Chapter 1

Introduction

Objectives

- To understand the meaning of Java language specification, API, JDK, and IDE.
- To create, compile, and run Java programs
- To use sound Java programming style and document programs properly
- To explain the differences between syntax errors, runtime errors, and logic errors
- To develop Java programs using Eclipse.

What is programming?

- *Programming* means to create (or develop) software, which is also called a *program.*
- In basic terms, software contains the instructions that tell a computer—or a computerized device—what to do.
- Software developers create software with the help of powerful tools called *programming languages.*

Programming Languages

- Computers do not understand human languages, so programs must be written in a language a computer can use.
- There are hundreds of programming languages, and they were developed to make the programming process easier for people.
- However, all programs must be converted into the instructions the computer can execute.
- There are many programming languages, this book teaches you how to create programs by using the Java programming language.
- There is no “best” language. Each one has its own strengths and weaknesses

Popular high level programming languages

TABLE 1.1 Popular High-Level Programming Languages

Language	*Description*
Ada	Named for Ada Lovelace, who worked on mechanical general-purpose computers. The Ada language was developed for the Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily by beginners.
C	Developed at Bell Laboratories. C combines the power of an assembly language with the ease of use and portability of a high-level language.
C++	C++ is an object-oriented language, based on C.
C#	Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft.
COBOL	COmmon Business Oriented Language. Used for business applications.
FORTRAN	FORmula TRANslation. Popular for scientific and mathematical applications.
Java	Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications.
Pascal	Named for Blaise Pascal, who pioneered calculating machines in the seventeenth century. It is a simple, structured, general-purpose language primarily for teaching programming.
Python	A simple general-purpose scripting language good for writing short programs.
Visual Basic	Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces.

Interpreter vs Compiler

- A program written in a high-level language is called a *source program* or *source code*.
- Because a computer cannot execute a source program, a source program must be translated into machine code for execution.
- The translation can be done using an *interpreter* or a *compiler*.
  - An interpreter reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away.

    Note that a statement from the source code may be translated into several machine instructions.
  - A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed.

Interpreter vs Compiler (cont'd)

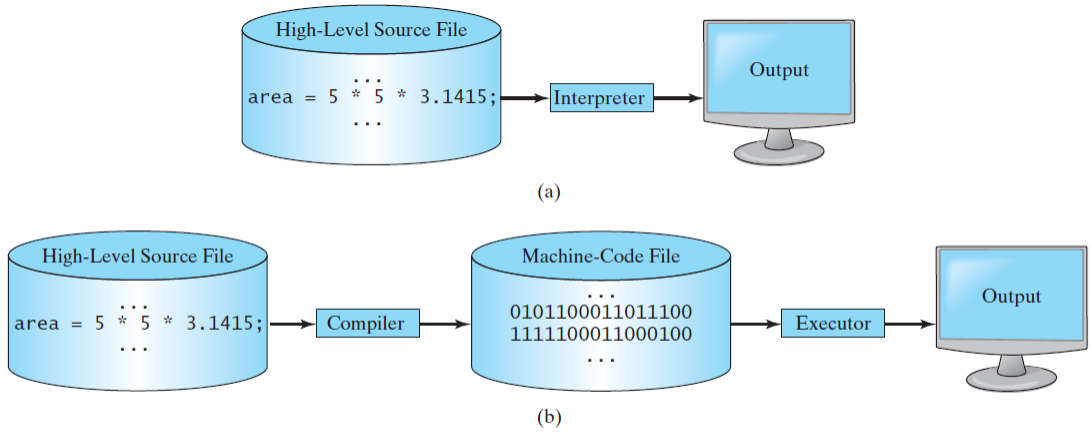


FIGURE 1.4 (a) An interpreter translates and executes a program one statement at a time. (b) A compiler translates the entire source program into a machine-language file for execution.

Java

- Java was developed by a team led by James Gosling at Sun Microsystems. Sun Microsystems was purchased by Oracle in 2010.
- As stated by its designer, Java is *simple*, *object oriented*, *distributed*, *interpreted*, *robust*, *secure*, *architecture neutral*, *portable*, *high performance*, *multithreaded*, and *dynamic.*
- Java is a versatile programming language: you can use it to develop applications for desktop computers, servers, and small handheld devices. The software for Android cell phones is developed using Java.

The Java Language Specification, API, JDK

- Java syntax is defined in the Java language specification. You can find the complete Java language specification at http://docs.oracle.com/javase/specs/
- The application program interface (API), also known as library, contains predefined classes and interfaces for developing Java programs.
- The JDK is the software for developing and running Java programs. Oracle releases each version with a *Java Development Toolkit (JDK).* https://www.oracle.com/technetwork/java/javase/downloads/index.html
- **You must first install and configure the JDK before you can compile and run programs.** Make sure you download the JDK and not the JRE (Java Runtime Environment).
  - JRE is Java Runtime Environment which is all you need as an end-user to run programs, but does not include the compiler and developer tools.
  - The JDK also includes the JRE, so it is not necessary to download both the JDK and JRE separately.

Java SE, EE, ME

- Java is a full-fledged and powerful language that can be used in many ways. It comes in three editions:
  - Java *Standard Edition (Java SE)* to develop client-side applications. The applications can run standalone or as applets running from a Web browser.
  - *Java Enterprise Edition (Java EE)* to develop server-side applications, such as Java servlets, JavaServer Pages (JSP), and JavaServer Faces (JSF).
  - *Java Micro Edition (Java ME)* to develop applications for mobile devices, such as cell phones.
- The textbook uses Java SE to introduce Java programming. There are many versions of Java SE (https://en.wikipedia.org/wiki/Java_version_history). The latest version is Java SE 12. The version in MPI lab in Java SE 8 (LTS) where LTS stands for Long Term Support (support up to 3 years for the next upgrade).

Version History

- The latest versions are Java 12, released in March 2019, and Java 11, a currently supported long-term support (LTS) version, released on September 25, 2018.
- Oracle released for the "legacy" Java 8 LTS the last free "public update" in January 2019 for commercial use, while it will otherwise still support Java 8 with public updates for personal use up to at least December 2020. (https://www.oracle.com/technetwork/java/java-se-support-roadmap.html)
- Oracle (and others) "highly recommend that you uninstall older versions of Java", because of serious risks due to unresolved security issues.
- Since Java 9 (and 10) is no longer supported, Oracle advises its users to "immediately transition" to Java 11 (Java 12 is also an non-LTS option).

A simple Java program

- Let's begin with a simple Java program that displays the message **Welcome to Java!**
- A Java program is executed from the **main** method in the class.

LISTING 1.1 Welcome.java

```
public class Welcome {
  public static void main(String[] args) {
    // Display message Welcome to Java! on the console
    System.out.println("Welcome to Java!");
  }
}
```

Every Java program has at least one class. Each class has a name.

A class may contain several methods. The **main** method is the entry point where the program begins execution.

Comments are not programming statements and are ignored by the compiler.

This statement displays the string **Welcome to Java!** on the console.
String is a programming term meaning a sequence of characters.

Some rules regarding the Java Program

- By convention, class names start with an uppercase letter.
- Java source programs are case sensitive. It would be wrong, for example, to replace **main** in the program with **Main.**
- Every statement in Java ends with a semicolon (**;**), known as the *statement terminator.*
- A string must be enclosed in double quotation marks.
- *Reserved words*, or *keywords*, have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word **class**, it understands that the word after **class** is the name for the class. Other reserved words in this program are **public**, **static**, and **void**.
- A *line comment is* preceded by two slashes (**//**), or enclosed between **/*** and ***/** on one or several lines, called a *block comment* or *paragraph comment*.

Class block, Method block

- A pair of curly braces in a program forms a *block* that groups the program's components. A block begins with an opening brace (**{**) and ends with a closing brace (**}**).
- Every class has a *class block* that groups the data and methods of the class.
- Similarly, every method has a *method block* that groups the statements in the method.
- Blocks can be *nested*, meaning that one block can be placed within another, as shown in the following code.

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Class block

Method block

Special characters

TABLE 1.2 Special Characters

Character	*Name*	*Description*
{}	Opening and closing braces	Denote a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denote an array.
//	Double slashes	Precede a comment line.
" "	Opening and closing quotation marks	Enclose a string (i.e., sequence of characters).
;	Semicolon	Mark the end of a statement.

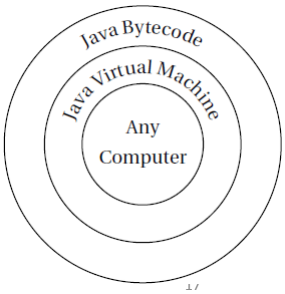
- If your program violates a rule—for example, if the semicolon is missing, a brace is missing, a quotation mark is missing, or a word is misspelled—the Java compiler will report syntax errors.

Creating and Compiling a Java Program

- *You save a Java program in a .java file and compile it into a .class file. The .class file is executed by the Java Virtual Machine.*
- If your program has compile errors, you have to modify the program to fix them, and then recompile it.
- If your program has runtime errors or does not produce the correct result, you have to modify the program, recompile it, and execute it again.
- The source file must end with the extension **.java** and must have the same exact name as the public class name. For example, the file for the source code in Listing 1.1 should be named **Welcome.java**, since the public class name is **Welcome**.
- If there aren't any syntax errors, the *compiler* generates a bytecode file with a **.class** extension. Thus, the preceding command generates a file named **Welcome.class**

Java Compilation

- Java source code is compiled into Java bytecode.
- Java bytecode is a low-level language. The *bytecode* is similar to machine instructions but is architecture neutral and can run on any platform that has a *Java Virtual Machine (JVM)*.
- Rather than a physical machine, the virtual machine is a program that interprets Java bytecode. This is one of Java's primary advantages: *Java bytecode can run on a variety of hardware platforms and operating systems.*
- Java source code is compiled into Java bytecode and Java bytecode is interpreted by the JVM. Your Java code may use the code in the Java library. The JVM executes your code along with the code in the library.



Executing a Java Program

- To execute a Java program is to run the program's bytecode.
- You can execute the bytecode on any platform with a JVM, which is an interpreter. It translates the individual instructions in the bytecode into the target machine language code one at a time rather than the whole program as a single unit.
- Each step is executed immediately after it is translated.

Some possible errors during compilation

- If you execute a class file that does not exist, a **NoClassDefFoundError** will occur.
- If you execute a class file that does not have a **main** method or you mistype the **main** method (e.g., by typing **Main** instead of **main**), a **NoSuchMethodError** will occur.

Compiling and Running a Java Program using command-line interface

To compile and run a program using the command line interface, we follow these steps:

1. Create your Java main class source file using a text editor, and save it to a file with extension ".java".
2. Change to the folder (using command "cd") containing the source file.
3. Locate your JDK folder path, for example, "C:\Program Files\Java\jdk1.8.0_211\bin".
4. Type the command to compile: *your-jdk-path*\javac YourClassName.java
5. If your program compiles with no errors, verify there's a binary ".class" file "YourClassName.class" in the same folder as your source file.
6. Type the command to run: java YourClassName

Setting JAVA_HOME Variable

- It is possible to run a program without specifying the location of executables like javac, java etc. like ***javac A.java*** instead of ***C:\Program Files\Java\jdk1.8.0_211\bin\javac A.java***
- To set the JAVA_HOME variable:
  1. Do one of the following:
     Windows 7 – Right click **My Computer** and select **Properties** > **Advanced**
     Windows 8 – Go to **Control Panel** > **System** > **Advanced System Settings**
     Windows 10 – Search for **Environment Variables** then select **Edit the system environment variables**
  2. Click the **Environment Variables** button.
  3. Under **System Variables**, click **New**. In the **Variable Name** field, enter "JAVA_HOME" and in the **Variable Value** field, enter your JDK path (e.g. ***C:\Program Files\Java\jdk1.8.0_211\bin)***.
  4. You need to close and re-open any command windows that were as there's no way to reload environment variables from an active command prompt. If the changes don't take effect after reopening the command window, restart Windows.

The javac command for compilation

- Figure 1.9 shows the **javac** command for compiling **Welcome.java**.
- The compiler generates the **Welcome.class** file, and this file is executed using the **java** command.

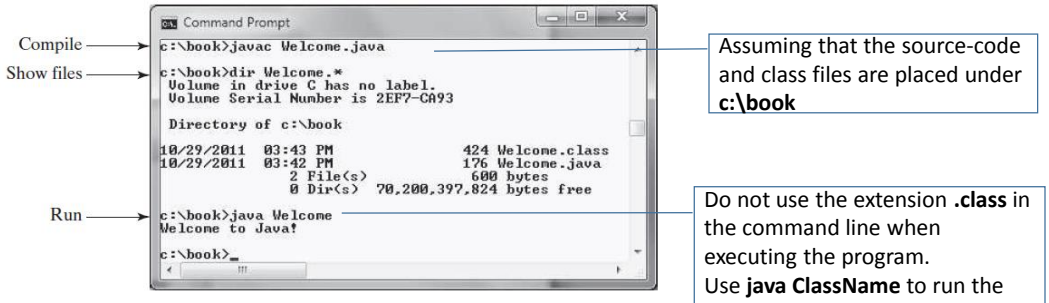


FIGURE 1.9 The output of Listing 1.1 displays the message "Welcome to Java!"

Programming style and documentation

- *Good programming style and proper documentation make a program easy to read and help programmers prevent errors.*
  - Appropriate Comments and Comment Styles
    - Include a summary at the beginning of the program that explains what the program does, its key features, and any unique techniques it uses.
  - Proper Indentation and Spacing
    - *Indentation* is used to illustrate the structural relationships between a program's components or statements. Java can read the program even if all of the statements are on the same line, but humans find it easier to read and maintain code that is aligned properly.
  - Block Styles
    - A *block* is a group of statements surrounded by braces. There are two popular styles, *next-line* style and *end-of-line* style.

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

Next-line style

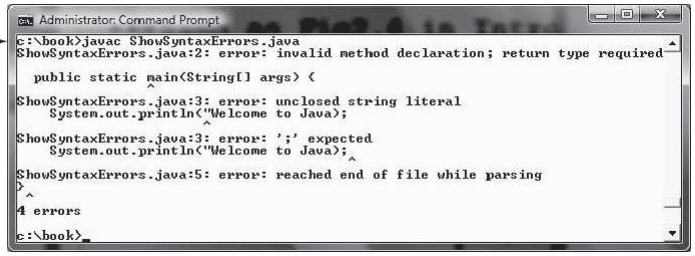
```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

End-of-line style

Programming Errors – Syntax errors

- *Programming errors can be categorized into three types: syntax errors, runtime errors, and logic errors.*
- Syntax errors:
  - Errors that are detected by the compiler are called *syntax errors* or *compile errors.*
  - Syntax errors result from errors in code construction, such as mistyping a keyword, omitting some necessary punctuation, or using an opening brace without a corresponding closing brace.

Compile →



Programming Errors – Runtime errors

- Runtime errors:
  - *Runtime errors* are errors that cause a program to terminate abnormally. They occur while a program is running if the environment detects an operation that is impossible to carry out.
  - Input mistakes typically cause runtime errors. For instance, if the program expects to read in a number, but instead the user enters a string, this causes data-type errors to occur in the program.

LISTING 1.5 ShowRuntimeErrors.java

```
1  public class ShowRuntimeErrors {
2    public static void main(String[] args) {
3      System.out.println(1 / 0);
4    }
5  }
```

Run →

FIGURE 1.11 The runtime error causes the program to terminate abnormally.

Programming Errors – Logic errors

- Logic errors:
  - *Logic errors* occur when a program does not perform the way it was intended to. Errors of this kind occur for many different reasons.
  - For example, suppose you wrote the program in Listing 1.6 to convert Celsius **35** degrees to a Fahrenheit degree:

LISTING 1.6 ShowLogicErrors.java

```
1  public class ShowLogicErrors {
2    public static void main(String[] args) {
3      System.out.println("Celsius 35 is Fahrenheit degree ");
4      System.out.println((9 / 5) * 35 + 32);
5    }
6  }
```

  - You will get Fahrenheit **67** degrees, which is wrong. It should be **95.0**.
  - In Java, the division for integers is the quotient—the fractional part is truncated—so in Java **9 / 5** is **1**.
  - To get the correct result, you need to use **9.0 / 5**, which results in **1.8**.

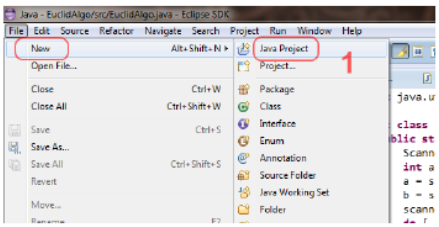
Eclipse: Developing Java Programs

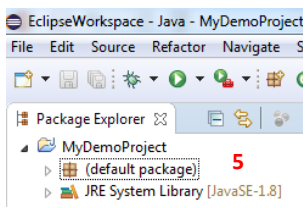
You can use a Java development tool (e.g., NetBeans, Eclipse, and TextPad)—software that provides an *integrated development environment (IDE)* for developing Java programs quickly.

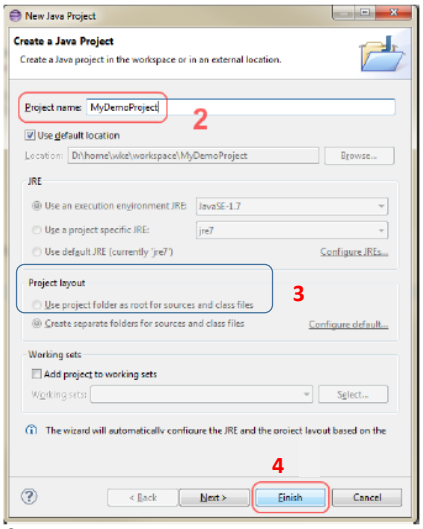
To start with Eclipse, create a project to hold all the files with the steps below:

1. Choose *File→ New → Java Project* to display the New Project wizard.
2. Type **MyDemoProject** in the Project name field. As you type, the Location field is automatically set by default. You may customize the location for your project.
3. Configure Project Layout:
   - By default, Eclipse stores certain project files in the project folder, your Java source code in the project's src sub-folder, and the class files produced by the Java compiler in the project's bin sub-folder. This organization benefits large complex projects but complicates smaller projects.
   - Eclipse can be configured to, by default, create a *new* project with all its files in the project folder. Select the option *Use project folder as root for sources and class files* so that the .java and .class files are in the same folder for easy access.
4. Click *Finish* to create the project.

Eclipse: Developing Java Programs





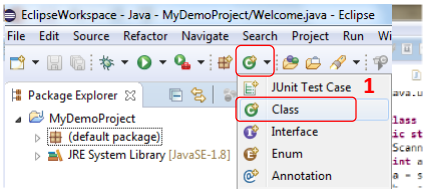


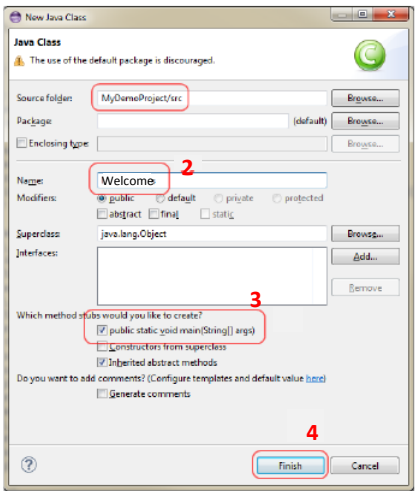
Eclipse: Creating a Java Class

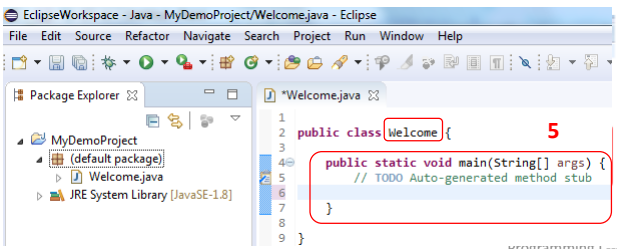
After a project is created, you can create Java programs in the project using the following steps:

1. Choose *File→ New → Class* to display the New Java Class wizard.
2. Type **Welcome** in the Name field.
3. Check the option *public static void main(String[] args)*.
4. Click *Finish* to generate the template for the source code Welcome.java.
5. Now, you can import necessary classes before the main class, and write statements in the main method.

Eclipse: Creating a Java Class

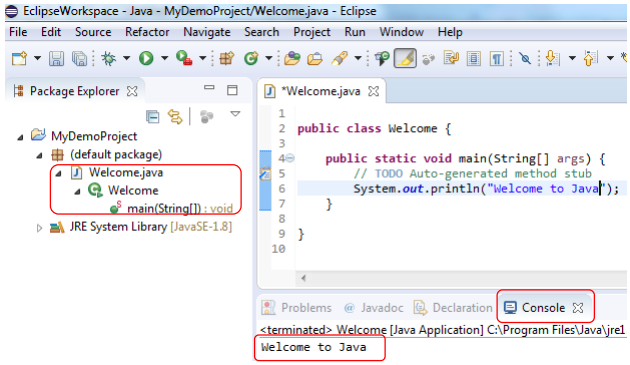






Eclipse: Compiling and Running a Class

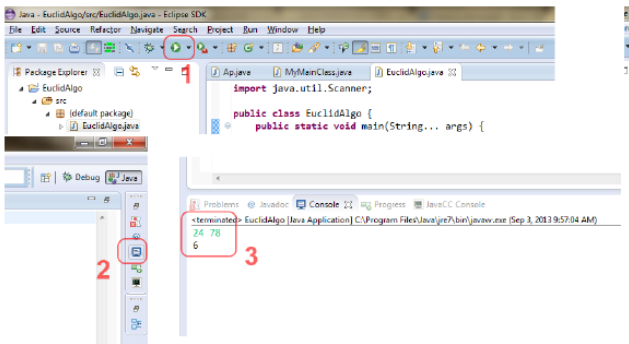
- When you save your program source file, the program is compiled automatically by Eclipse, and the binary ".class" file is created.
- To run the program, right-click the class in the project to display a context menu. Choose *Run, Java Application* in the context menu to run the class. The output is displayed in the Console pane.

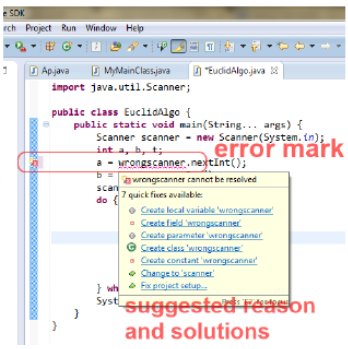


- Eclipse allows you to keep more than one programs in a project.
- To run a particular program, open and right-click on the source file ⇒ Run As ⇒ Java Application.
- Clicking the "Run" button (with a "Play" icon) runs the recently-run program (based on the previous configuration).

Eclipse: **Interacting with the Console Window**

- If the program has no errors (no red marks), we can run the program and supply input and see output from the console window.
- In some cases, Eclipse shows a ORANGE LIGHT-BULB (for HINTS) next to the ERROR RED-CROSS. You can click on the LIGHT-BULB to get a list of HINTS to resolve this particular error, which may or may not work!
- SYNTAX WARNING: marked by a orange triangular exclamation sign. Unlike errors, warnings may or may not cause problems. Try to fix these warnings as well. But you can RUN your program with warnings.





Eclipse Workspace: closing projects

- A Workspace is a folder on your hard-drive where Eclipse stores your java projects. Java code is organized in projects in Eclipse.
- An eclipse workspace can contain any number of projects. A project can be either in the open state or closed state.
- You will see your project in Package Explorer on the left of your screen.
- If a project is not under active development it can be closed, but they still reside on the local file system. Closed projects require less memory. Also, since they are not examined during builds, closing a project can improve build time. To close a project, from the Project select the Close Project menu item.
- A closed project is visible in the Package Explorer view but its contents cannot be edited using the Eclipse user interface.
- To reopen a closed project, in the Package Explorer view, select the closed project and click on the Project menu and select Open Project.

Eclipse Workspace: perspectives

- An eclipse window can have multiple perspectives open in it but only one perspective can be active at any point of time.
- A user can switch between open perspectives or open a new perspective. A perspective controls what appears in some menus and tool bars.
- An eclipse perspective is the name given to an initial collection and arrangement of views and an editor area.
- The default perspective is called java. Another useful perspective is the debug perspective. (Windows→ Perspective → Open Perspective).
- To stop running an application:
  - Open the Debug perspective
  - Select process in Devices list (bottom right)
  - Hit **Stop** button (top right of Devices pane)

Eclipse Workspace: deleting projects

There are two options to delete a project via Edit → Delete:

- To delete a project and remove its contents from the file system
- To delete a project from the workspace without removing its contents from the file system

Eclipse: Export function

Saving (exporting) a project in a different location for later use.

- If you work in the lab you will want to save your work to a memory stick. When you logout of the lab computers all your work is erased from the machine.
- To export an entire project right click on the project and select Export… → General → File System and then press the **Next >** button to browse to the location you want to save the project to.

Eclipse: open an existing project

- In Eclipse, to open an existing project which is copied from another source, there are two options:
  - Use File→ Open Projects From File System
    This works for Eclipse projects with .project descriptor
  - Use its **Import** function
    File→ Import… → General → Existing projects into Workspace, then select the radio option: *Select root directory* and specify the project's directory path.
    - Import is for projects that you have created either using Eclipse or other compatible IDEs, and that have a project file associated to them.
    - You can use import to copy an existing project to a new project folder inside the workspace directory.
  - It is common mistake to choose File → Open File… ✘✘
- **Adding Existing Classes to A Project:**
  File→ Import… → General → File System

What software do you need?

- **Java SE Development Kit 8u211**
  https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - How to know whether jdk is installed in my computer or not?
    - Go to Control Panel-->Program and Features and **check** if **Java /JDK** is listed there.
    - Open command prompt and type **java** -**version**. If you get the **version** info, **Java** is installed correctly and PATH is also set correctly.
- To use Eclipse for Java programming, you need to first install Java Development Kit (JDK).
- Download Eclipse Neon from
  https://www.eclipse.org/downloads/packages/release/neon/3/eclipse-ide-java-developers

Note:
- For the latest Java SE, see https://www.oracle.com/technetwork/java/javase/downloads/index.html
- For the latest Eclipse version, see https://www.eclipse.org/downloads/

Supplementary notes on using Eclipse

- **Intelli-Sense (ctrl-space):** You can use ctrl-space to activate the "intelli-sense" (or content assist). That is, Eclipse will offer you the choices, while you are typing.
- **Source Formatting (ctrl-shift-f):** Right-click on the source. Choose "Source" ⇒ "Format" to let Eclipse to layout your source codes with the proper indentation
- **Source Toggle Comment (ctrl-/):** To comment/uncomment a block of codes, choose "Source" ⇒ "Toggle Comment".
- **Refactor (or Rename) (alt-shift-r):** You can rename a variable, method, class, package or even the project easily in Eclipse. Select and right-click on the entity to be renamed ⇒ "Refactor" ⇒ "Rename". Eclipse can rename all the occurrences of the entity.

Supplementary notes on using Eclipse

- **Line Numbers:** To show the line numbers, choose "Window" menu ⇒ "Preferences" ⇒ "General" ⇒ "Editors" ⇒ "Text Editors" ⇒ Check the "Show Line Numbers" Box. You can also configure many editor options, such as the number of spaces for tab. Alternatively, you can right-click on the left-margin, and check "Show Line Numbers".
- **Changing Font Type and Size:** From "Window" menu ⇒ "Preferences" ⇒ "General" ⇒ "Appearance" ⇒ "Colors and Fonts" ⇒ expand "Java" ⇒ "Java Editor Text Font" ⇒ "Edit". (Alternatively, in "Window" ⇒ "Preferences" ⇒ type "font" as filter text and choose the appropriate entry.)
- **Error Message Hyperlink:** Click on an error message will hyperlink to the corresponding source statement.
- **Mouse Hover-over:** In debug mode, you could configure to show the variable's value when the mouse hovers over the variable. Select "Window" menu ⇒ "Preferences" ⇒ "Java" ⇒ "Editor" ⇒ "Hover".

Supplementary notes on using Eclipse

- **Spell Check:** To enable spell check, select Window ⇒ Preferences ⇒ type "spell" in the filter ⇒ General ⇒ Editors ⇒ Text Editors ⇒ Spelling ⇒ Check "Enable spell checking". Also provide a "User defined dictionary" (with an initially empty text file). To correct mis-spell words, right-click and press ctrl-1 (or Edit menu ⇒ Quick Fix).
- **Viewing two files in split screen:** Simply click and hold on the title of one file and drag it to the lower side of the screen. [To view the same file on split screen, create a new editor window by selecting Window ⇒ New Editor; and drag one window to the lower side of the screen.]
- **Package Explorer vs. Navigator:** We usually use "Package Explorer" in programming, but it will not show you all the folders and files under the project. On the other hand, "Navigator" is a file manager that shows the exact file structure of the project (similar to Windows Explorer). You can enable the Navigator by "Window" ⇒ Show view ⇒ Navigator.

Chapter Summary

- Computer *programs*, known as *software*, are the invisible instructions that control the hardware and make it perform tasks.
- Computer *programming* is the writing of instructions (i.e., code) for computers to perform.
- *High-level languages* are English-like and easy to learn and program. A program written in a high-level language is called a *source program.*
- A *compiler* is a software program that translates the source program into a *machine language program.*
- Java is platform independent, meaning that you can write a program once and run it on any computer.
- The Java source file name must match the public class name in the program. Java source code files must end with the **.java** extension.

Chapter Summary

- Every class is compiled into a separate bytecode file that has the same name as the class and ends with the **.class** extension.
- To compile a Java source-code file from the command line, use the **javac** command.
- To run a Java class from the command line, use the **java** command.
- Every Java program is a set of class definitions. The keyword **class** introduces a class definition. The contents of the class are included in a *block.*
- A block begins with an opening brace (**{**) and ends with a closing brace (**}**).
- Methods are contained in a class. To run a Java program, the program must have a **main** method. The **main** method is the entry point where the program starts when it is executed.

Chapter Summary

- Every *statement* in Java ends with a semicolon (**;**), known as the *statement terminator.*
- *Reserved words,* or *keywords,* have a specific meaning to the compiler and cannot be used for other purposes in the program.
- Java source programs are case sensitive.
- Programming errors can be categorized into three types: *syntax errors*, *runtime errors*, and *logic errors.*
- Errors reported by a compiler are called syntax errors or *compile errors.*
- Runtime errors are errors that cause a program to terminate abnormally.
- Logic errors occur when a program does not perform the way it was intended to.

Ideas for further practice

- Write a program that displays the area and perimeter of a rectangle with the width of **4.5** and height of **7.9.**
- (*Compute expressions*) Write a program that displays the result of

$$\frac{9.5 \times 4.5 - 2.5 \times 3}{45.5 - 3.5}$$