

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



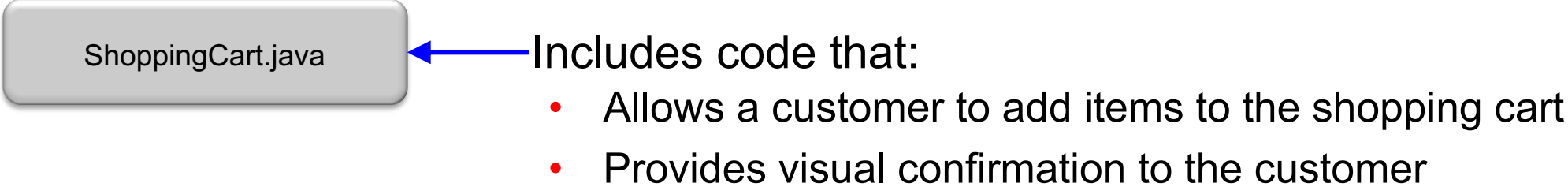
Topics

- Java classes and packages
- The `main` method



Java Classes

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



ShoppingCart.java

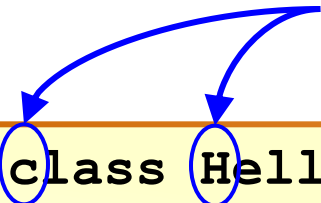
Includes code that:

- 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!



The diagram shows two blue arrows originating from the text "Java is case-sensitive!". One arrow points to the word "class" in the code snippet, and the other points to the word "Hello". In the code, "class" is circled in blue, while "Hello" is not, despite both starting with a capital letter. This illustrates that Java is case-sensitive, meaning "class" and "Class" would be treated as different identifiers.

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

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 greeting;
```

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

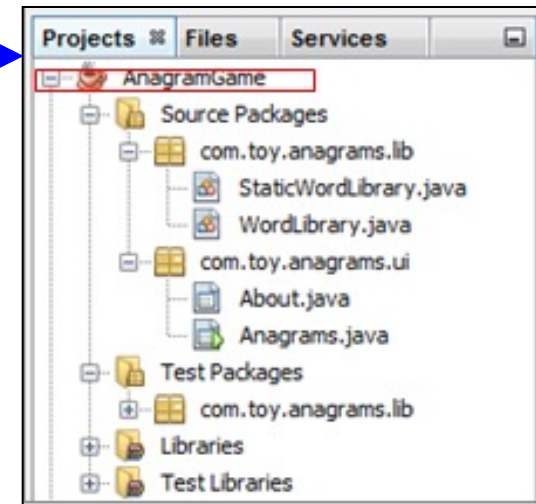
Package name

The class's unique name is:
greeting.Hello

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`.

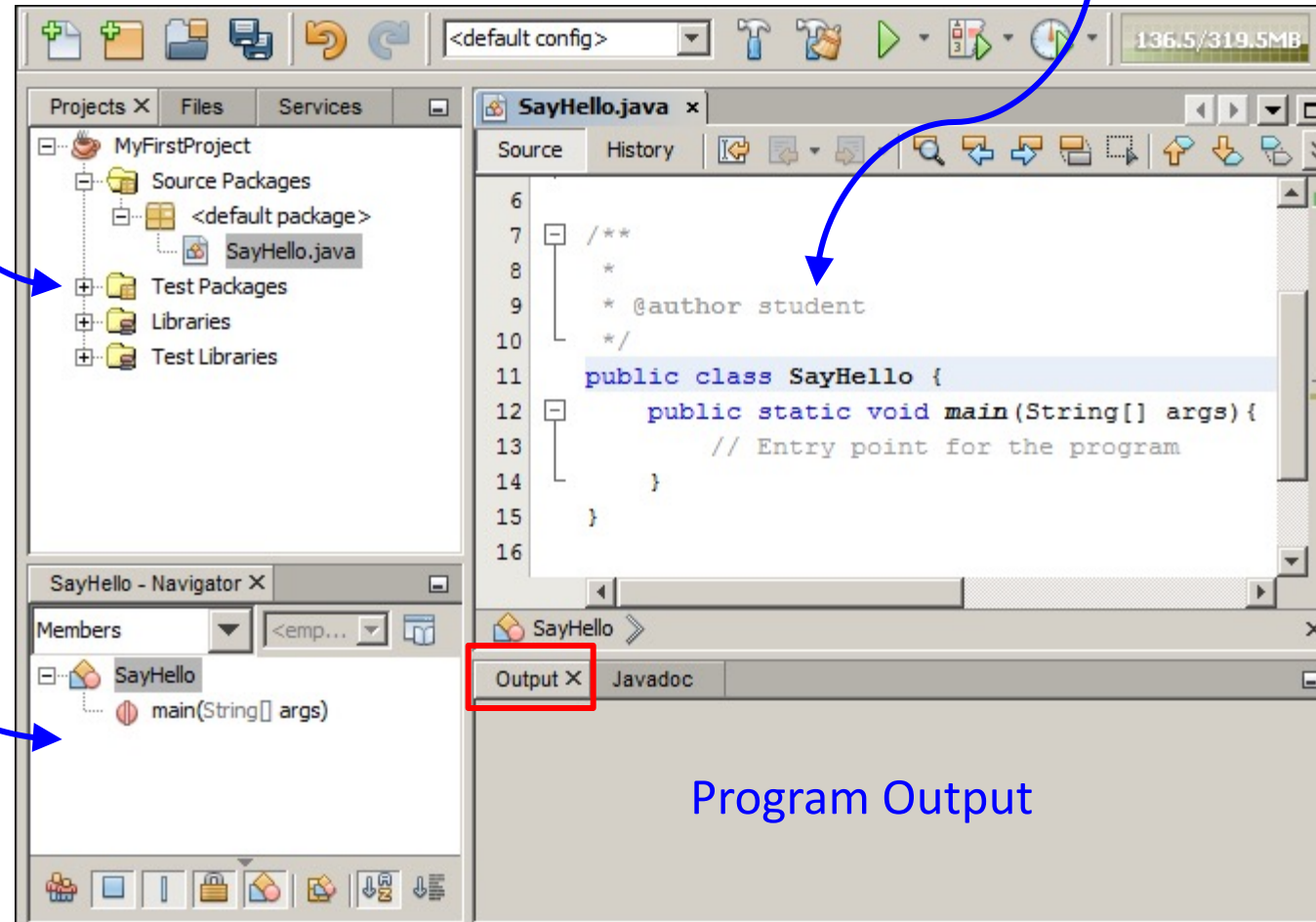


The NetBeans IDE

Project
Navigator

Class
Navigator

Code Editor



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**.

The screenshot shows the 'New Java Application' dialog box. The 'Name and Location' section contains the following fields:

- Project Name: MyFirstProject
- Project Location: /home/oracle/labs (with a 'Browse...' button)
- Project Folder: /home/oracle/labs/MyFirstProject

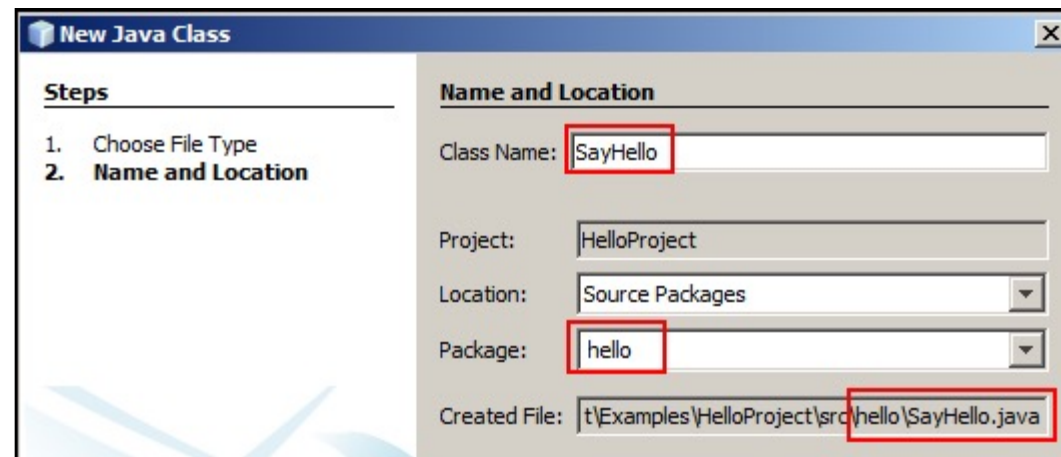
Below these fields, there are two checkboxes:

- ☐ Use Dedicated Folder for Storing Libraries. Below this is a 'Libraries Folder:' field with a 'Browse...' button. A note states: 'Different users and projects can share the same compilation libraries (see Help for details).'
- ☒ Create Main Class. This checkbox is checked, and there is an empty text field next to it.

At the bottom of the dialog, there are five buttons: '< Back', 'Next >', 'Finish' (highlighted with a red box), 'Cancel', and 'Help'.

Creating a Java Class

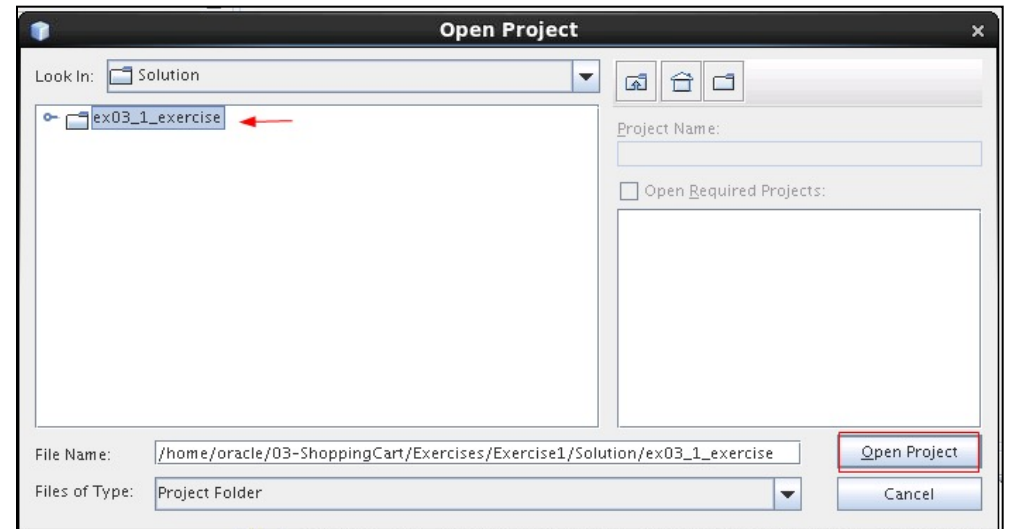
1. Select **File > New File**.
2. Select your project and choose **Java Class**.
3. Name the class.
4. Assign a package.
5. Click **Finish**.



Opening an Existing Java Project

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

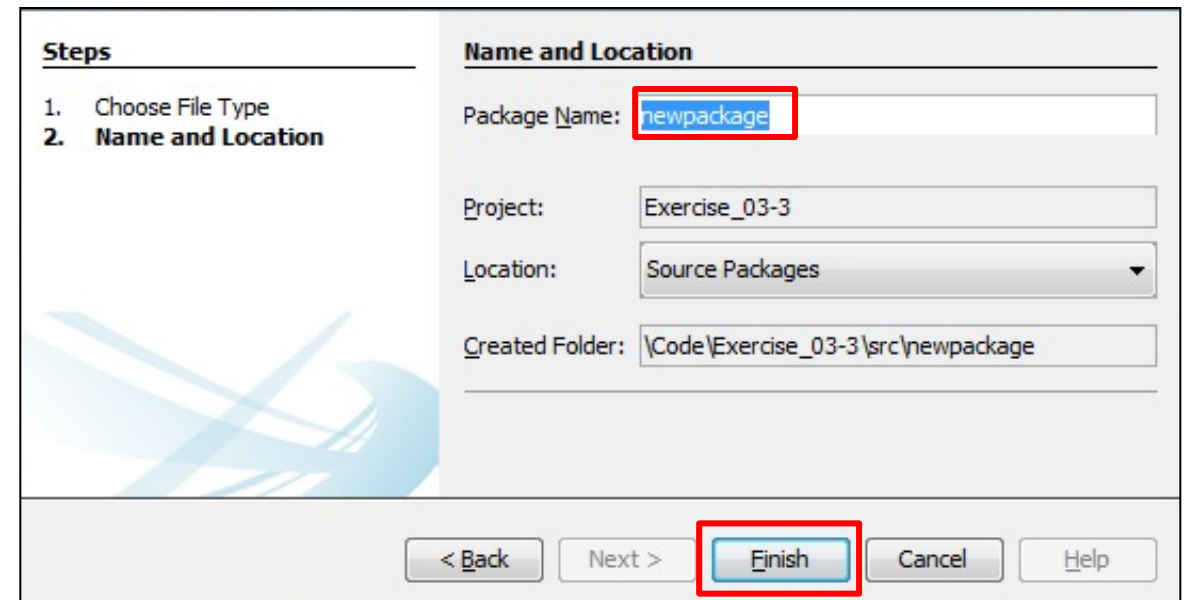
1. Select **File > Open Project**.
2. Navigate to the directory that contains your projects.
3. Select the project file you want. (This file must be unzipped.)
4. Click **Open Project**.



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**.



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 `{ }` .

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

Program output

**main
method**


Output to the Console

- Syntax:

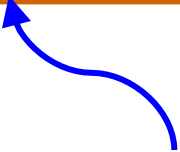
```
System.out.println (<some string value>);
```

- Example:

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



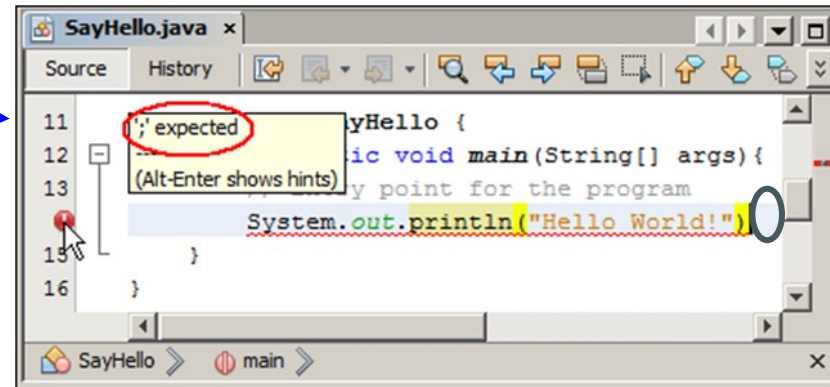
String literal



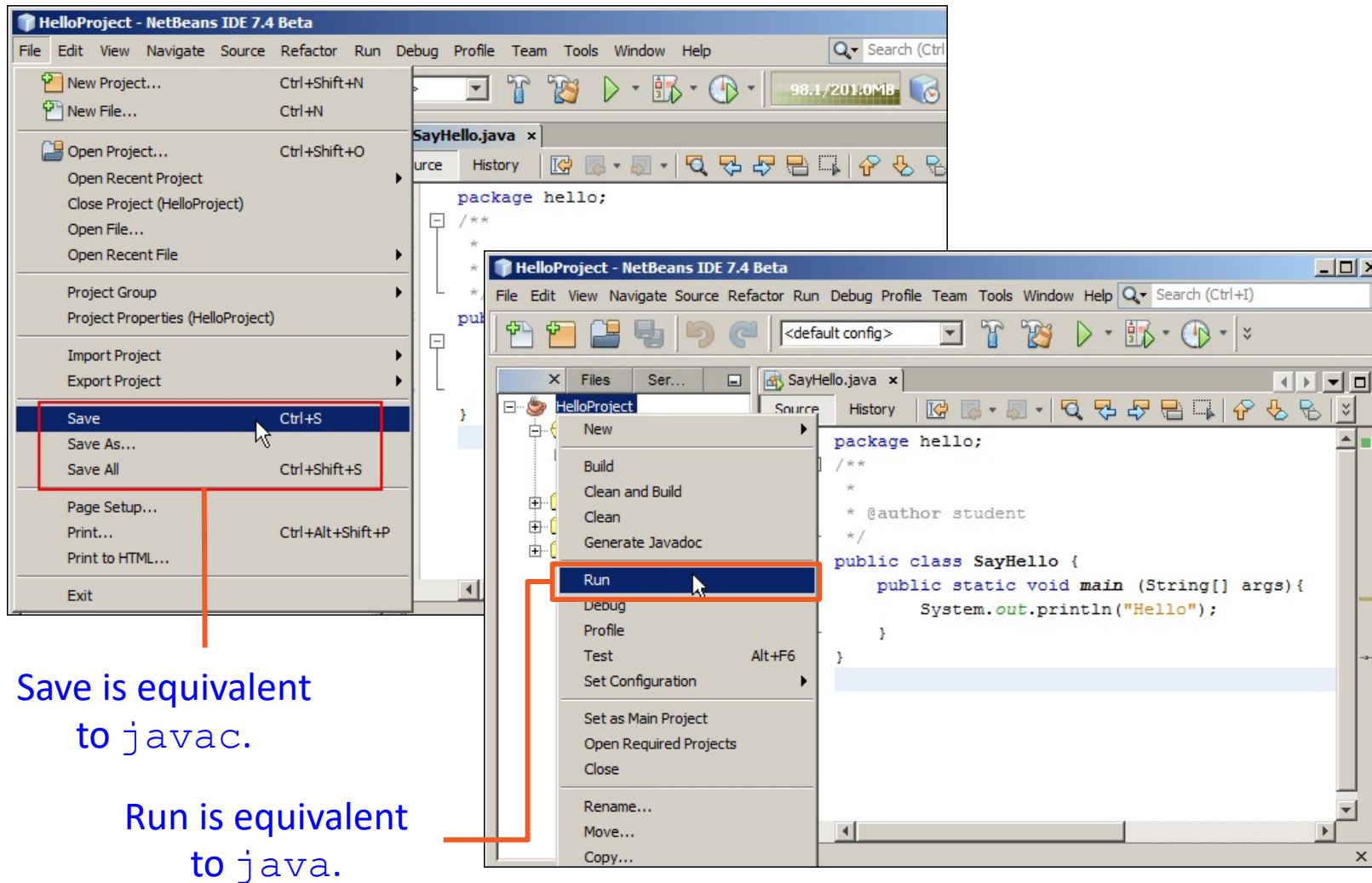
Be sure to include the
semicolon

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



Compiling and Running a Program by Using NetBeans



Save is equivalent
to javac.

Run is equivalent
to java.

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){ }`



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

