3

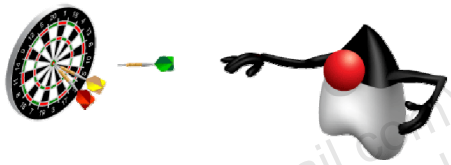
Creating a Java Main Class

ORACLE

Objectives

After completing this lesson, you should be able to:

- Create a Java class
- Write a main method
- Use `System.out.println` to write a String literal to system output

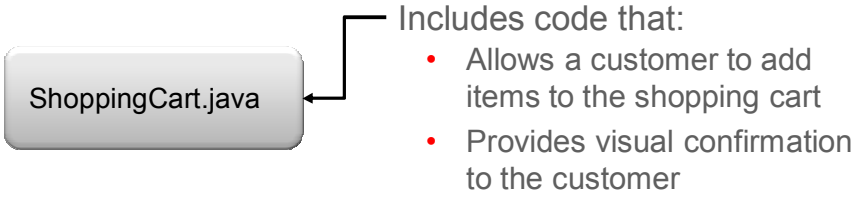


Topics

- Java classes and packages
- The `main` method

Java Classes

A Java class is the building block of a Java application.



Program Structure

- A class consists of:
  - The class name. Class names begin with a capital letter.
  - The body of the class surrounded with braces { }
    - Data (called fields)
    - Operations (called methods)
- Example:

Java is case-sensitive!

```
public class Hello {
   // fields of the class
   // methods
}
```


- A class is declared using the keyword, `class`, followed by the class name.
- Convention dictates that the class name start with a capital letter. If there are two words in the class name (SayHello), each word should begin with a capital letter. In the example above, the class name is Hello.
- The keyword `public` is called a *modifier*. You learn about these in the lesson titled "Using Encapsulation."
- **Java is case-sensitive**. It does not recognize the following two words as being the same thing: `class` and `Class`.
- A class would typically contain data (called fields) and operations (called methods). You learn about this a little later.
- Notice that the body of the `Hello` class is enclosed in braces (`{ }`).

Java Packages

- A package provides a namespace for the class.
  - This is a folder in which the class will be saved.
  - The folder name (the package) is used to uniquely identify the class.
  - Package names begin with a lowercase letter.
- Example:

Package name

```
package greeting;

public class Hello {
    // fields and methods here
}
```

The class's unique name is: greeting.Hello

The use of a package when you create a Java class is not mandatory, but it is strongly recommended. Notice the semicolon after `package greeting;`

Semicolons are required at the end of each statement. It is similar to the period at the end of a sentence. The sentence may wrap to another line, but it is not complete until the period. The Java compiler interprets a statement as being complete when it encounters the semicolon.

Using the Java Code Console

For the exercises in this course, you use a browser-based Java IDE.

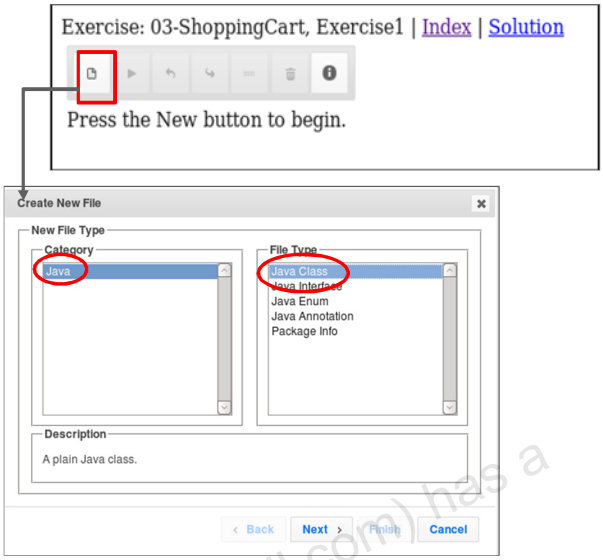
1. Open a browser and enter: http://localhost:8080/JavaCC
2. Click the Lessons link.
3. Click the exercise number for the current lesson.

IDE stands for *Integrated Development Environment*. This is a type of software that makes it easier to develop applications. In the practices for the lesson titled “Describing Objects and Classes” and beyond, you use the NetBeans IDE, which is a full-featured Java IDE.

For the short exercises that are sprinkled throughout the lecture portions of this course, you will use a simple web-based IDE called Java Code Console. It was written specifically for this purpose. This IDE is hosted on your student machine.

Using the Java Code Console: Creating a New Java Class

1. Click the New button to create a new file.
2. Select the Java category and Java Class file type.
3. Click Next.



Most of the exercises provide a pre-existing Java class or classes, to which you add code. If the exercise you selected has a pre-existing `.java` file for you, it will appear in a tabbed view on this page.
In some cases, however, you need to create a new Java class. The steps are described here and in the next slide.

- Click the New button as shown in the top screenshot.
- The Create New File Wizard opens. Select Java in the Category column. Select Java Class in the File Type column.
- Click Next.
- Instructions are also displayed on the Java Code Console page, just below the code editor panel. However, if no Java file is provided and you are going to create a new one, the code editor is not shown. Therefore, the instructions appear just below the toolbar.

Using the Java Code Console: Creating a New Java Class for an Exercise

4. Enter a class name and a package name.
5. Click Finish.

Enter a class name and package name, as specified in the exercise instructions. When you click Finish, the code for this class appears in a tabbed page, bearing the class name.

- In the left screenshot, you see the `ShoppingCart.java` tab, containing the package statement and the class declaration. This is the code editor panel.
- You will write your code between the braces.
- To test your code, click the Run button as indicated above.
- To return to the list of exercises, click the Index link.
- To view the solution for the exercise, click the Solution link. It will replace the current `.java` file with the solution `.java` file in the same tab, but you will be able to toggle between your code and the solution.

Exercise 3-1: Creating a Class

In this exercise, you use the Java Code Console to create a new Java Class.

- Click New to create a new class:
  - Class name = `ShoppingCart`
  - Package name = `exercise`
- Leave the tabbed view open in the browser because you will modify the code in the next exercise.

The fully-qualified class name should be `exercise.ShoppingCart`. Do not click the Solution link until after exercise 3-2.

Topics

- Java classes
- The `main` method

The `main` Method

- It is a special method that the JVM recognizes as the starting point for every Java program.
- The syntax is always the same:

```
public static void main (String args[]) {
   // code goes here in the code block
}
```

- It surrounds entire method body with braces { } .

- The main method is a special method that the Java Virtual Machine recognizes as the starting point for a Java program.
- Any program that you want to run must have a public `main` method.
- A class containing a main method is referred to as a “main class.”

Note: Brackets (`[]`) can be placed to the right of `String` or to the right of `args`, but the former is recommended:

```
(String[] args)
(String args[])
```

A `main` Class Example

Class name

```
public class Hello {

   public static void main (String[] args) {
      // Entry point to the program.
      // Write code here:
      System.out.println ("Hello World!");
   }
}
```

Comments

main method

Program output

Here you see a simple example of a class (Hello) that includes a `main` method. The `main` method writes a message to the console (“Hello World!”). This is called *program output*.

You can include comments that the compiler will ignore, by preceding the comment line with two forward slashes: //

Output to the Console

- Syntax:

```
System.out.println (<some string value>);
```

- Example:

String literal

```
System.out.println ("This is my message.");
```


Use the `System.out.println` method to print a message to the console. Use double quotation marks to enclose the text of the message (called a String literal).

Fixing Syntax Errors

- If you have made a syntax error, the error message appears in the Output panel.
- Common errors:
  - Unrecognized word (check for case-sensitivity error)
  - Missing semicolon
  - Missing close quotation mark
  - Unmatched brace

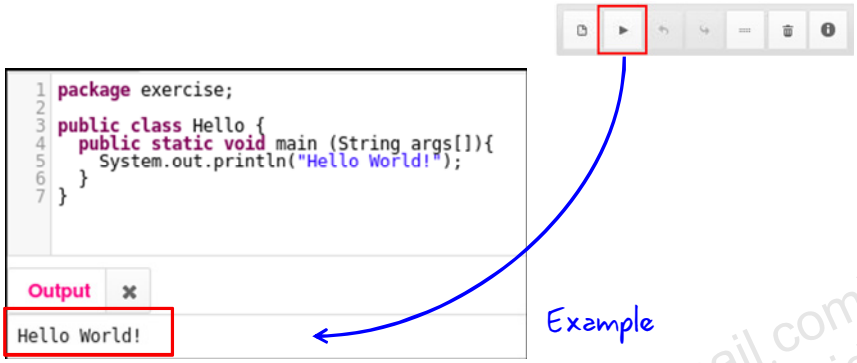
The Java Code Console helps you out by using color coding. Keywords (such as class) are in purple. String literals are shown in blue. When you click Run, it will attempt to compile your code and then run it. Syntax errors will be caught during compilation. The error message may or may not be helpful in troubleshooting, so here are some common errors to avoid:

- Unrecognized class (It may say “Cannot find symbol.”). This happens when you misspell a method or class name, or if you get the case wrong. Remember, Java is case-sensitive!
- Every Java statement must end with a semicolon. This example error is shown above in line 5.
- An uneven number of symbols such as braces, brackets, and quotation marks will also cause errors.

Exercise 3-2: Creating a `main` Method

In this practice, you manually enter a `main` method that prints a message to the console.

- In the code editor, add the `main` method structure to the ShoppingCart class.
- In the code block, use a `System.out.println` method to print "Welcome to the Shopping Cart!"
- Click the **Run** button to test program.



In the example above, you see a class called Hello. Within the braces that define the scope of the class, you see the `main` method. The method body is also surrounded by braces.

When you click the Run button as indicated in the image on the right, the program is both compiled (using `javac`) and executed (using `java`).

The `println` statement results in the "Hello World!" string appearing in the Output tab.

Quiz

Which `main` method syntax is correct?

a. `Public static void main (String[ ] args){ }`

b. `public Static void Main (String[ ] args){ }`

c. `public static void main (String ( ) args)[ ]`

d. `public static void main (String[ ] args){ }`

Answer: d

- a is incorrect. It should be “public”, not “Public”.
- b is incorrect. Both “Static” and “Main” should begin with a lowercase letter.
- c is incorrect because there should be brackets following “String” and braces defining the method scope.
- d is correct.

Summary

In this lesson, you should have learned how to:

- Create a class using the Java Code Console
- Create (declare) a Java class
- Define a main method within a class
- Use `System.out.println` to write to the program output
- Run a program in the Java Code Console