

# **Murach's Java Programming (6<sup>th</sup> Edition)**

## **Chapter 1**

# **An introduction to Java**

# Objectives

## Applied

1. Given a NetBeans or Eclipse project that contains the source code for a Java application, use NetBeans or Eclipse to open the project, view and compile the source code, and run the application.
2. Given the source code for a Java application, use NetBeans or Eclipse to create a project, enter the source code, compile the source code, and run the application.

## Knowledge

1. Describe how Java compares with C++ and C# based on these features: syntax, platform independence, speed, and memory management.
2. Name and describe two types of desktop applications that you can create with Java.
3. Describe how Java compiles and interprets code.
4. Explain how the use of bytecode lets Java achieve platform independence.

## Objectives (continued)

5. Describe the benefits of using a Java IDE like NetBeans or Eclipse.
6. Explain why you don't need to compile the source code for an application before you use NetBeans or Eclipse to run the application.

## Java timeline

Year	Month	Release
1996	January	JDK 1.0
1997	February	JDK 1.1
1998	December	SDK 1.2
1999	August	Java 2 Platform, Standard Edition (J2SE)
	December	Java 2 Platform, Enterprise Edition (J2EE)
2000	May	J2SE with SDK 1.3
2002	February	J2SE with SDK 1.4
2004	September	J2SE 5.0 with JDK 1.5
2006	December	Java SE 6 with JDK 1.6
2011	July	Java SE 7 with JDK 1.7
2014	March	Java SE 8 with JDK 1.8
2017	September	Java SE 9 with JDK 1.9

## Java timeline (continued)

Year	Month	Release
2018	March	Java SE 10 with JDK 10
2018	September	Java SE 11 with JDK 11
2019	March	Java SE 12 with JDK 12
2019	September	Java SE 13 with JDK 13
2020	March	Java SE 14 with JDK 14
2020	September	Java SE 15 with JDK 15
2021	March	Java SE 16 with JDK 16
2021	September	Java SE 17 with JDK 17

# Java editions

- Java SE (Standard Edition)
- Java EE (Enterprise Edition)
- Java ME (Micro Edition)

# Operating systems that support Java

- Windows
- macOS
- Linux
- Most versions of UNIX
- Android
- Most other modern operating systems

## Java compared to C++

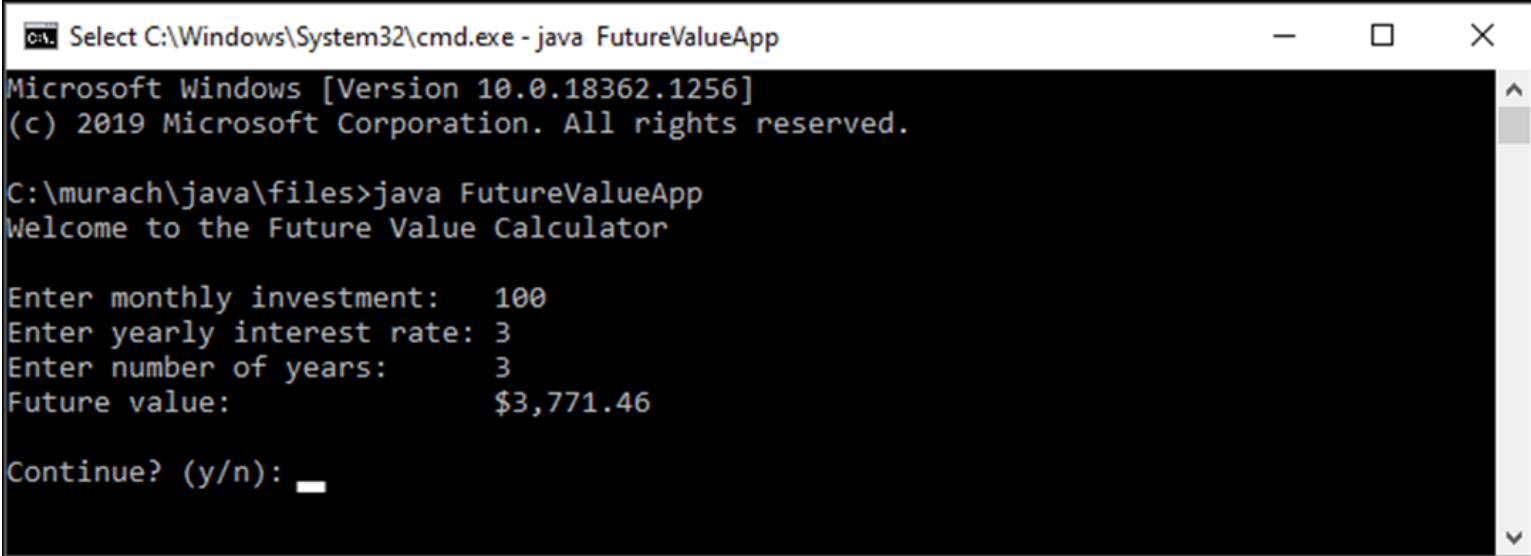
Feature	Description
Syntax	Java syntax is similar to C++ syntax.
Operating system	Compiled Java code can run on any operating system that has a Java runtime environment. C++ code must be compiled once for each type of system that it's going to be run on.
Speed	C++ runs faster than Java in some contexts, but Java runs faster in other contexts.
Memory	Java handles most memory operations automatically, but C++ programmers must write code that manages memory.



## Java compared to C#

Feature	Description
Syntax	Java syntax is similar to C# syntax.
Operating system	Like Java, compiled C# code can run on any operating system that has a runtime environment for it.
Speed	Java runs faster than C# in most contexts.
Memory	Like Java, C# handles most memory operations automatically.

# A console application



The screenshot shows a Windows command prompt window titled "C:\Windows\System32\cmd.exe - java FutureValueApp". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt displays the following text:

```
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

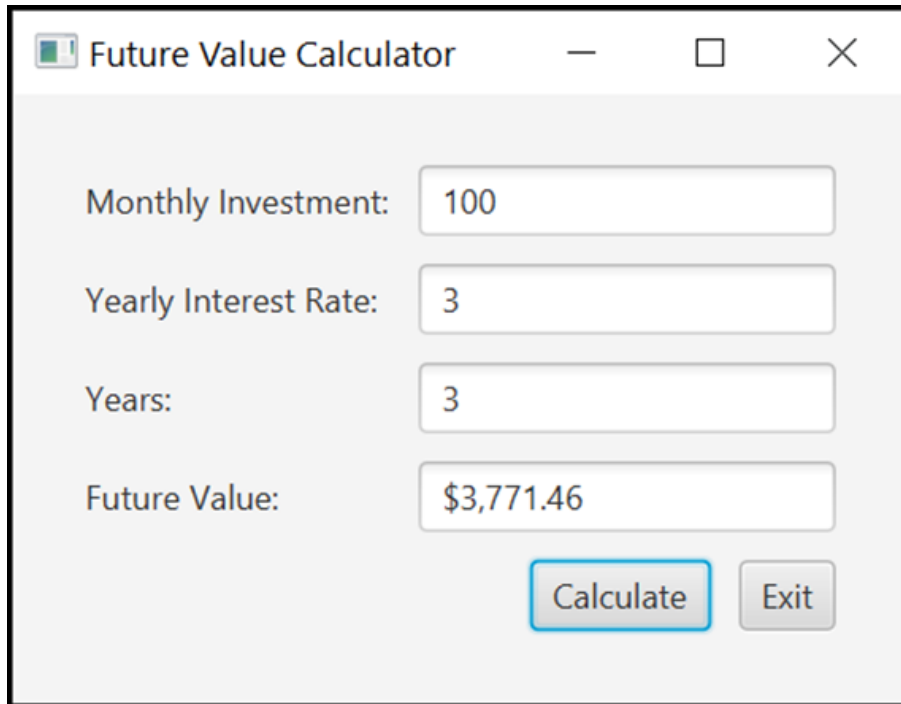
C:\murach\java\files>java FutureValueApp
Welcome to the Future Value Calculator

Enter monthly investment: 100
Enter yearly interest rate: 3
Enter number of years: 3
Future value: $3,771.46

Continue? (y/n): _
```

The user has entered '100' for monthly investment, '3' for yearly interest rate, and '3' for number of years. The program has calculated a future value of \$3,771.46 and is now asking if the user wants to continue.

# A GUI application



The image shows a screenshot of a graphical user interface (GUI) application titled "Future Value Calculator". The window has a standard title bar with a minimize button, a maximize button, and a close button. The main area of the window contains four input fields and two buttons. The first three input fields are labeled "Monthly Investment:", "Yearly Interest Rate:", and "Years:", and each contains the value "100", "3", and "3" respectively. The fourth input field is labeled "Future Value:" and contains the calculated result "\$3,771.46". Below the input fields are two buttons: "Calculate" and "Exit". The "Calculate" button is highlighted with a blue border, indicating it is the active button.

Label	Value
Monthly Investment:	100
Yearly Interest Rate:	3
Years:	3
Future Value:	\$3,771.46

Buttons: Calculate, Exit

# The code for a console application (part 1)

```
import java.util.Scanner;
import java.text.NumberFormat;

public class FutureValueApp {

    public static void main(String[] args) {
        System.out.println(
            "Welcome to the Future Value Calculator\n");

        Scanner sc = new Scanner(System.in);
        String choice = "y";
        while (choice.equals("y")) {

            // get the input from the user
            System.out.print(
                "Enter monthly investment:  ");
            String input = sc.nextLine();
            double monthlyInvestment =
                Double.parseDouble(input);
            System.out.print(
                "Enter yearly interest rate: ");
            input = sc.nextLine();
            double interestRate = Double.parseDouble(input);
```

## The code for a console application (part 2)

```
        System.out.print(
            "Enter number of years:      ");
        input = sc.nextLine();
        int years = Integer.parseInt(input);

        // calculate the future value
        double futureValue = calculateFutureValue(
            monthlyInvestment,
            interestRate, years);

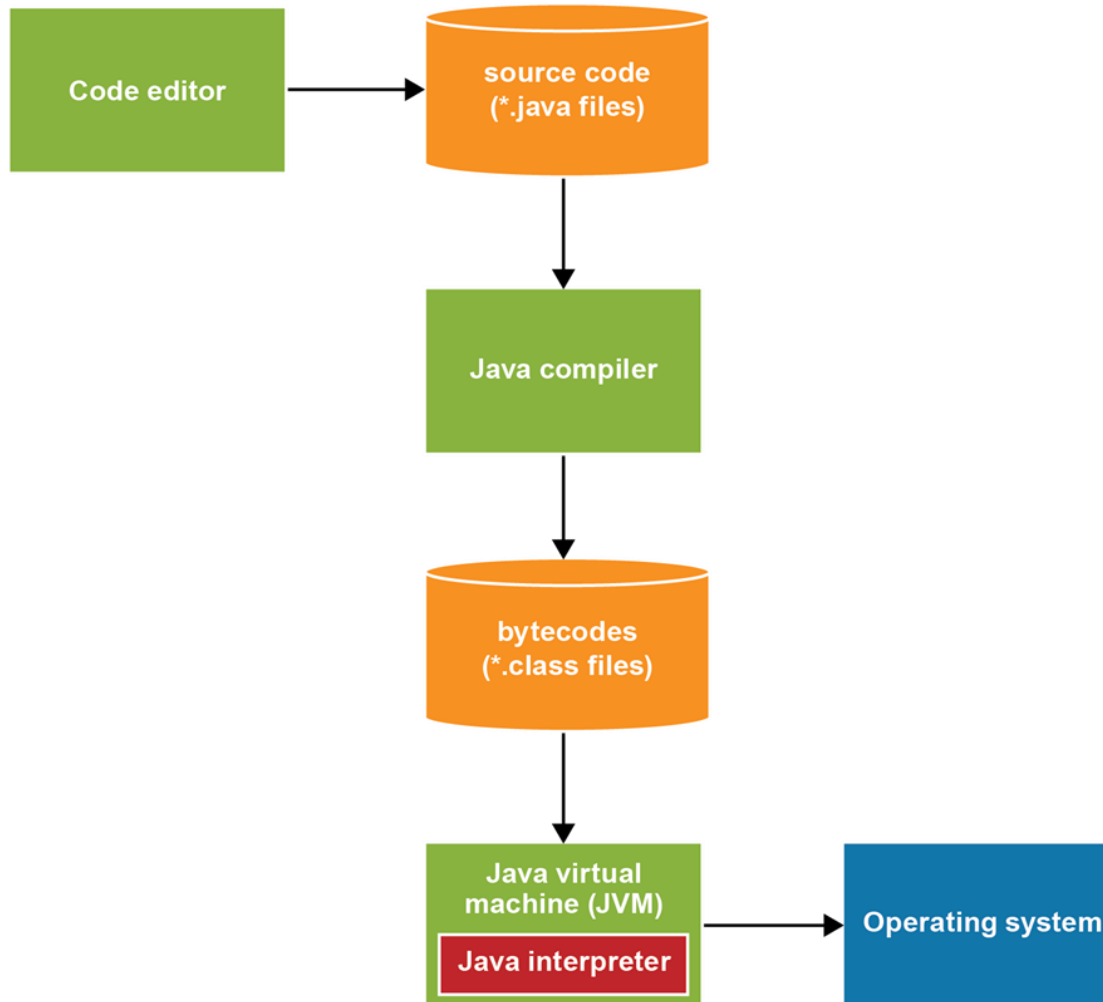
        // format and display the result
        NumberFormat currency =
            NumberFormat.getCurrencyInstance();
        System.out.println("Future value:      "
            + currency.format(futureValue));
        System.out.println();

        // see if the user wants to continue
        System.out.print("Continue? (y/n): ");
        choice = sc.nextLine();
        System.out.println();
    }
}
```

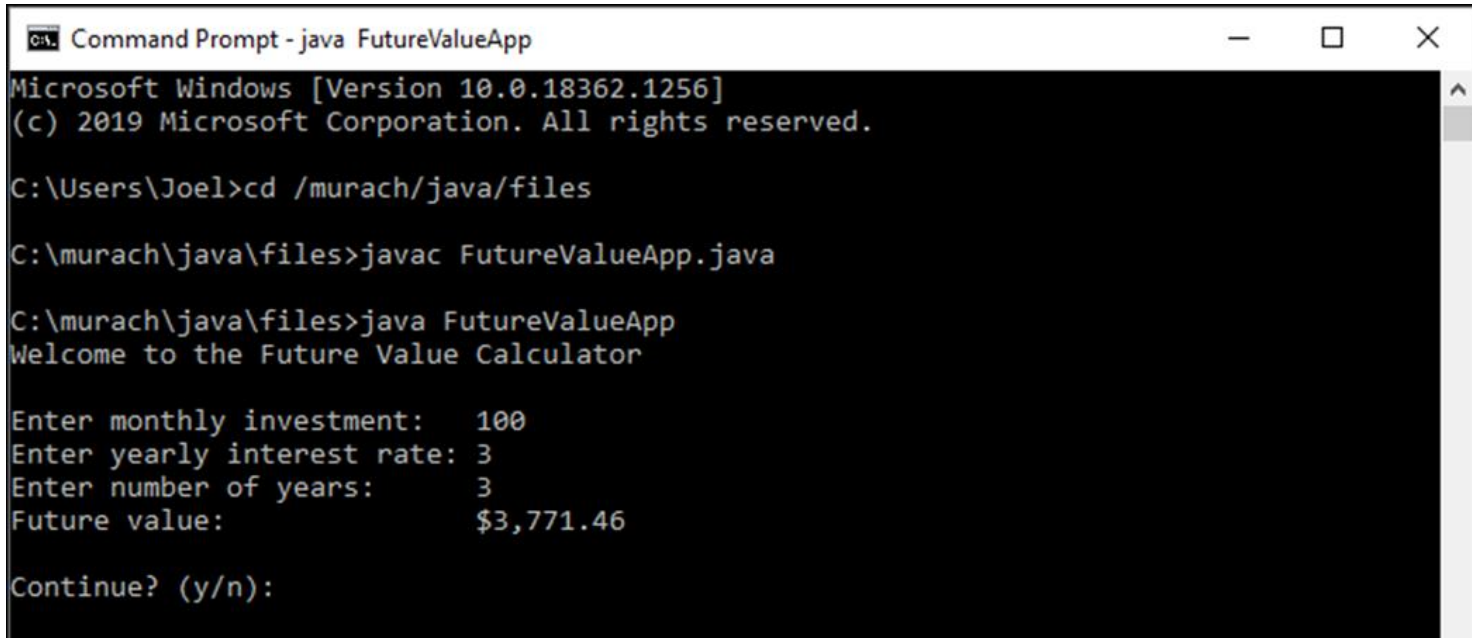
## The code for a console application (part 3)

```
private static double calculateFutureValue(  
    double monthlyInvestment,  
    double interestRate, int years) {  
    // convert yearly to monthly  
    double monthlyInterestRate = interestRate / 12 / 100;  
    int months = years * 12;  
  
    double futureValue = 0.0;  
    for (int i = 1; i <= months; i++) {  
        futureValue = (futureValue + monthlyInvestment)  
            * (1 + monthlyInterestRate);  
    }  
    return futureValue;  
}  
}
```

# How Java compiles and interprets code



# A command prompt for Windows



```
Command Prompt - java FutureValueApp
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Joel>cd /murach/java/files

C:\murach\java\files>javac FutureValueApp.java

C:\murach\java\files>java FutureValueApp
Welcome to the Future Value Calculator

Enter monthly investment: 100
Enter yearly interest rate: 3
Enter number of years: 3
Future value: $3,771.46

Continue? (y/n):
```



# How to start a command prompt

## For Windows

- Select the Command Prompt app from the Start menu. To do that, you can search the Start menu for the Command Prompt app.

## For macOS

- Use Finder to select Applications. Then, expand Utilities and double-click on Terminal.

# Syntax to change the current directory

```
cd /dir1/dir2/dir3
```

## A note on using slashes

- When you change the current directory, front slashes work for both Windows and macOS, but backslashes only work for Windows.

# How to compile and run a Java application

## Syntax to compile a Java application

```
javac ProgramName.java
```

## Syntax to run a Java application

```
java ProgramName
```

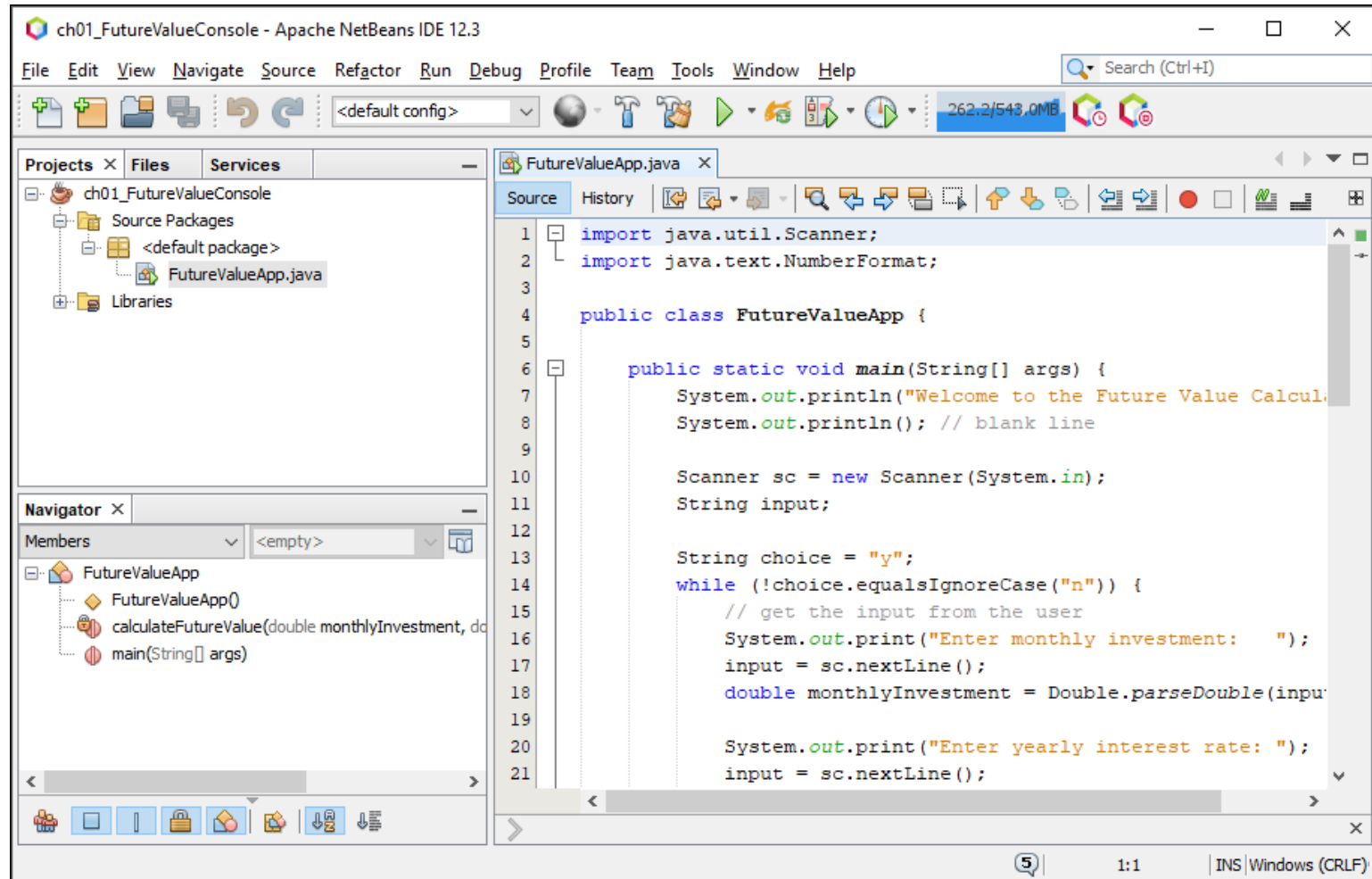
# Popular Java IDEs

- NetBeans
- Eclipse
- IntelliJ IDEA
- Android Studio

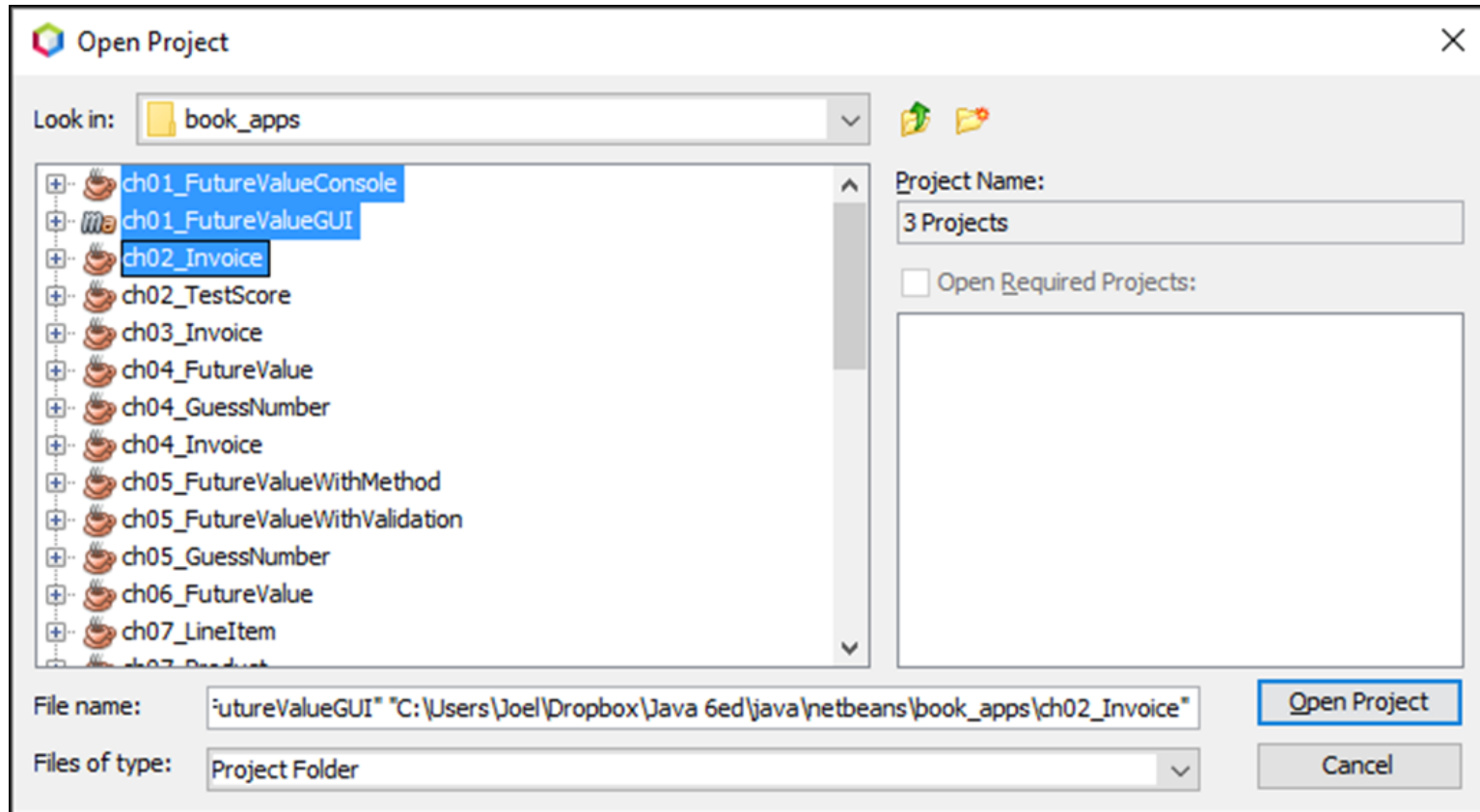
## Features provided by most IDEs

- A code editor with code completion and error detection.
- Automatic compilation of classes when you run the application.
- A debugger that lets you set breakpoints, step through code, and view the values of active variables.

# NetBeans with a Java project



# The dialog box for opening a project in NetBeans



# How to open and close a project in NetBeans

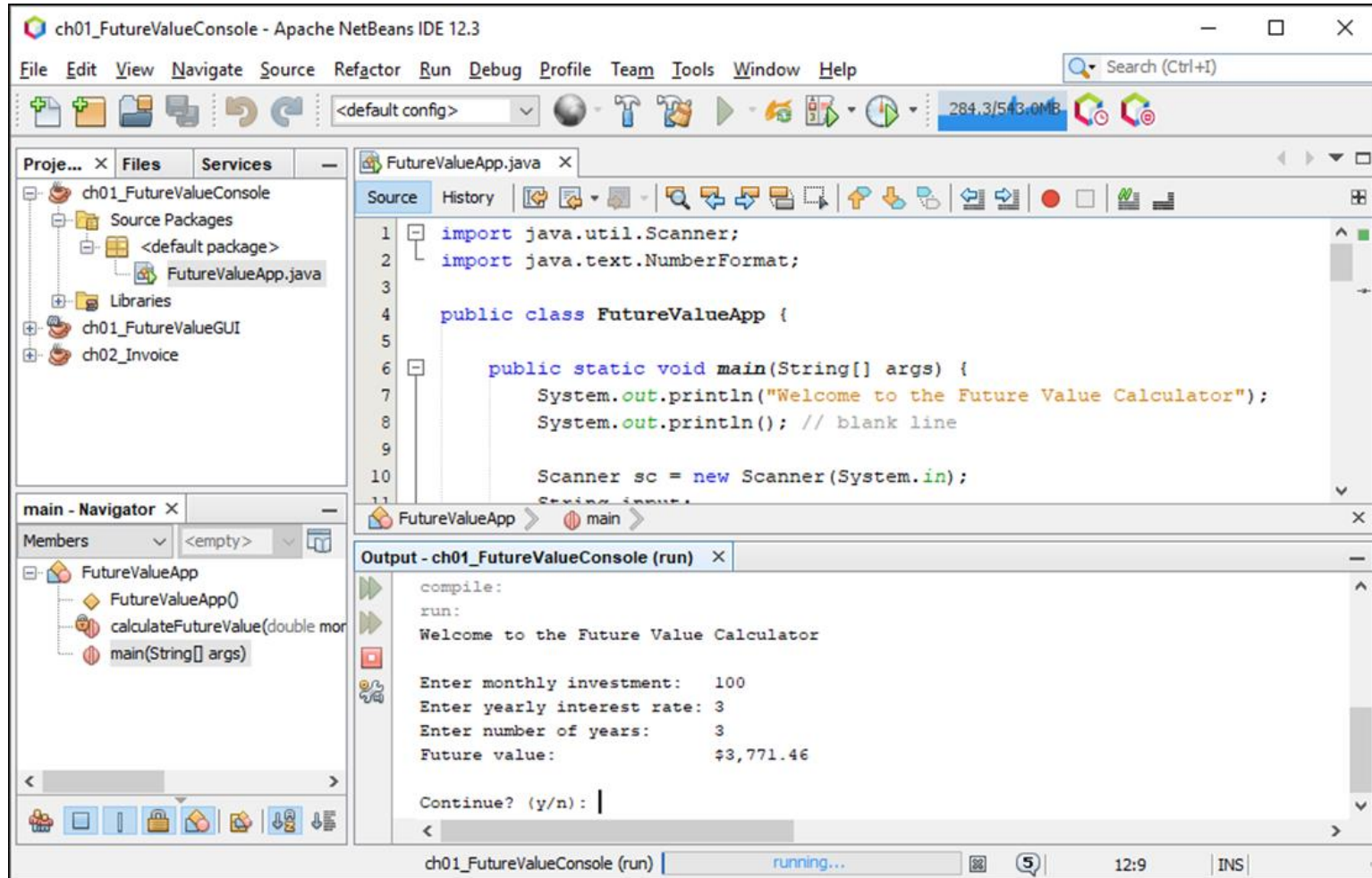
- To open a project, click the Open Project button in the toolbar or select File→Open Project from the menu system. Then, use the Open Project dialog box that's displayed to locate and select the project and click the Open Project button.
- You can also open a project by selecting File→Open Recent Project and then selecting the project from the list that's displayed.
- To close a project, right-click on the project in the Projects window and select Close, or select the project and then use the File→Close Project command.
- You can use the Open Project dialog box to open multiple projects. To do that, hold down the Ctrl key, select the projects that you want to open, and click the Open Project button.



# How to right-click with macOS

- Enable right-clicking by editing the system preferences for the mouse.
- Control-click instead of right-clicking.

# NetBeans with three open projects and an Output window



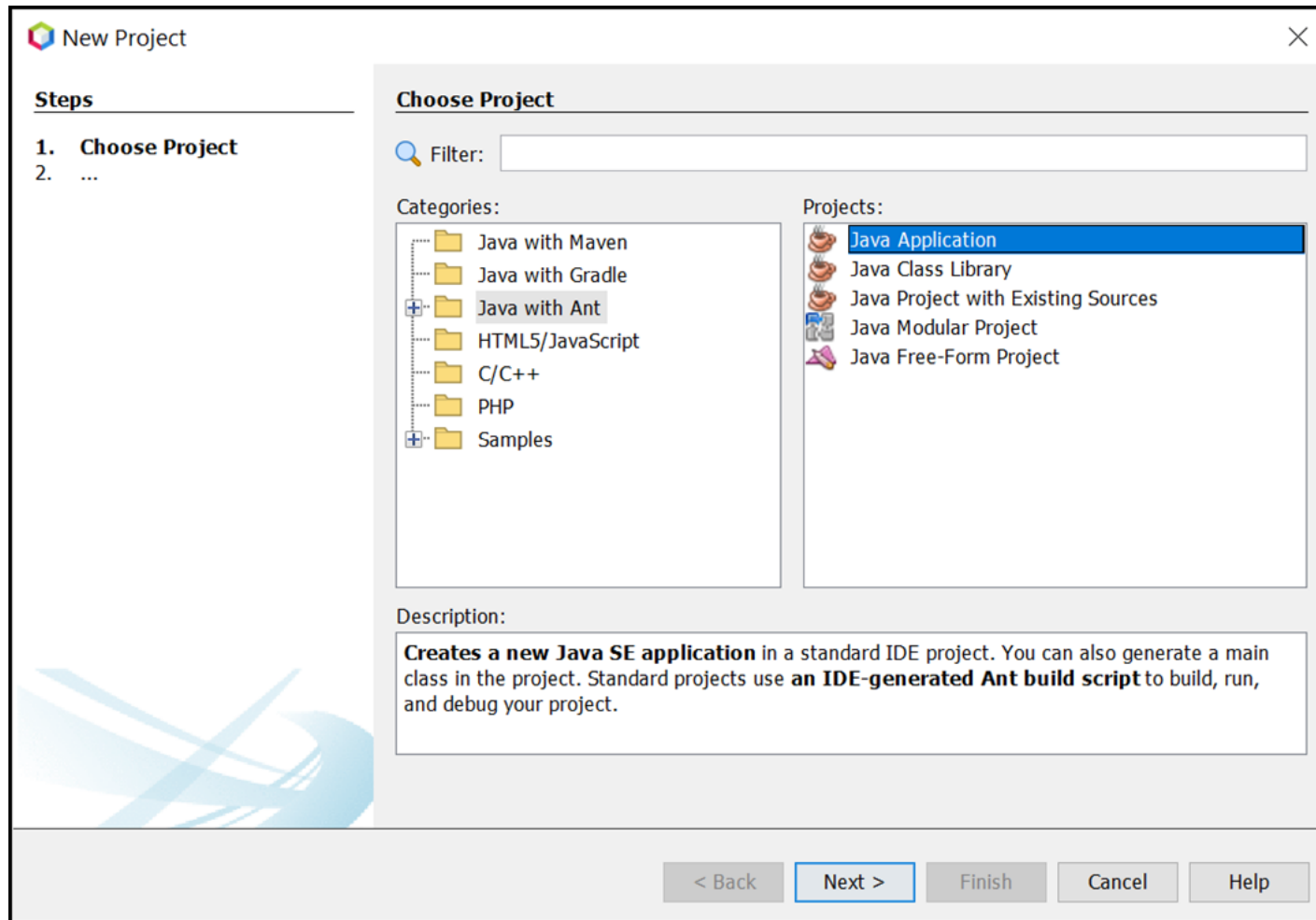
# How to compile and run a project with NetBeans

- If you open multiple projects, they all appear in the Projects window. To select a project, click on it in the Projects window.
- To run the selected project, press F6 or click the Run Project button in the toolbar.
- When you run a project, NetBeans automatically compiles it.
- To compile a project without running it, right-click on the project and select Build.
- To delete all compiled files for a project and compile the project again, right-click the project and select Clean and Build. This removes files that are no longer needed.

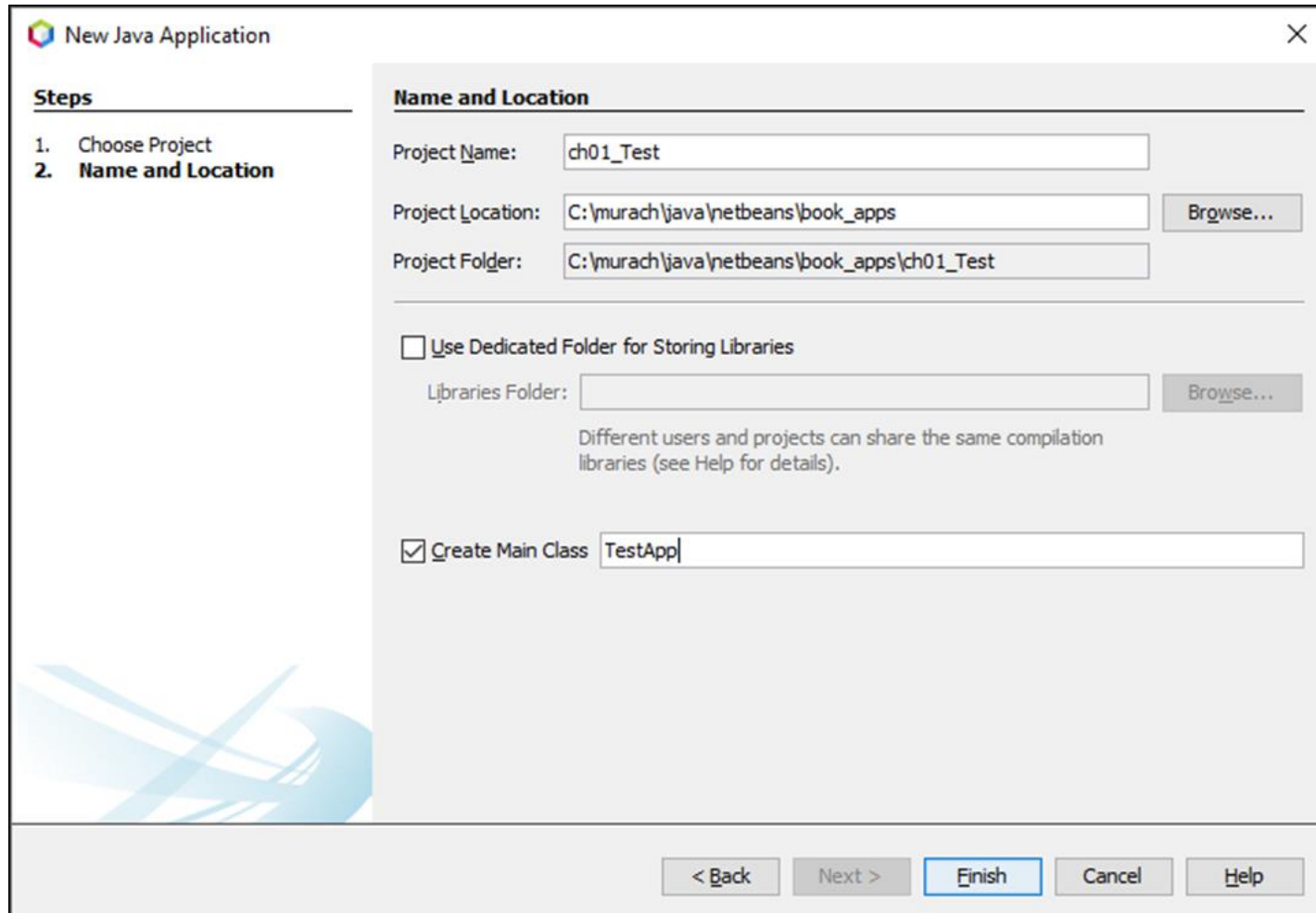
## How to work with the Output window

- When you run an application that prints data to the console, NetBeans displays that data in the Output window.
- If an application requests input from the console, the Output window pauses to accept the input. Then, you can click in the Output window, type the input, and press the Enter key.
- The Output window also displays other information such as messages or errors that occur when the application is compiled or run.

# The first dialog box for creating a new project



# The second dialog box for creating a new project



The screenshot shows the 'New Java Application' dialog box, specifically the 'Name and Location' step. The dialog has a title bar with the NetBeans logo and a close button. On the left, a 'Steps' panel lists two steps: '1. Choose Project' and '2. Name and Location', with the second step being the active one. The main area contains several input fields and checkboxes. The 'Project Name' field is set to 'ch01\_Test'. The 'Project Location' field is set to 'C:\murach\java\netbeans\book\_apps', with a 'Browse...' button to its right. The 'Project Folder' field is set to 'C:\murach\java\netbeans\book\_apps\ch01\_Test'. Below these, there is a checkbox for 'Use Dedicated Folder for Storing Libraries', which is currently unchecked. Below this checkbox is a 'Libraries Folder' field and another 'Browse...' button. A note states: 'Different users and projects can share the same compilation libraries (see Help for details)'. At the bottom of the main area, there is a checked checkbox for 'Create Main Class' followed by a text field containing 'TestApp'. At the very bottom of the dialog, there are five buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), 'Cancel', and 'Help'.

**New Java Application**

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

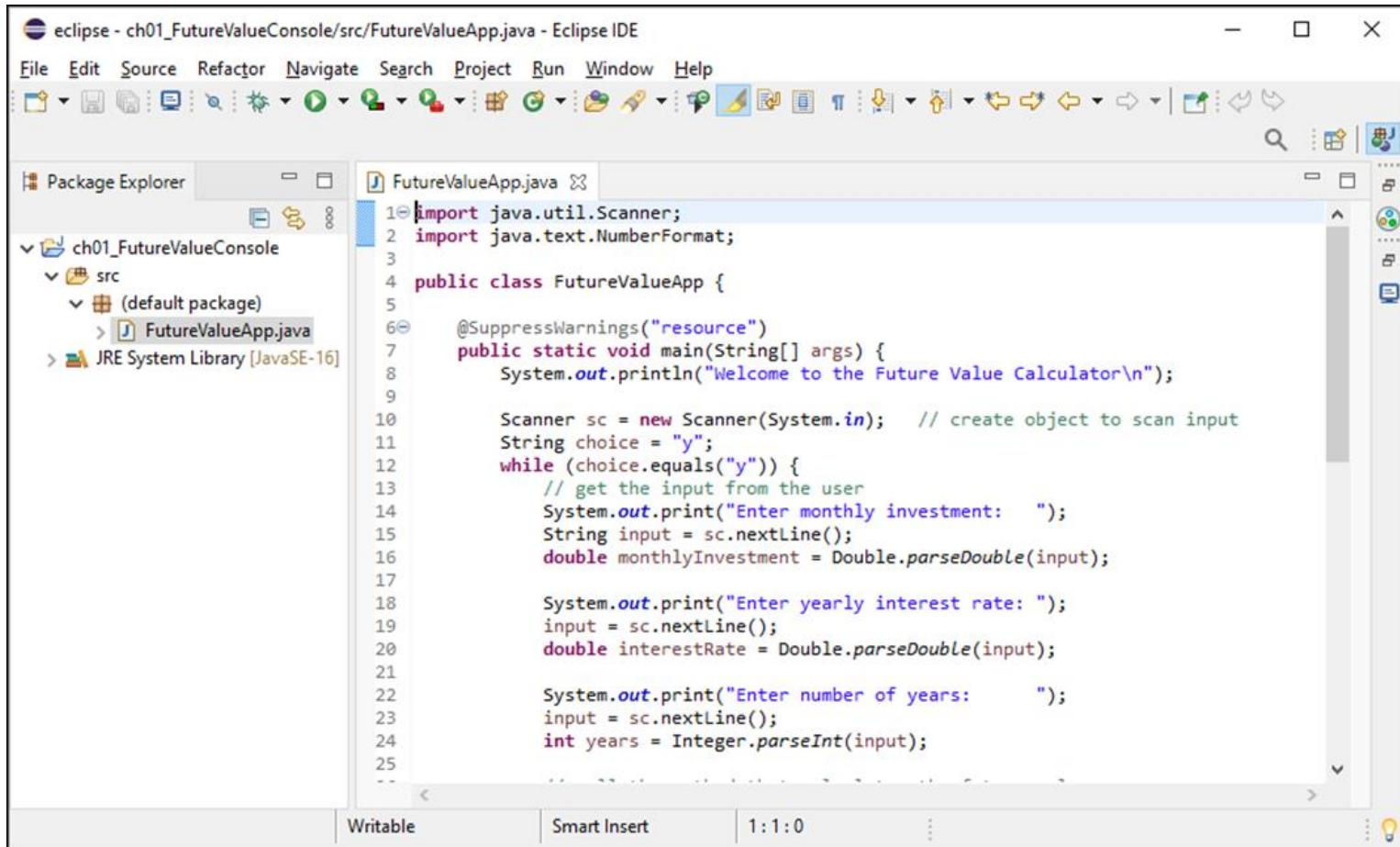
Libraries Folder:

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

☒ Create Main Class

< Back   Next >   **Finish**   Cancel   Help

# Eclipse with a Java project

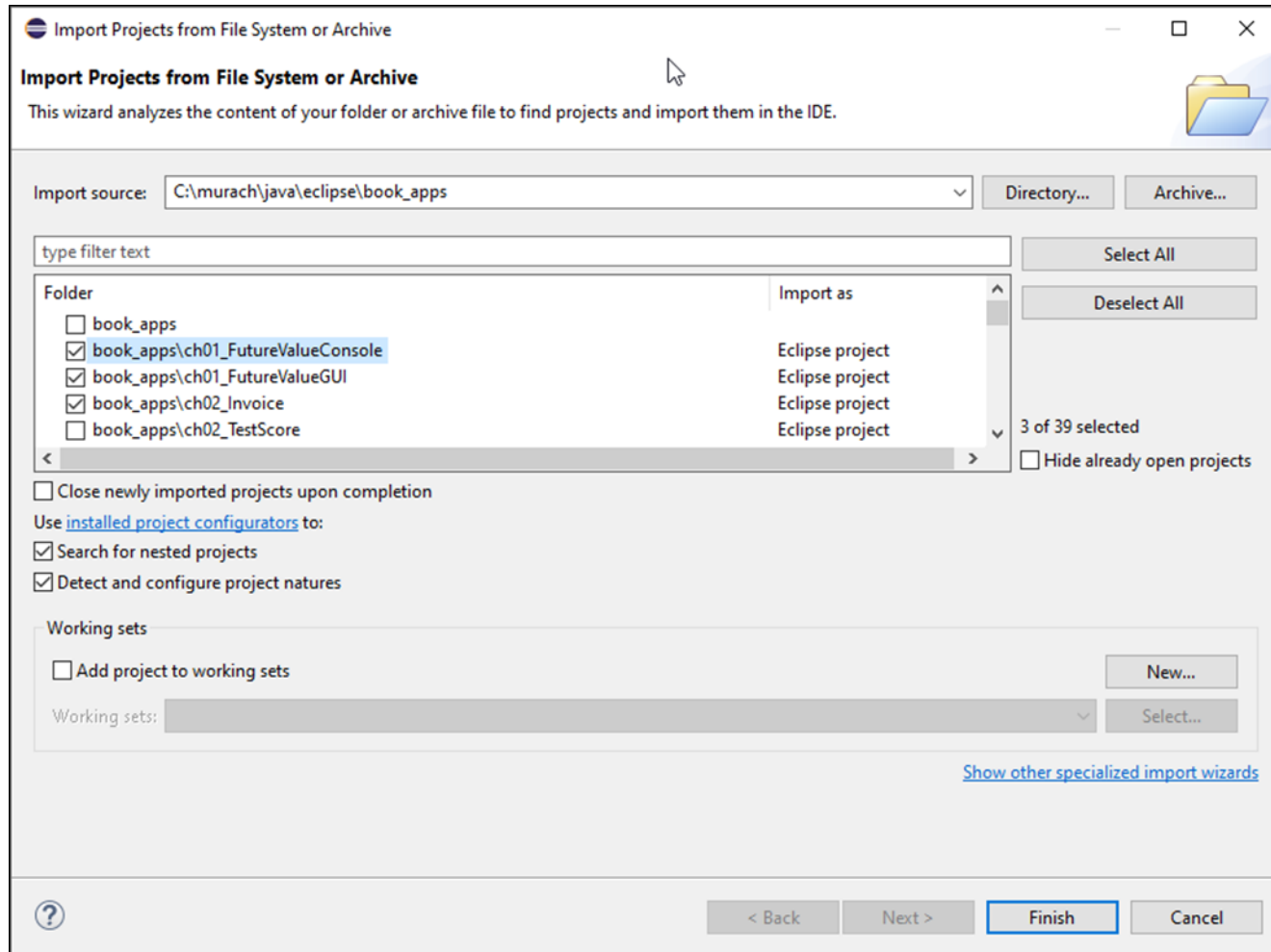


The screenshot displays the Eclipse IDE interface. The title bar reads 'eclipse - ch01\_FutureValueConsole/src/FutureValueApp.java - Eclipse IDE'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure: 'ch01\_FutureValueConsole' containing a 'src' package, which in turn contains '(default package)' and the file 'FutureValueApp.java'. The main editor window shows the code for 'FutureValueApp.java'.

```
1 import java.util.Scanner;
2 import java.text.NumberFormat;
3
4 public class FutureValueApp {
5
6     @SuppressWarnings("resource")
7     public static void main(String[] args) {
8         System.out.println("Welcome to the Future Value Calculator\n");
9
10        Scanner sc = new Scanner(System.in); // create object to scan input
11        String choice = "y";
12        while (choice.equals("y")) {
13            // get the input from the user
14            System.out.print("Enter monthly investment: ");
15            String input = sc.nextLine();
16            double monthlyInvestment = Double.parseDouble(input);
17
18            System.out.print("Enter yearly interest rate: ");
19            input = sc.nextLine();
20            double interestRate = Double.parseDouble(input);
21
22            System.out.print("Enter number of years: ");
23            input = sc.nextLine();
24            int years = Integer.parseInt(input);
25            --
26        }
```

The status bar at the bottom indicates 'Writable', 'Smart Insert', and '1:1:0'.

# The dialog box for opening a project in Eclipse





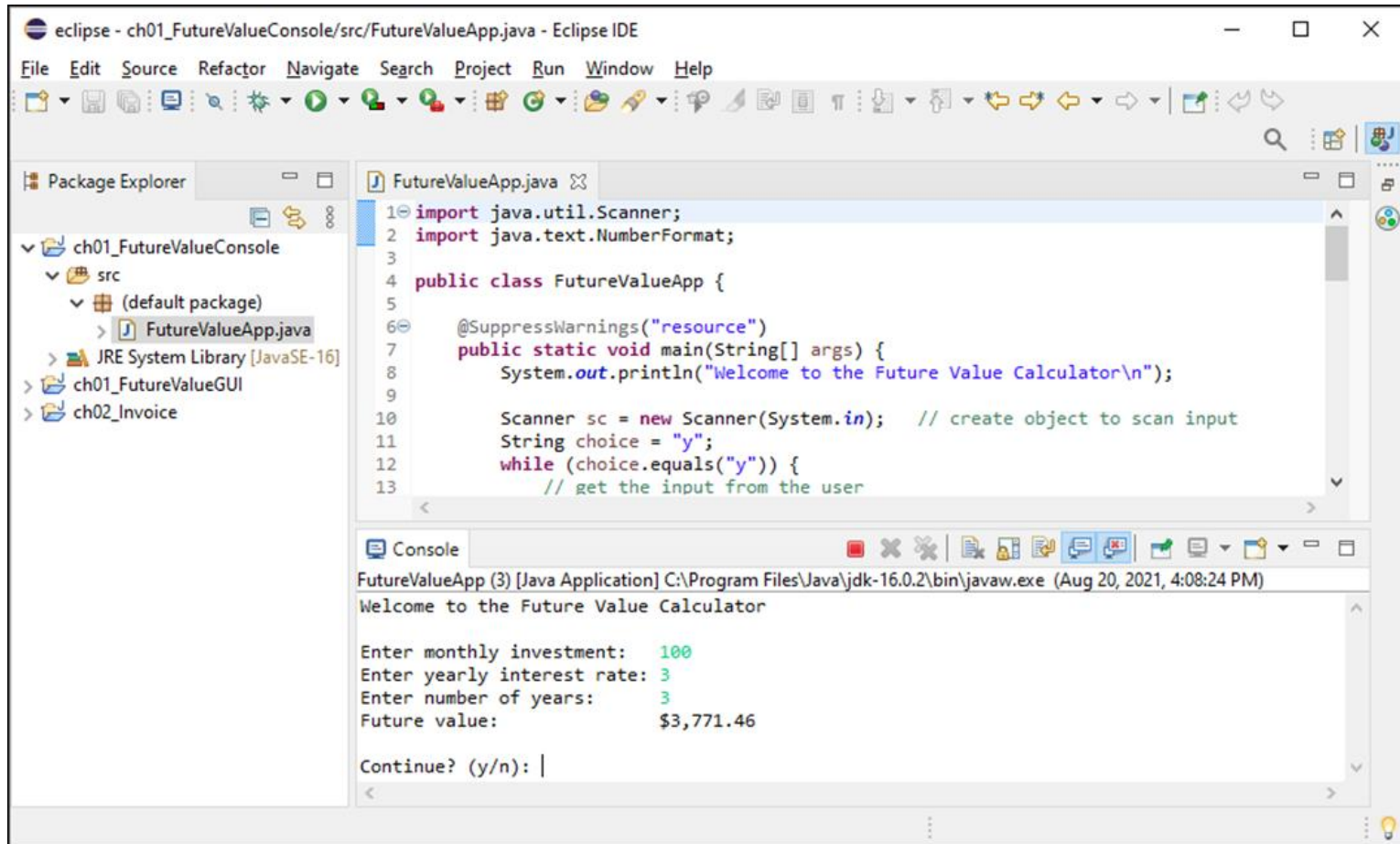
## How to open and close a project in Eclipse

- To open a project, select File→Open Projects from File System. In the dialog box, click the Directory button, navigate to the folder that contains the projects you want to import, select the projects, and click Finish.
- To close a project and remove it from the Package Explorer, right-click on the project in the Package Explorer window and select Delete. Then, make sure the “Delete project contents on disk” checkbox is unchecked and select OK.

### Note for macOS

- To enable right-clicking with macOS, you can edit the system preferences for the mouse. Or, if you prefer, you can control-click instead of right-clicking.

# Eclipse with three open projects and a Console window



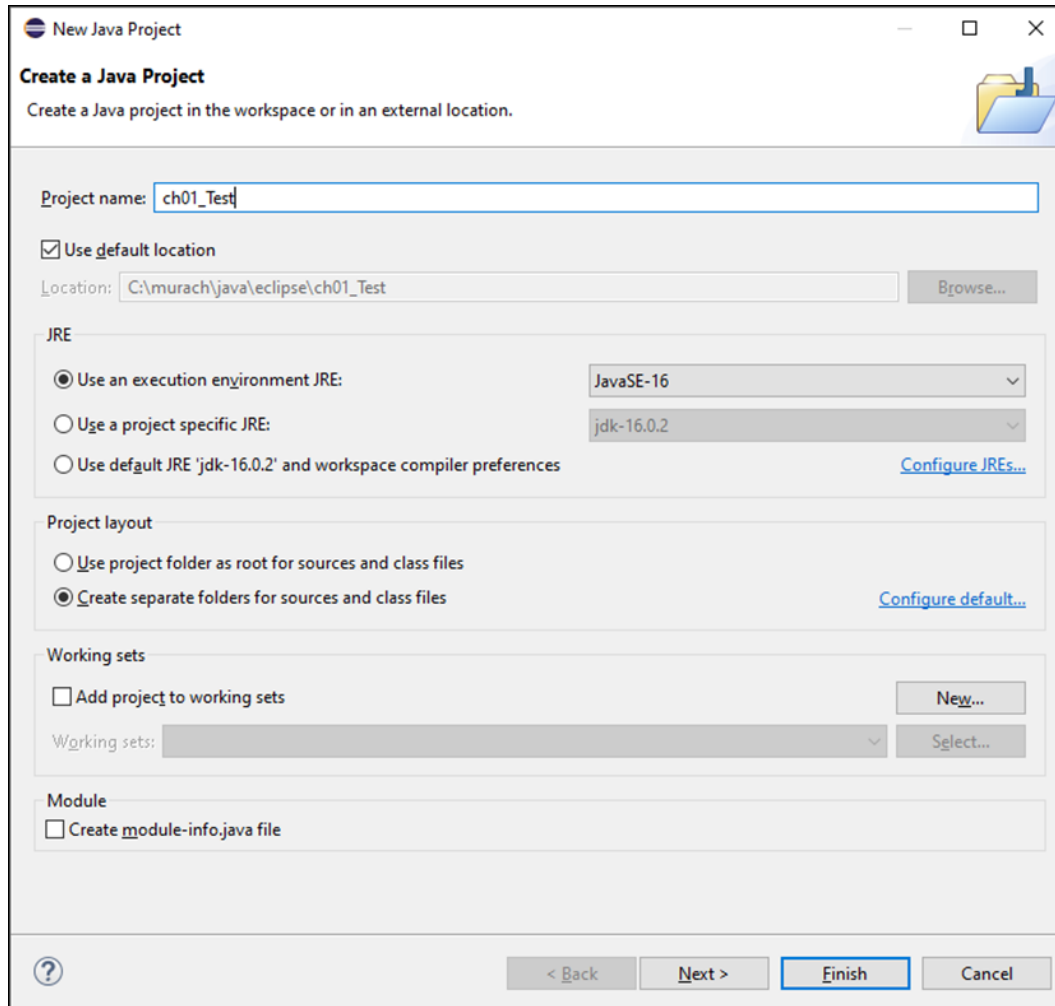
# How to compile and run a project with Eclipse

- If you open multiple projects, they all appear in the Package Explorer. To select a project, click on it in the Package Explorer.
- To run the selected project, click the Run button in the toolbar. When you run a project, Eclipse automatically compiles it.
- To delete all compiled files for a project and compile the project again, select the project in the Package Explorer. Then, select Project→Clean.

## How to work with the Console window

- When you run an application that prints data to the console, the Console window displays that data.
- If an application requests input from the console, the Console window pauses to accept the input. Then, you can click in the Console window, type the input, and press the Enter key.
- The Console window can also display error messages when you run an application.

# The dialog box for creating a new project



The screenshot shows the 'New Java Project' dialog box in the Eclipse IDE. The title bar reads 'New Java Project'. Below the title bar, the text 'Create a Java Project' is followed by the instruction 'Create a Java project in the workspace or in an external location.' and a folder icon.

The dialog is divided into several sections:

- Project name:** A text field containing 'ch01\_Test'.
- Use default location:** A checked checkbox. Below it, a text field shows the location 'C:\murach\java\eclipse\ch01\_Test' and a 'Browse...' button.
- JRE:** A section with three radio buttons:
  - ☒ Use an execution environment JRE: A dropdown menu showing 'JavaSE-16'.
  - ☐ Use a project specific JRE: A dropdown menu showing 'jdk-16.0.2'.
  - ☐ Use default JRE 'jdk-16.0.2' and workspace compiler preferences. A link 'Configure JREs...' is to the right.
- Project layout:** A section with two radio buttons:
  - ☐ Use project folder as root for sources and class files.
  - ☒ Create separate folders for sources and class files. A link 'Configure default...' is to the right.
- Working sets:** A section with a checkbox 'Add project to working sets' and a 'New...' button. Below it, a 'Working sets:' dropdown menu and a 'Select...' button.
- Module:** A section with a checkbox 'Create module-info.java file'.

At the bottom, there is a help icon (question mark), and four buttons: '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

## How to create a new project

1. Select File→New→Java Project.
2. Enter a name for the project.
3. Uncheck from the “Create module-info.java file” checkbox.
4. Click Finish.

# The dialog box for creating a new class

**New Java Class**

**Java Class**

⚠ The use of the default package is discouraged.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☒ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

## How to create a new class

1. In the Package Explorer, right-click on the project and select New→Class.
2. In the Package text box, delete the name for the package.
3. In the Name text box, enter the name of the class.
4. Select the first checkbox that adds a main() method to the class.
5. Click Finish.