click Generate to insert the code.
   d. Save your class.
7. Modify the `EmployeeTest` main class to test your `Employee` class:
   a. Add an import statement to your class for the `Employee` object:
   ```
   import com.example.domain.Employee;
   ```
   b. In the `main` method of `EmployeeTest`, create an instance of your `Employee` class:
   ```
   Employee emp = new Employee();
   ```
   c. Using the employee object instance, add data to the object using the setter methods. For example:

```
emp.setEmpId(101);
emp.setName("Jane Smith");
emp.setSsn ("012-34-5678");
emp.setSalary(120_345.27);
```

   Note that after you type the "`emp.`", Netbeans provides you with suggested field names (in green) and method names (in black) that can be accessed via the `emp` reference you typed. You can use this feature to cut down on typing. After typing the dot following `emp`, use the arrow keys or the mouse to select the appropriate method from the list. To narrow the list down, continue typing some of the first letters of the method name. For example, typing `set` will limit the list to the method names that begin with `set`. Double-click the method to choose it.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

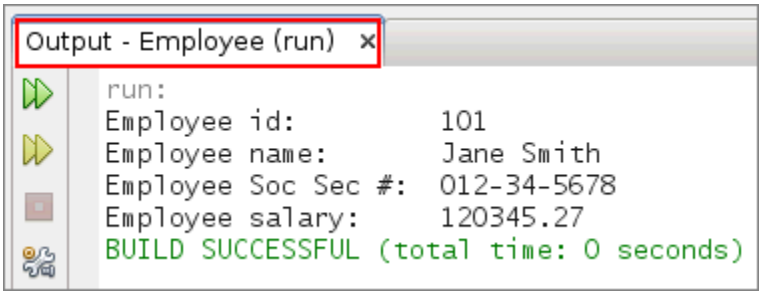
d. In the body of the main method, use the `System.out.println` method to write messages to the console output.

```
System.out.println ("Employee id:         " + emp.getEmpId());
System.out.println ("Employee name:       " + emp.getName());
System.out.println ("Employee Soc Sec #: " + emp.getSsn());
System.out.println ("Employee salary:     " + emp.getSalary());
```

The `System` class is in the `java.lang` package, which is why you do not have to import it (by default, you always get `java.lang`). You will learn more about how this multiple dot notation works, but for now understand that this method takes a string argument and writes that string to the console output.

e. Save the `EmployeeTest` class.

8. Examine the Project Properties.
   a. Right-click the project and select Properties.
   b. In the Project Properties window, perform the below steps:
      1) Expand Build, if necessary, and select Compiling. The option at the top, "Compile on Save," is selected by default. This means that as soon as you saved the Employee and `EmployeeTest` classes, they were compiled.
      2) Select **Run**. You will see that the Main Class is `com.example.EmployeeTest`. This is the class the Java interpreter will execute. The next field is Arguments, which is used for passing arguments to the main method. You will use arguments in a future lesson.
      3) Click Cancel to close the Project Properties.
9. Run the `Employee` project.
   a. To run your `Employee` project, right-click the project and select Run. If your classes have no errors, your should see the following output in the Output window:



10. (Optional) Add some additional employee instances to your test class.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.