Creating a Java Main Class

Objectives

After completing this lesson, you should be able to:

- Use the NetBeans IDE to create and test Java classes
- 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.

Includes code that:

- Allows a customer to add items to the shopping cart
- Provides visual confirmation to the customer

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`

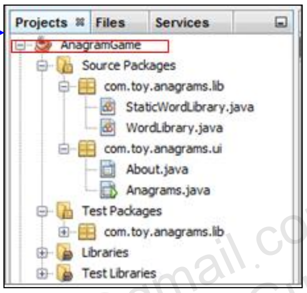
The use of a package when you create a Java class is not mandatory, but it is strongly recommended. Notice the semicolon after `package greeting;`

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.

Java IDEs

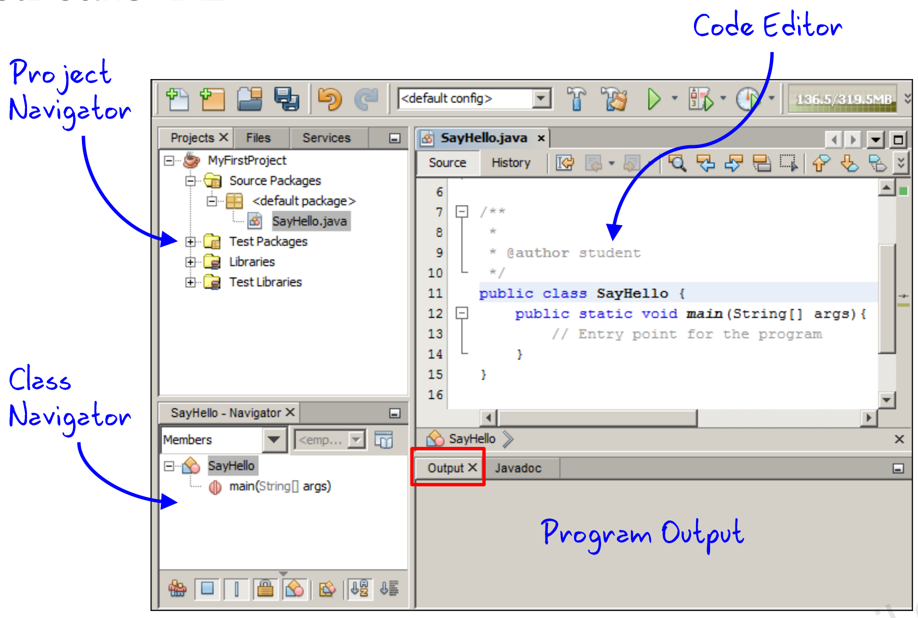
A Java Integrated Development Environment (IDE) is a type of software that makes it easier to develop Java applications.

- An IDE provides:
  - Syntax checking
  - Various automation features
  - Runtime environment for testing
- It enables you to organize all your Java resources and environment settings into a *Project*.
- Projects contain packages.
- Packages contain files, such as `.java`.



Some well-known Java IDEs are NetBeans (used in this class to perform the practices and exercises), Eclipse, and JDeveloper.

The NetBeans IDE



```
/**
 *
 * @author student
 */
public class SayHello {
    public static void main(String[] args){
        // Entry point for the program
    }
}
```

The Java project provides a mechanism by which you can organize all of the source and class files and other resources (connection profiles, configuration information, and so on) required by the Java application.

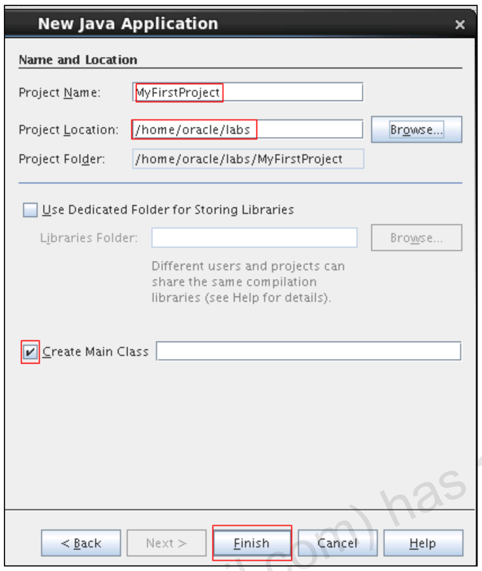
- When you begin working in NetBeans, you either create a project or open an existing one.
- The Project Navigator gives you a visual representation of the project contents.
- You can open files from your project in the code editor by double-clicking the file or using the context menu.

When you select a class within the project, the structure of that class is displayed in the Class Navigator, shown in the lower left part of the NetBeans window.

When you run a file or the entire Java program, any program output appears in the Output panel in the lower right part of the window.

Creating a Java Project

1. Select **File > New Project**.
2. Select Java Application.
3. Name and set the location for the project.
4. Select “Create Main Class” if you want it done for you automatically.
5. Click **Finish**.



New Java Application

Name and Location

Project Name: MyFirstProject

Project Location: /home/oracle/labs

Project Folder: /home/oracle/labs/MyFirstProject

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

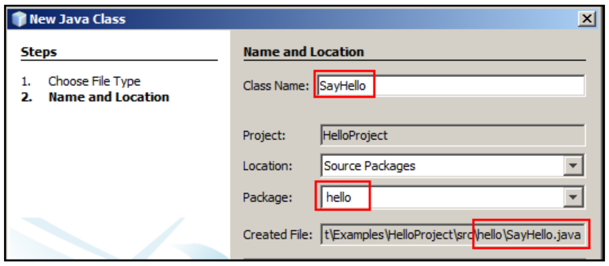
☑ Create Main Class

A NetBeans project is a mechanism for organizing the related files and resources used in a Java Application. To create a new project, perform the following steps:

1. Select **File > New Project** from the menu.
2. On the first page of the New Project Wizard (not shown here), select Java as the category and Java Application as the project type. Click **Next**.
3. On the second page of the wizard (shown above), enter a name for the project, and then enter or browse to the directory location to store project files.
4. It is possible to have NetBeans automatically generate a main class for the project.
5. Click **Finish**.

Creating a Java Class

1. Select **File > New File**.
2. Select your project and choose **Java Class**.
3. Name the class.
4. Assign a package.
5. Click **Finish**.



Class Name: SayHello

Project: HelloProject

Location: Source Packages

Package: hello

Created File: t\Examples\HelloProject\src\hello\SayHello.java

To create a class within your new project, perform the following steps:

1. Select **File > New File** from the menu.
2. On the first page of the New File Wizard, select your project, and then accept the default file type of Java Class. Click **Next**.
3. On the next page of the wizard, enter a name for the Java class. By convention, Java classes should start with an uppercase letter and each subsequent word in the class name should be capitalized (for example, `SayHello`). This is illustrated in the screenshot above.
4. Assign a package for the class.
5. Click **Finish**.

Note: If the package for this new class already exists, you can create the class by right-clicking the package in the Project Navigator panel in NetBeans and selecting **New > Java class** from the context menu instead of starting from the File menu.

Exercise 3-1:Creating a New Project and Java Class

In this exercise, you use NetBeans to create a new Java Class.

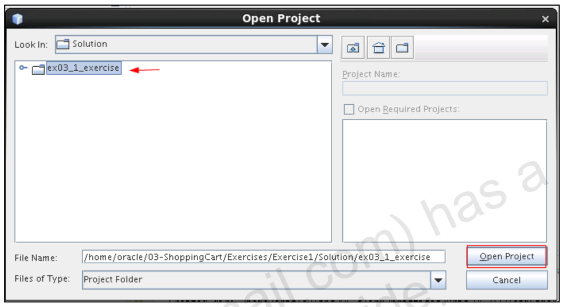
1. Create a new project called **`Exercise_03-1`**.
   – Deselect the box to create the `main` method. You will write the `main` method yourself in the next exercise.
2. Create a new **Java Class** file in this project.
   – Class name = `ShoppingCart`
   – Package name =`exercise`

The fully-qualified class name should be `exercise.ShoppingCart`. Note: You won't be able to run and test your code until create the main method in the next exercise.

Opening an Existing Java Project

If you need to open an existing project in NetBeans, perform the following steps:

1. Select **File > Open Project.**
2. Navigate to the directory that contains your projects.
3. Select the project file you want. (This file must be unzipped.)
4. Click **Open Project**.



File Name: /home/oracle/03-ShoppingCart/Exercises/Exercise1/Solution/ex03_1_exercise

Files of Type: Project Folder

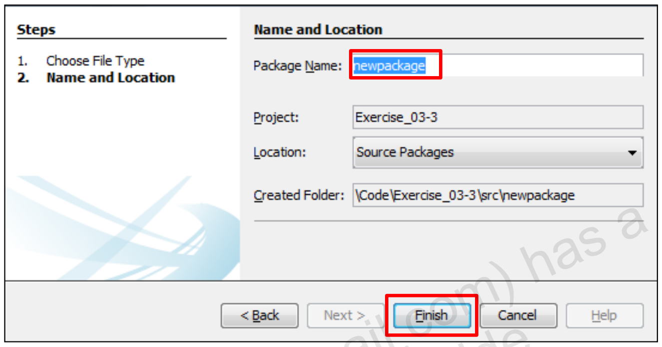
To open an existing project, perform the following steps:

1. Select **File > Open Project**.
2. Navigate to the directory that contains your projects.
3. Select the project file you want. (This file must be unzipped.)
4. Click **Open Project.**

Creating a New Java Package

If you ever need to create a new package, perform the following steps in NetBeans:

1. Right-click your project.
2. Select **New > Java Package**.
3. Name the package.
4. Click **Finish**.



Package Name: newpackage
Project: Exercise_03-3
Location: Source Packages
Created Folder: \Code\Exercise_03-3\src\newpackage

To create a new package within your new project, perform the following steps:

1. Right-click your project.
2. Select **New > Java Package**.
3. Name the package.
4. Click **Finish**.

Topics

- Java classes and packages
- 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
}
```

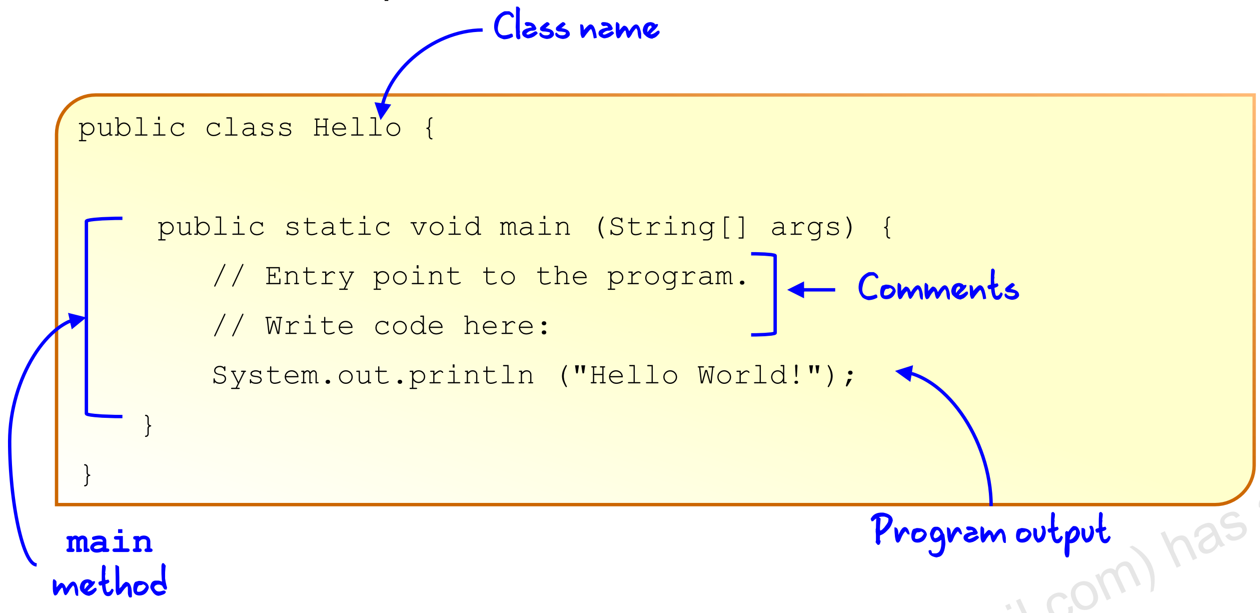
- It surrounds entire method body with braces { } .

- 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

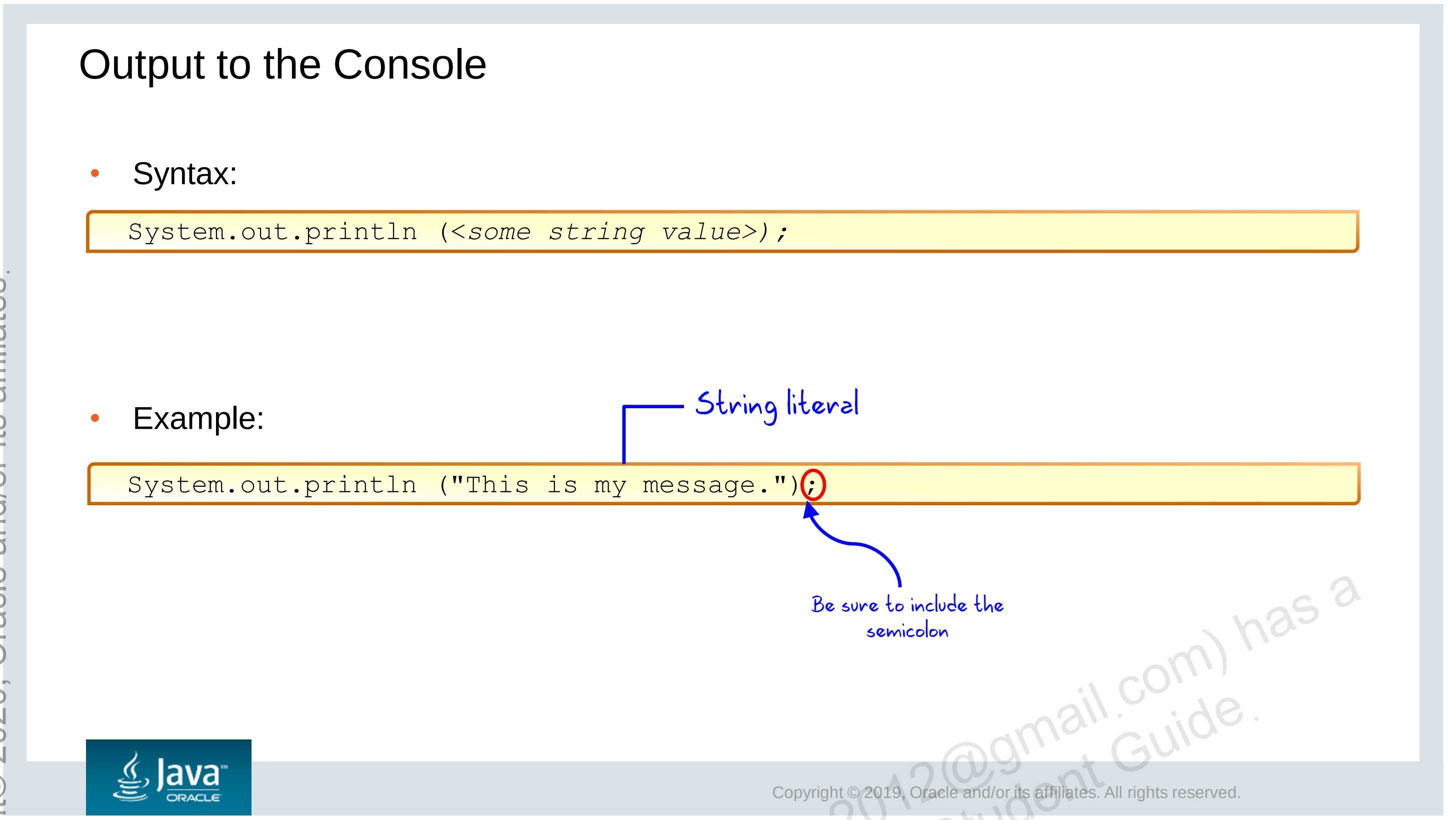


```
public class Hello {

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

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

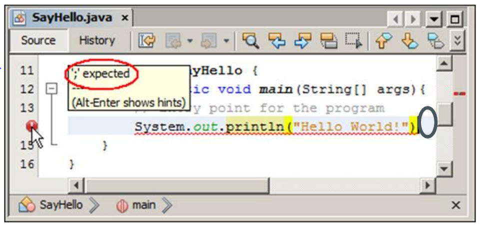
```
System.out.println ("This is my message.");
```

Be sure to include the semicolon

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).

Avoiding Syntax Errors

- NetBeans will tell you if you have done something wrong.
- Common errors include:
  - Unrecognized word (check for case-sensitivity error)
  - Missing close quotation mark
  - Unmatched brace
  - Missing semicolon



```
11          yHello {
12          ic void main(String[] args){
13          y point for the program
                System.out.println("Hello World!")
15          }
16      }
```

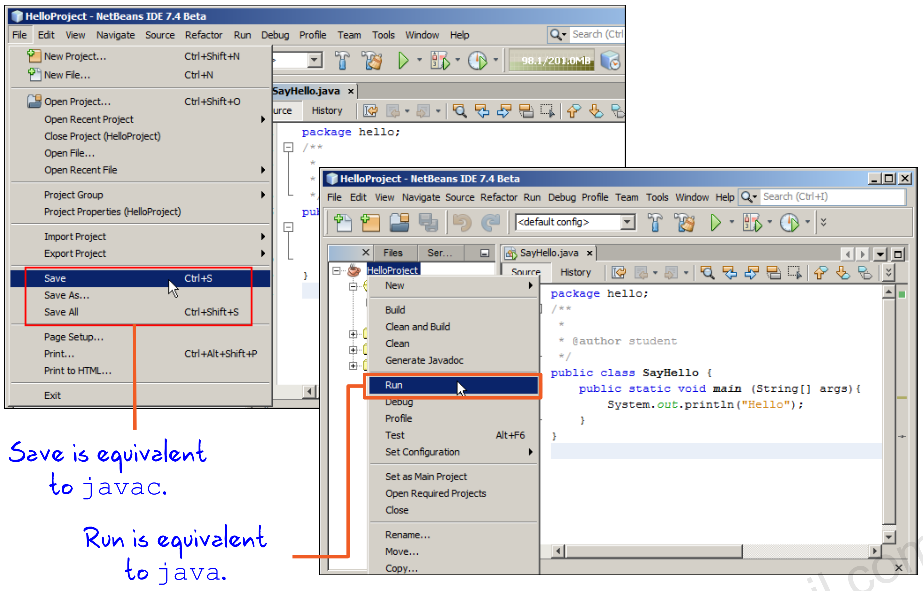
Most Java editors check the code syntax and show alerts by using icons and red underlines where there are errors in the code.

To avoid syntax problems, be sure to do the following:

- Observe any red bubble indicators in the code editor to locate syntax errors.
- Have a semicolon at the end of every line where one is required.
- Have an even number of symbols such as braces, brackets, and quotation marks.

The screenshot shows an error in Line 13, in which there is a missing semicolon. If you place your cursor over the red bubble, the editor offers a suggestion for fixing the error.

Compiling and Running a Program by Using NetBeans



```
package hello;
/**
 *
 * @author student
 */
public class SayHello {
    public static void main (String[] args){
        System.out.println("Hello");
    }
}
```

Save invokes the `javac <classname(s)>` command for all `.java` files in the project. Right-clicking the source code and selecting Run File invokes the `java <classname>` command. Be sure to look for red bubble indicators in the code editor to locate syntax errors.

Exercise 3-2: Creating a `main` Method

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

1. Continue editing **`Exercise_03-1`** or open **`Exercise_03-2`**.
2. In the code editor, add the `main` method structure to the ShoppingCart class.
3. In the code block of the `main` method, use a `System.out.println` method to print "Welcome to the Shopping Cart!"
4. Save your program.
5. Click the **Run** button to test program.
   – Select **`exercise.ShoppingCart`** as the main class.

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

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:

- Use the NetBeans IDE to create and test Java classes
- Write a `main` method
- Use `System.out.println` to write a String literal to system output