Fundamentals of Java

Session: 2

Application Development in Java

Objectives

- Explain the structure of a Java class
- List and explain steps to write a Java program
- Identify the benefits of NetBeans IDE
- Describe the various elements of NetBeans IDE
- Explain the steps to develop, compile, and execute Java program using NetBeans IDE
- Explain the various components of JVM
- Describe comments in Java

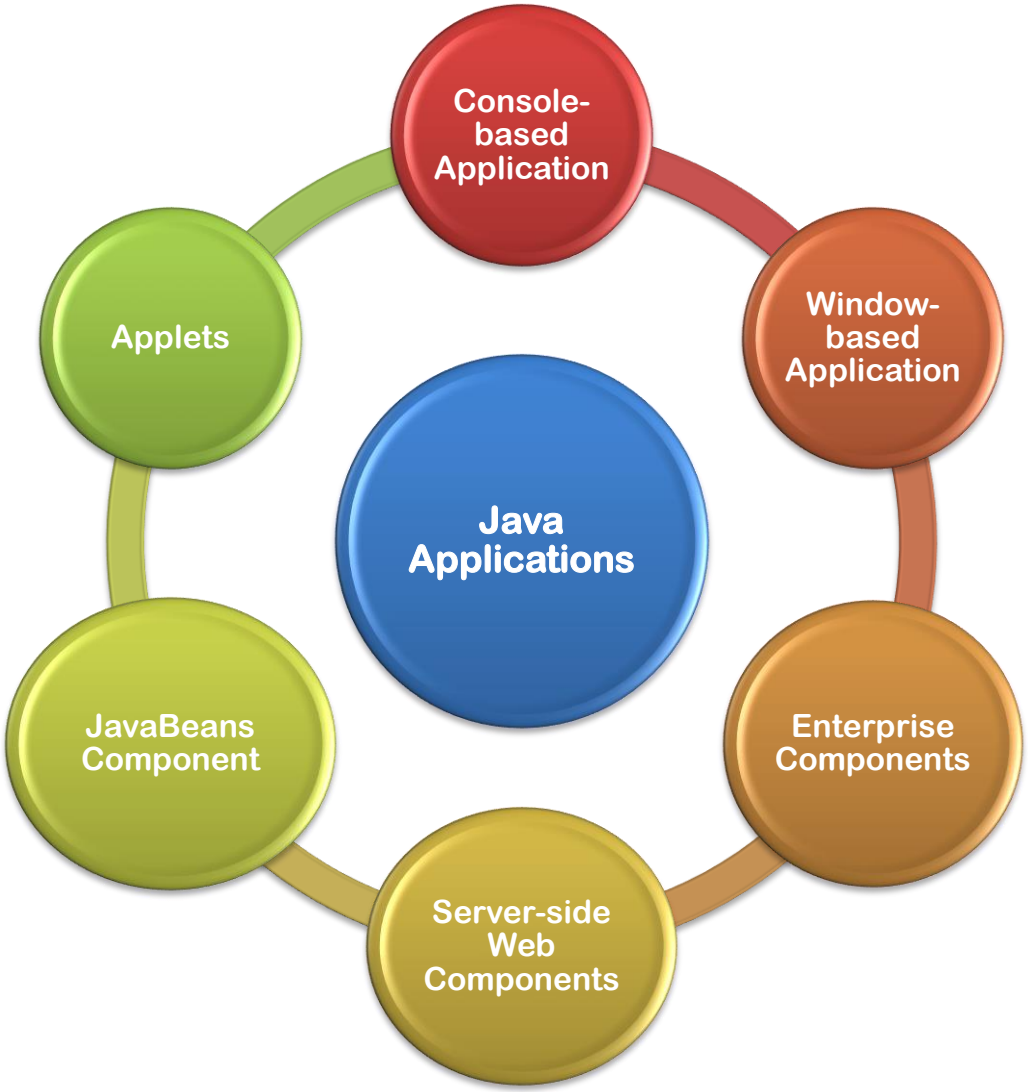
Introduction 1-2

- Java is a popular OOP language that supports developing applications for different requirements and domain areas.

- All types of applications can be developed:
  - In a simple text editor, such as **Notepad**.
  - In an environment that provides necessary tools to develop a Java application.
  - The environment is called as Integrated Development Environment (IDE).

Introduction 2-2

- Following figure shows the type of applications developed using Java



Structure of a Java Class 1-5

- The Java programming language is designed around object-oriented features and begins with a class design.
- The class represents a template for the objects created or instantiated by the Java runtime environment.
- The definition of the class is written in a file and is saved with a `.java` extension.
- Following figure shows the basic structure of a Java class.

```
package <package_name>;

import <other_packages>;

public class ClassName {
    <variables(also known as fields)>;

    <constructor method(s)>;

    <other methods>;
}
```

Structure of a Java Class 2-5

package

```
package <package_name>;
```

- Defines a namespace that stores classes with similar functionalities in them.
- The `package` keyword identifies:
  - Name of the package to which the class belongs.
  - Visibility of the class within the package and outside the package.
- The concept of package is similar to folder in the OS.
- In Java, all classes belongs to a package. If the package statement is not specified, then the class belongs to the default package.
- Example: All the user interface classes are grouped in the `java.awt` or `java.swing` packages.

Structure of a Java Class 3-5

import

```
import <other_packages>;
```

- Identifies the classes and packages that are used in a Java class.
- Helps to narrow down the search performed by the Java compiler by informing it about the classes and packages.
- Mandatory to import the required classes, before they are used in the Java program.
- Some exceptions wherein the use of **import** statement is not required are as follows:
  - If classes are present in the `java.lang` package.
  - If classes are located in the same package.
  - If classes are declared and used along with their package name. For example, `java.text.NumberFormat nf = new java.text. NumberFormat();`

Structure of a Java Class 4-5

class

```
public class ClassName {
```

- `class` keyword identifies a Java class.
- Precedes the name of the class in the declaration.
- `public` keyword indicates the access modifier that decides the visibility of the class.
- Name of a class and file name should match.

Variables

```
<variables(also known as fields)>;
```

- Are also referred to as instance fields or instance variables.
- Represent the state of objects.

Structure of a Java Class 5-5

Methods

```
<other methods>;
```

- Are functions that represent some action to be performed on an object.
- Are also referred to as instance methods.

Constructors

```
<constructor method(s)>;
```

- Are also methods or functions that are invoked during the creation of an object.
- Used to initialize the objects.

Developing a Java Program on Windows

- Basic requirements to write a Java program are as follows:
  - The **JDK 7** installed and configured on the system.
  - A text editor, such as **Notepad**.

- To create, compile, and execute a Java program, perform the following steps:

Create a Java Program 1-4

- Following code snippet demonstrates a simple Java program:

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

- `class` is a keyword and **`HelloWorld`** is the name of the class.
- The entire class definition and its members must be written within the opening and closing curly braces `{ }`.
- The area between the braces is known as the class body and contains the code for that class.

Create a Java Program 2-4

- Following code snippet demonstrates a simple Java program:

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

- `main()` - method is the entry point for a java-based console application.
- `public` - Is a keyword that enables the JVM to access the `main()` method.
- `static` – Is a keyword that allows a method to be called from outside a class without creating an instance of the class.
- `void` – Is a keyword that represents the data type of the value returned by the `main()` method. It informs the compiler that the method will not return any value.
- `args` - Is an array of type `String` and stores command line arguments. `String` is a class in Java and stores group of characters.

Create a Java Program 3-4

- Following code snippet demonstrates a simple Java program:

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

- `System.out.println()` statement displays the string that is passed as an argument.
- `System` is the predefined class and provides access to the system resources, such as console.
- `out` is the output stream connected to the console.
- `println()` is the built-in method of the output stream that is used to display a string.

Create a Java Program 4-4

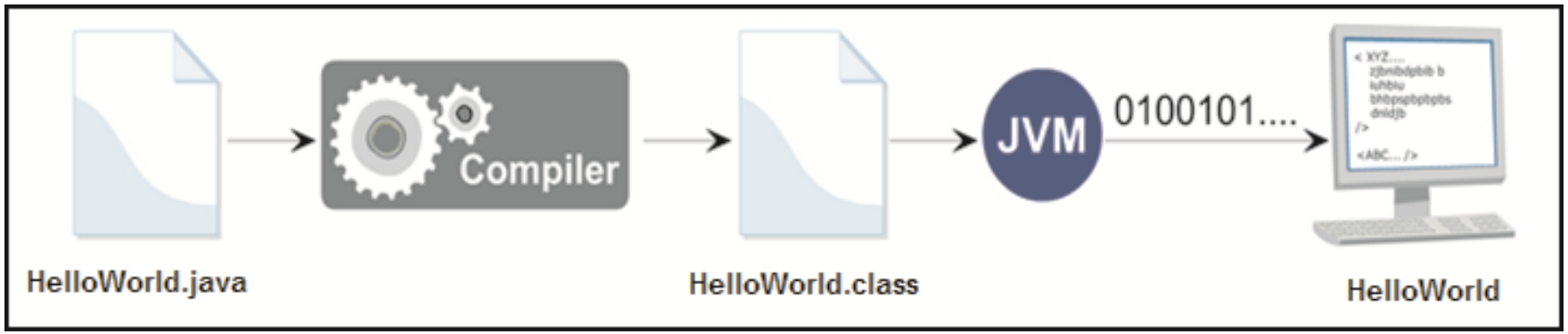
- Following code snippet demonstrates a simple Java program:

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

- Save the file as **`HelloWorld.java`**.
- The file name is same as class name, as the compilation of a Java code results in a class. Hence, the class name and the file name should be same.

Compile .java File 1-4

- Following figure shows the compilation process of the Java program.



- The **`HelloWorld.java`** file is known as source code file.
- It is compiled by invoking tool named `javac.exe`, which compiles the source code into a `.class` file.
- The `.class` file contains the bytecode which is interpreted by `java.exe` tool.
- `java.exe` interprets the bytecode and runs the program.

Compile .java File 2-4

- The syntax to use the `javac.exe` command:

Syntax

```
javac [option] source
```

where,

`source` - Is one or more file names that end with a `.java` extension.

Compile .java File 3-4

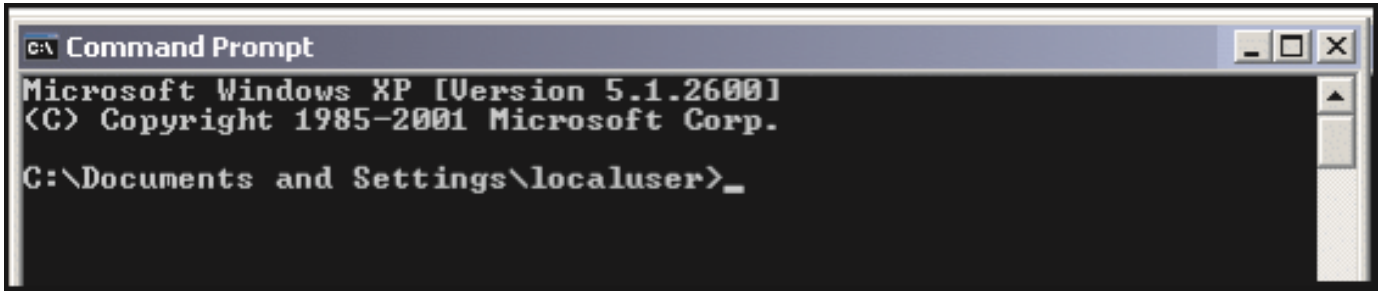
- Following table lists some of the options that can be used with the `javac` command:

Option	Description
-classpath	Specifies the location for the imported classes (overrides the CLASSPATH environment variable)
-d	Specifies the destination directory for the generated class files
-g	Prints all debugging information instead of the default line number and file name
-verbose	Generates message while the class is being compiled
-version	Displays version information
Sourcepath	Specifies the location of the input source file
-help	Prints a synopsis of standard options

- For example, `javac -d c:\` **`HelloWorld.java`** will create and save **`HelloWorld.class`** file in the `C:\` drive.

Compile .java File 4-4

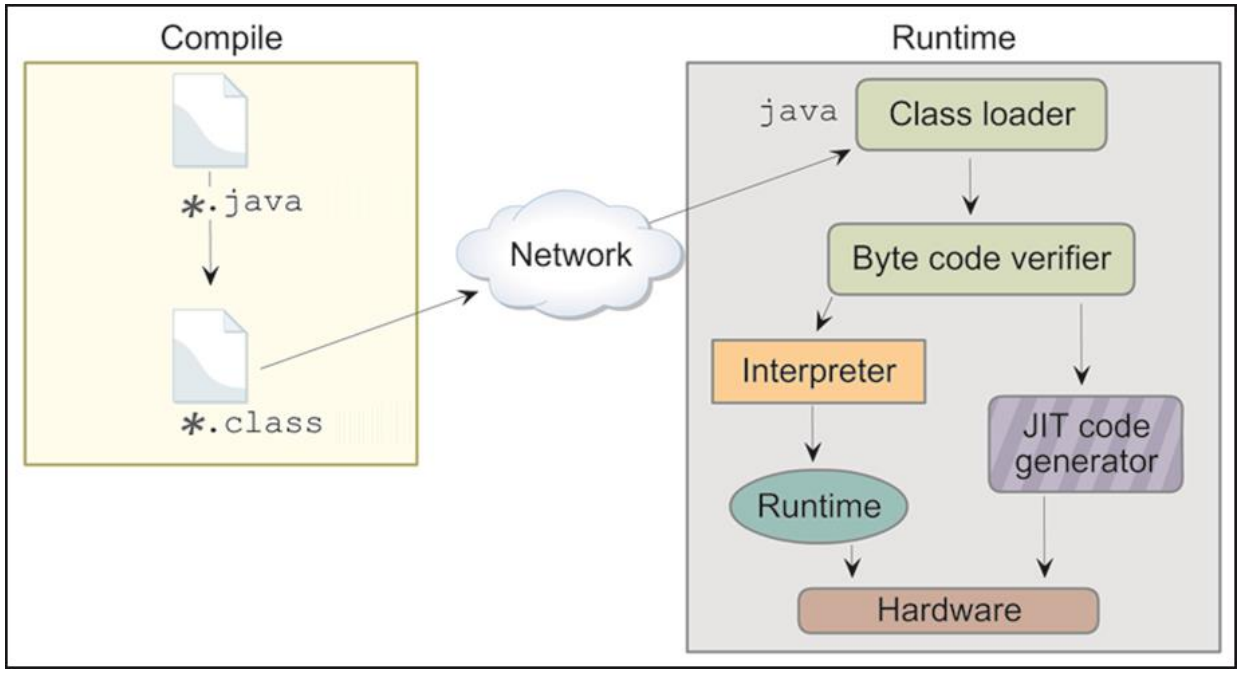
- To compile the **`HelloWorld.java`** program from the Windows platform, the user can:
  - Click **Start** menu.
  - Choose **Run**.
  - Enter the **cmd** command to display the **Command Prompt** window.
  - Following figure shows the **Command Prompt** window:



  - Set the drive and directory path to the directory containing `.java` file. For example, **`cd H:\Java`**.
  - Type the command, **`javac HelloWorld.java`** and press **Enter**.

Build and Execute Java Program 1-4

- The JVM is at the heart of the Java programming language.
- It is responsible for executing the `.class` file or bytecode file.
- The `.class` file can be executed on any computer or device, that has the JVM implemented on it.
- Following figure shows the components of JVM involved in the execution of the compiled bytecode.



Build and Execute Java Program 2-4

The **class loader** component of JVM loads all the necessary classes from the runtime libraries required for execution of the compiled bytecode.

The **bytecode verifier** then checks the code to ensure that it adheres to the JVM specification.

The bytecode is executed by the **interpreter**.

To boost the speed of execution, in Java version 2.0, a Hot Spot **Just-in-Time** (JIT) compiler was included at runtime.

During execution, the **JIT** compiler compiles some of the code into native code or platform-specific code to boosts the performance.

Build and Execute Java Program 3-4

- The Java interpreter command, `java` is used to interpret and run the Java bytecode.
- The syntax to use the `java.exe` command is as follows:

Syntax

```
java [option] classname [arguments]]
```

where,

`classname`: Is the name of the class file.

`arguments`: Is the arguments passed to the main function.

- To execute the **`HelloWorld`** class, type the command, `java` **`HelloWorld`** and press **Enter**.

Build and Execute Java Program 4-4

- Following table lists some of the options that can be used with the `java` command:

Option	Description
classpath	Specifies the location for the imported classes (overrides the CLASSPATH environment variable)
-v or –verbose	Produces additional output about each class loaded and each source file compiled
-version	Displays version information and exits
-jar	Uses a JAR file name instead of a class name
-help	Displays information about help and exits
-X	Displays information about non-standard options and exits

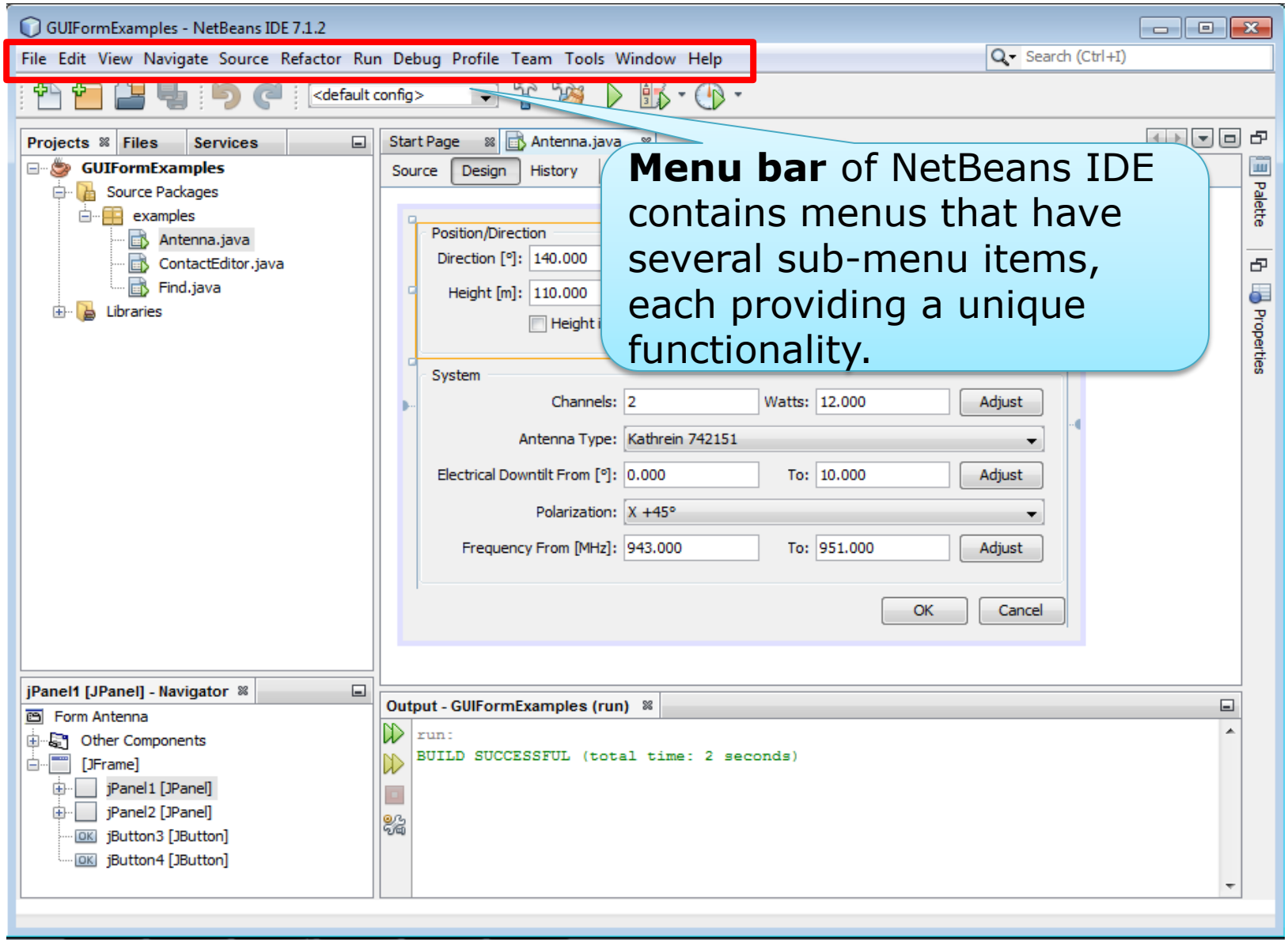
NetBeans IDE 1-7

- It is an open-source integrated development environment written purely in Java.
- It is a free and robust IDE that helps developers to create cross-platform desktop, Web, and mobile applications using Java.
- It contains features such as code completions, code template, and fix import for faster development.
- Some of its benefits are as follows:
  - Provides plug-in modules and supports rich client applications.
  - Provides graphical user interface for building, compiling, debugging, and packaging of applications.
  - Provides simple and user-friendly IDE configuration.

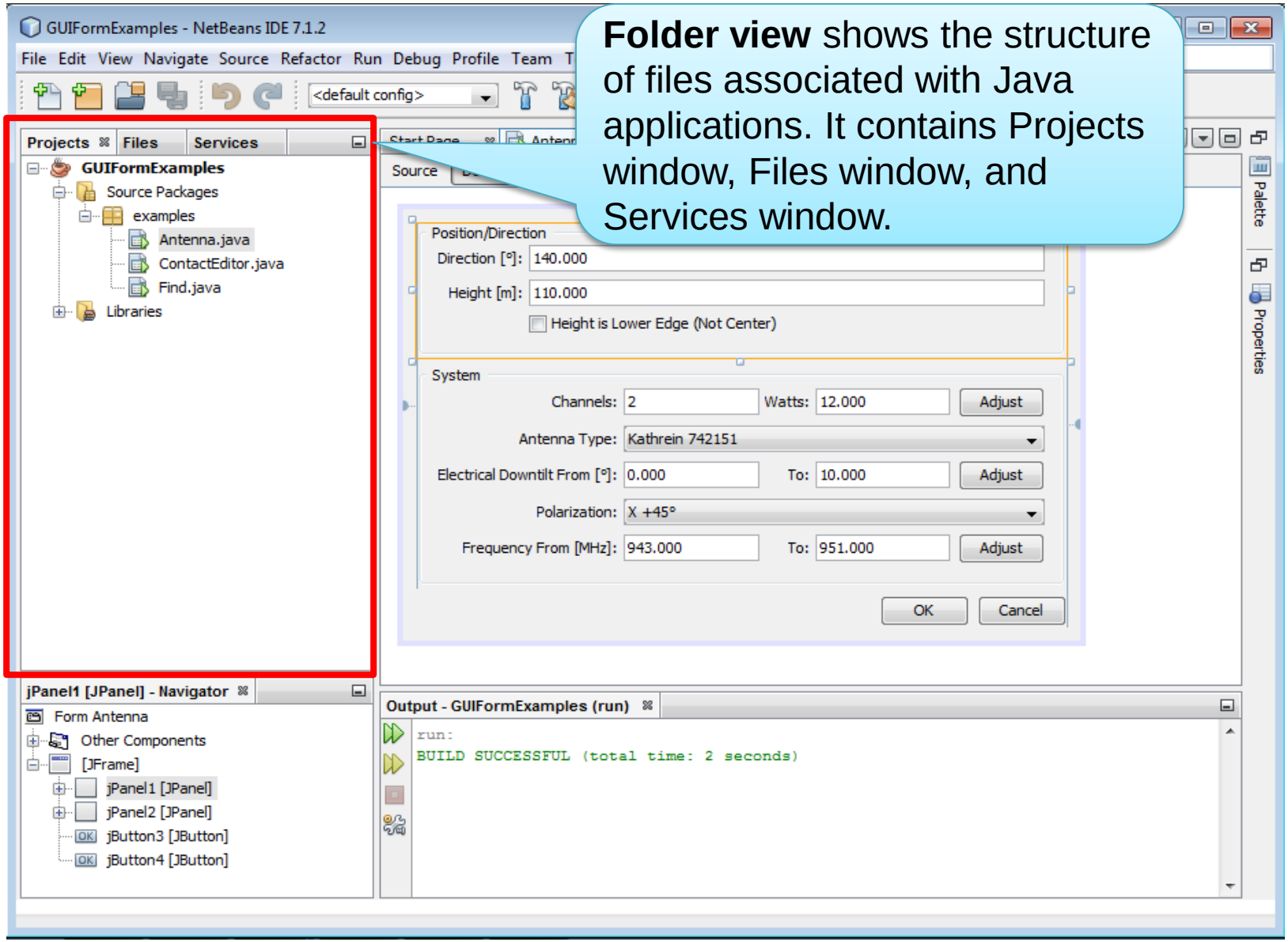
NetBeans IDE 2-7

- The NetBeans IDE has the following elements and views:
  - Menu Bar
  - Folders View
  - Components View
  - Coding and Design View
  - Output View

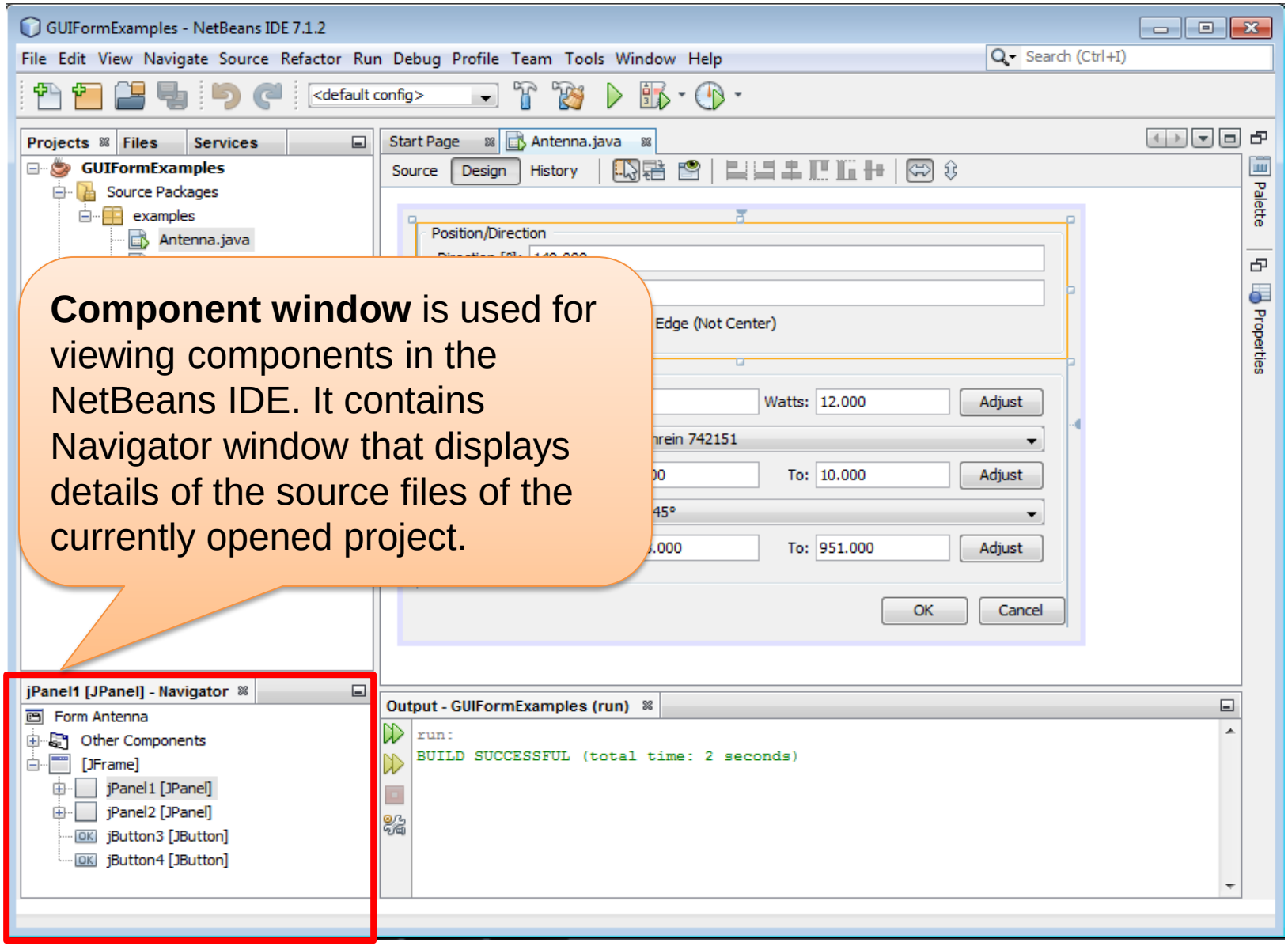
NetBeans IDE 3-7



NetBeans IDE 4-7

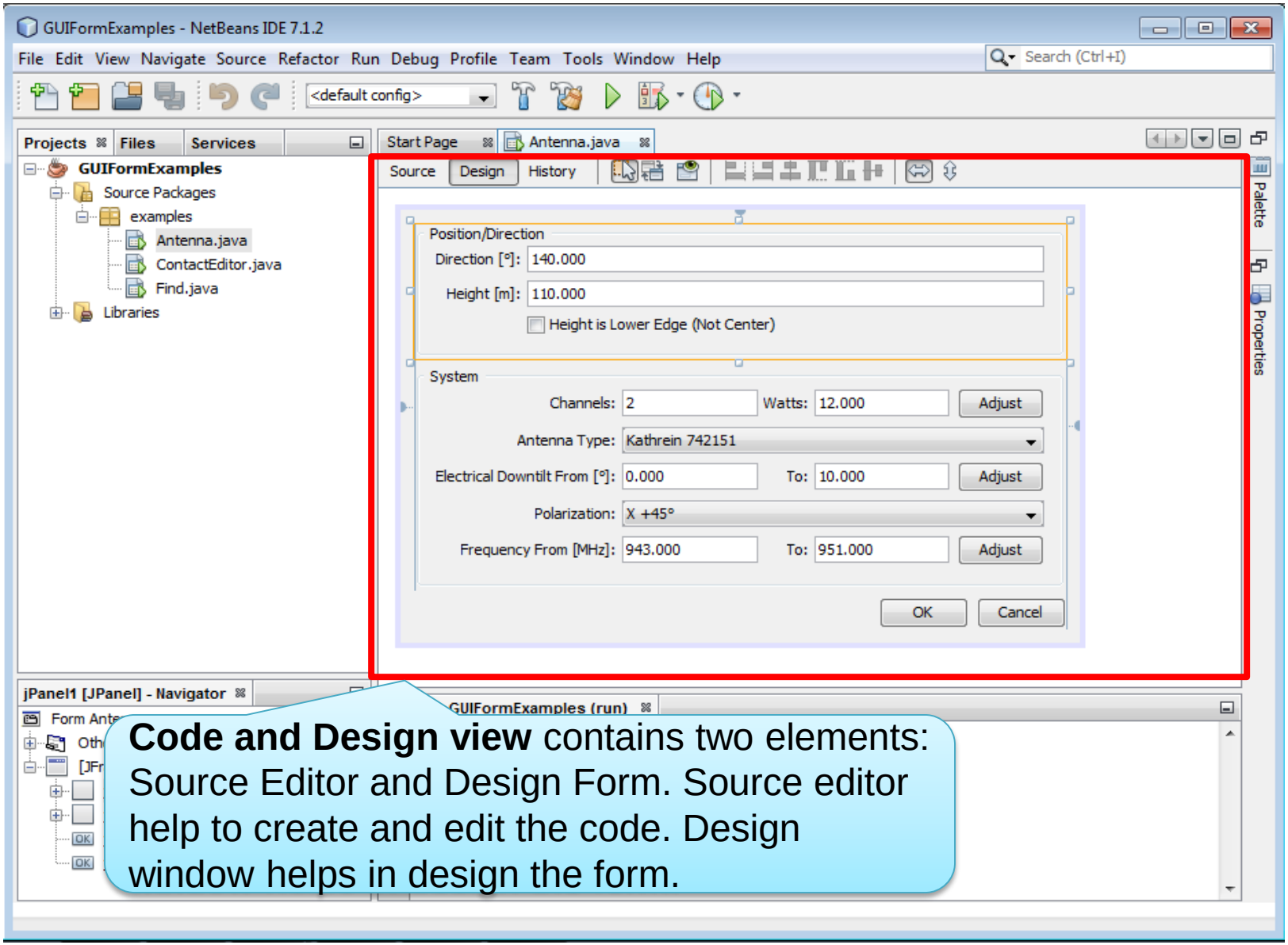


NetBeans IDE 5-7

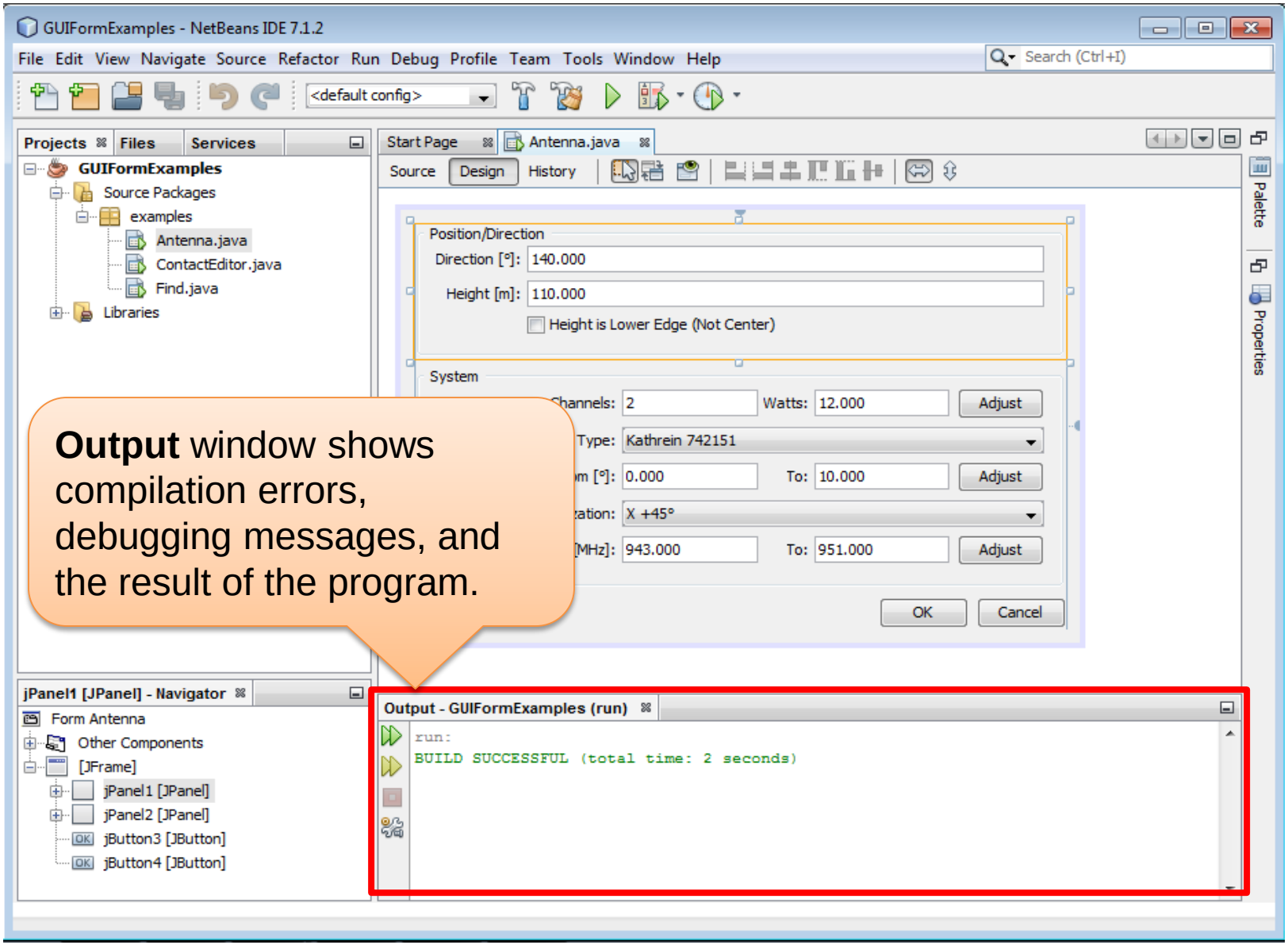


Component window is used for viewing components in the NetBeans IDE. It contains Navigator window that displays details of the source files of the currently opened project.

NetBeans IDE 6-7



NetBeans IDE 7-7



Download and Install NetBeans IDE 1-5

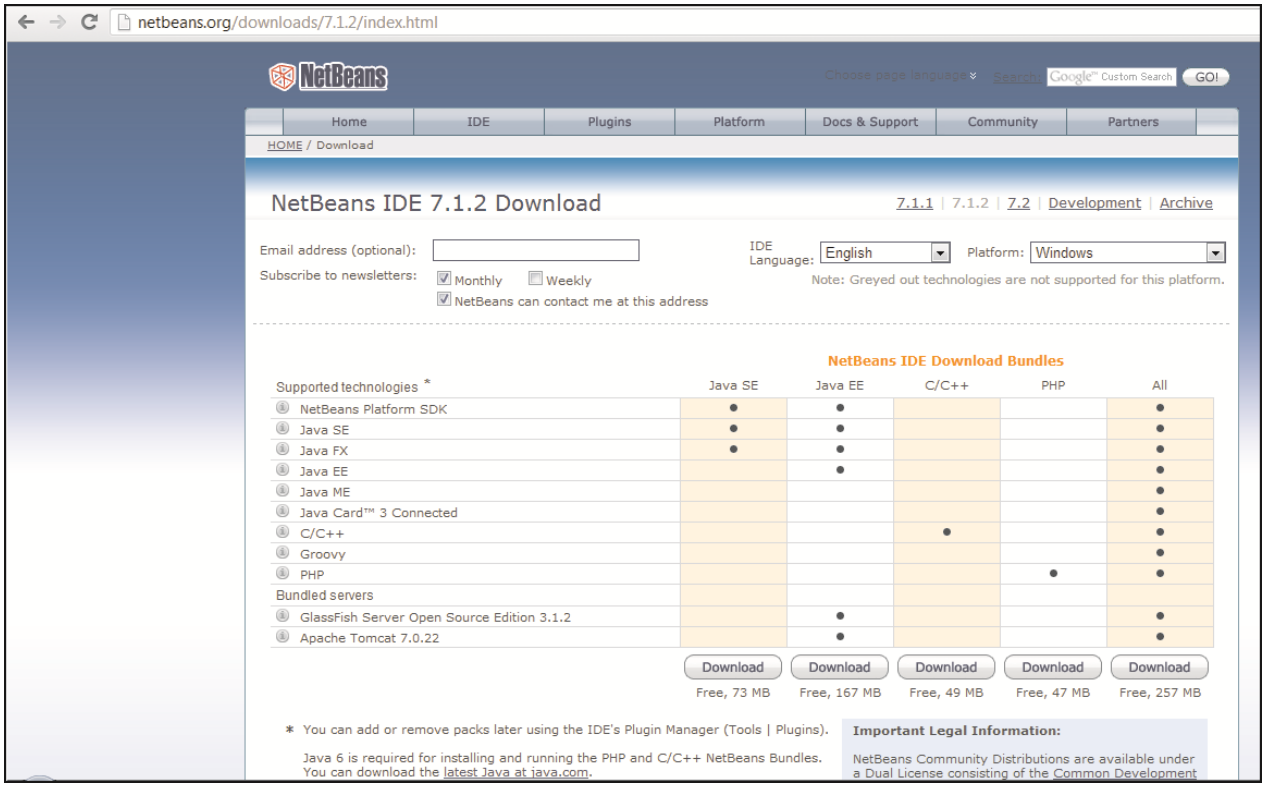
- The latest version of NetBeans IDE for different platforms, such as Windows, Linux, Solaris, and Mac OS X is available for download at `http://netbeans.org/downloads/index.html`.
- On the NetBeans IDE download Web page, you will find different installers.
- Each installer of NetBeans IDE contains the basic IDE and its related tools.

Download and Install NetBeans IDE 2-5

- The different installers are as follows:
  - **Java SE** - Supports all standard features that are necessary for Java SE development.
  - **Java EE** - Provides tools for developing Java SE and Java EE applications. This download option also includes GlassFish Server Open Source Edition and Apache Tomcat software.
  - **C/C++** - Supports development in the C, C++, Fortran, and Assembly languages.
  - **PHP** - Provides tools for PHP 5.x development, Zend, and Symfony Framework support.
  - **All** - This is a full download option, which contains all the runtimes and technologies available for the NetBeans IDE.

Download and Install NetBeans IDE 3-5

- To download the NetBeans IDE 7.1.2, perform the following steps:
  - Type `http://netbeans.org/downloads/7.1.2/index.html` in the **Address bar** of the Web browser .
  - Following figure shows the download Web page for the NetBeans 7.1.2 IDE:



Download and Install NetBeans IDE 4-5

- Select **IDE language** as **English** from the drop-down list. Also, select the **Platform** as **Windows** from the drop-down list.
- Click **Download** under the installer **All**. The **Save As** dialog box is opened with netbeans-7.1.2- ml-windows.exe installer file. This installer will support development of all technologies in the NetBeans IDE.
- Click **Save** to save the installer file on the local system.

- To install the NetBeans IDE, perform the following steps:

1. Double-click `netbeans-7.1.2-ml-windows.exe` to run the installer.
2. Click **Next** at the **Welcome** page of the installation wizard.
3. Click **Next** in the **License Agreement** page after reviewing the license agreement, and select the acceptance check box.

Download and Install NetBeans IDE 5-5

4. At the **JUnit License Agreement page**, decide if you want to install JUnit and click the appropriate option, click **Next**.
5. Select either the default installation directory or specific directory where the NetBeans IDE needs to be installed. Set the path of the **default JDK** installation and click **Next**.
6. The **GlassFish Server Source Edition 3.1.2** installation page is displayed. You can either select the default location or specify another location to install the GlassFish Server.
7. To install the **Apache Tomcat Server**, on the installation page, either select the default location or specify another location and then, click **Next**.
8. The **Summary** page is opened. The list of components that are to be installed is displayed,
9. Click **Install** to install the NetBeans IDE on the system.
10. After the installation is completed, click **Finish** to complete and close the setup page.

Writing a Java Program Using NetBeans IDE

- The basic requirements to write a Java program using the NetBeans IDE is as follows:
  - The **JDK 7** installed and configured on the system
  - The **NetBeans** IDE

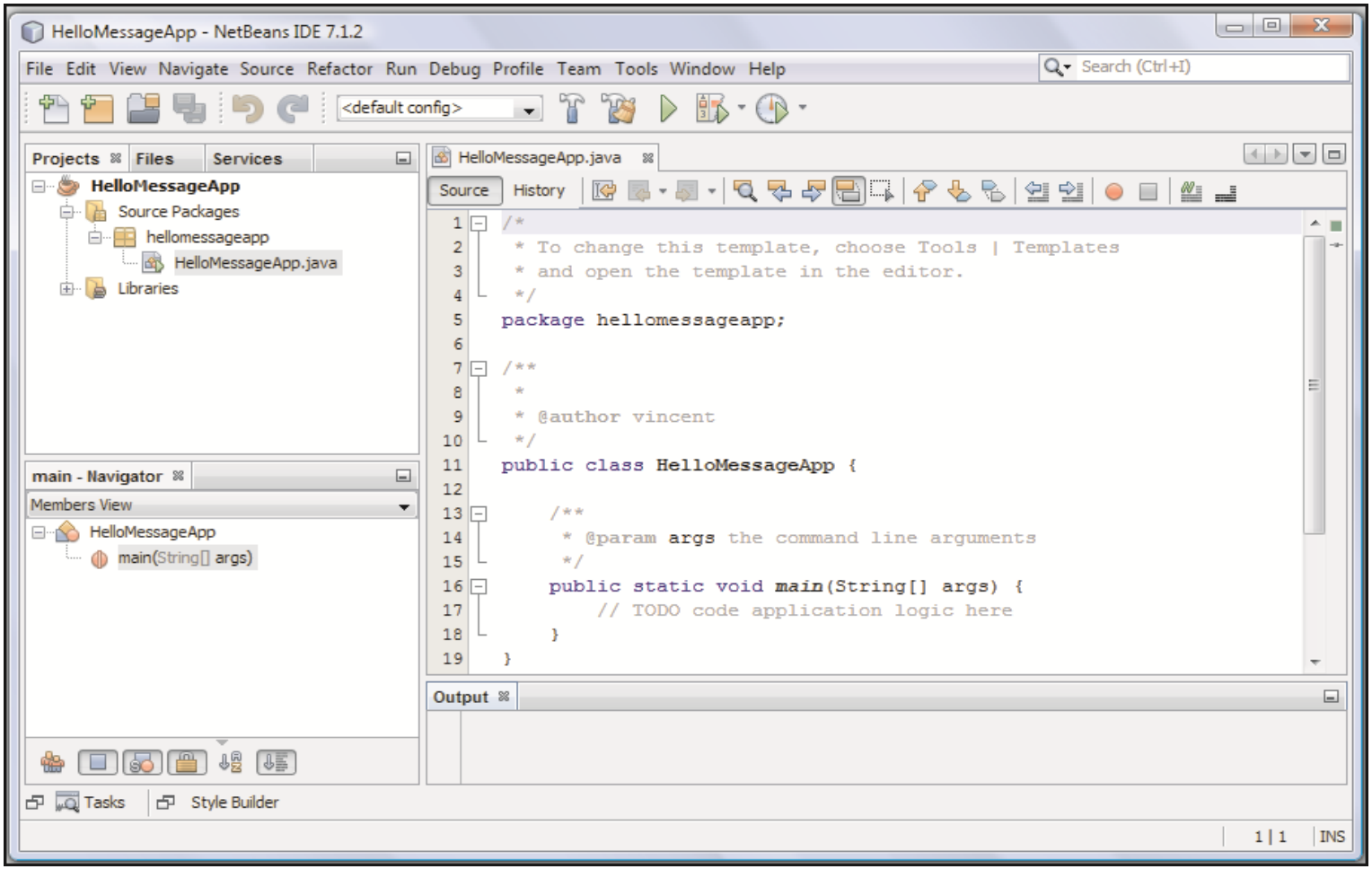
- To develop a Java program in NetBeans IDE, perform the following steps:

Create a Project in IDE 1-2

- To create a project in IDE, perform the following steps:
  - To launch NetBeans IDE, click **Start→All Programs→NetBeans** and select **NetBeans IDE 7.1.2**.
  - To create a new project, click **File→New→Project**. This opens the **New Project** wizard.
  - Under **Categories**, expand **Java** and then, select **Java Application** under Projects.
  - Click **Next**. This displays the **Name and Location** page in the wizard.
  - Type **`HelloMessageApp`** in the **Project Name** box.
  - Click **Browse** and select the appropriate location on the system.
  - Click **Finish**.

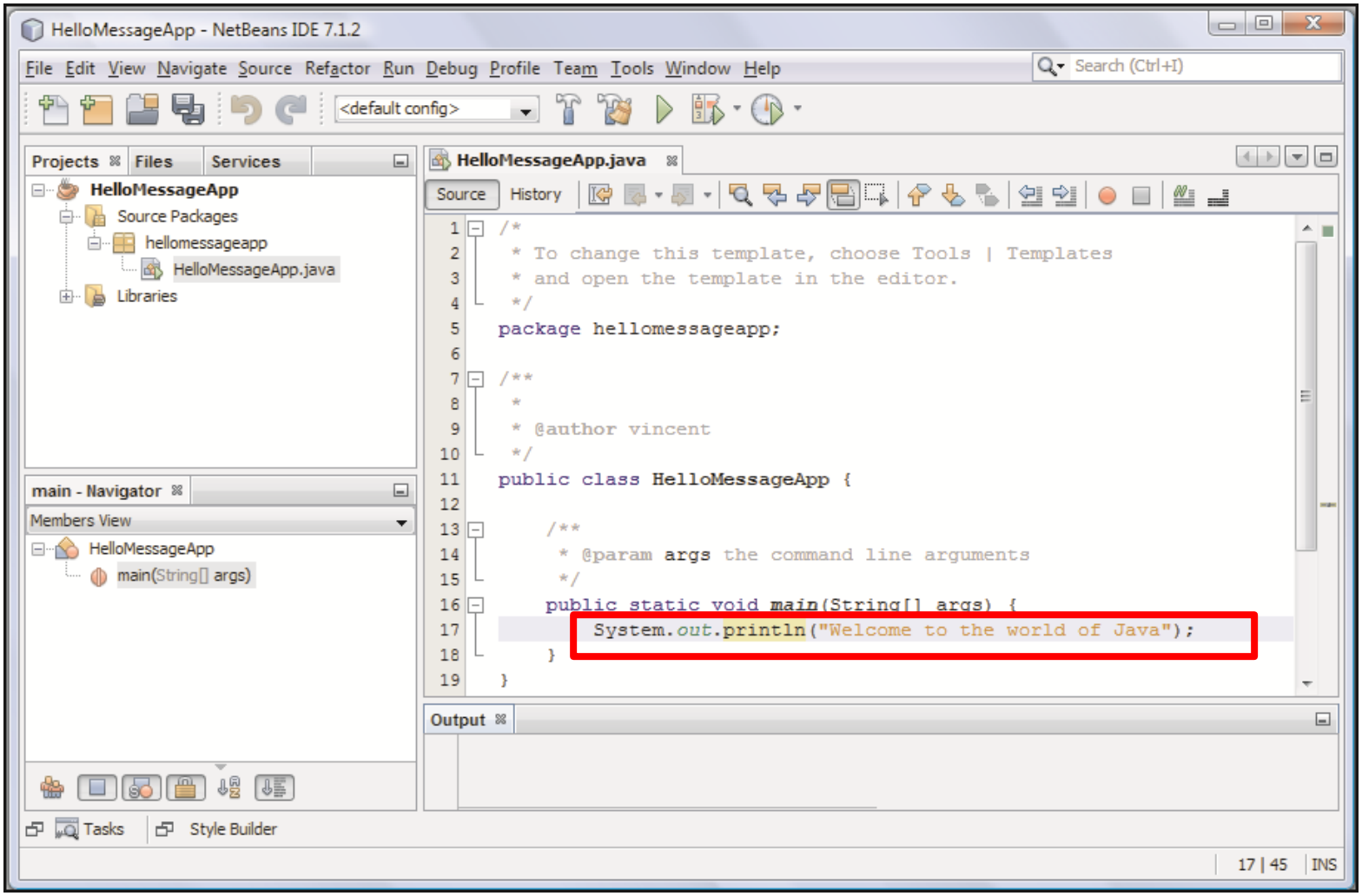
Create a Project in IDE 2-2

- Following figure shows the **`HelloMessageApp`** project in the NetBeans IDE:



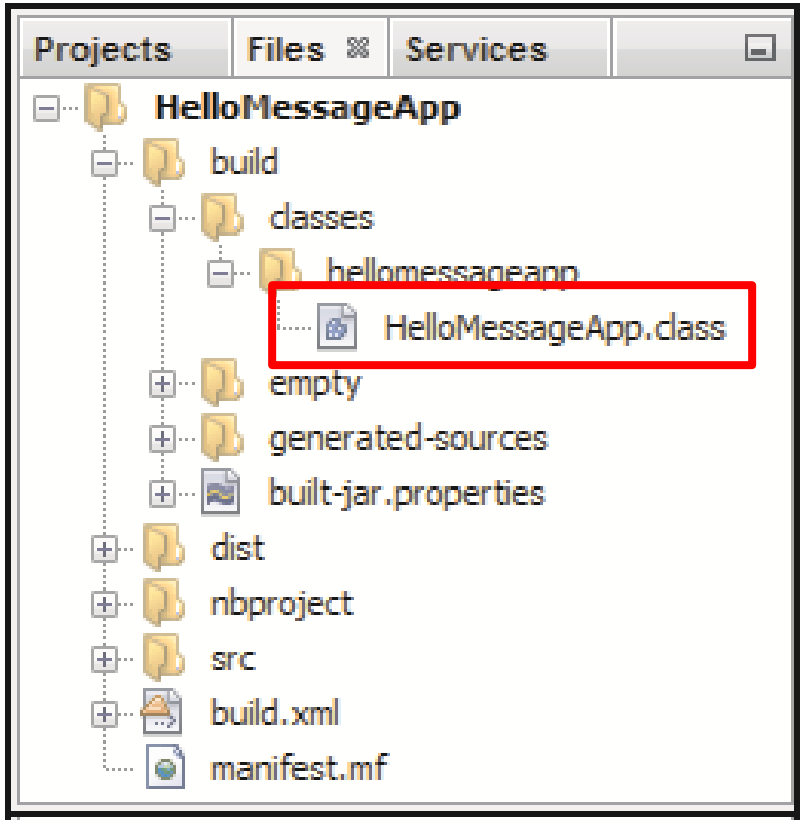
Add Code to the Generated Source Files

- The necessary skeleton of the program has been created by the IDE.
- Following figure shows the NetBeans IDE with the modified code:



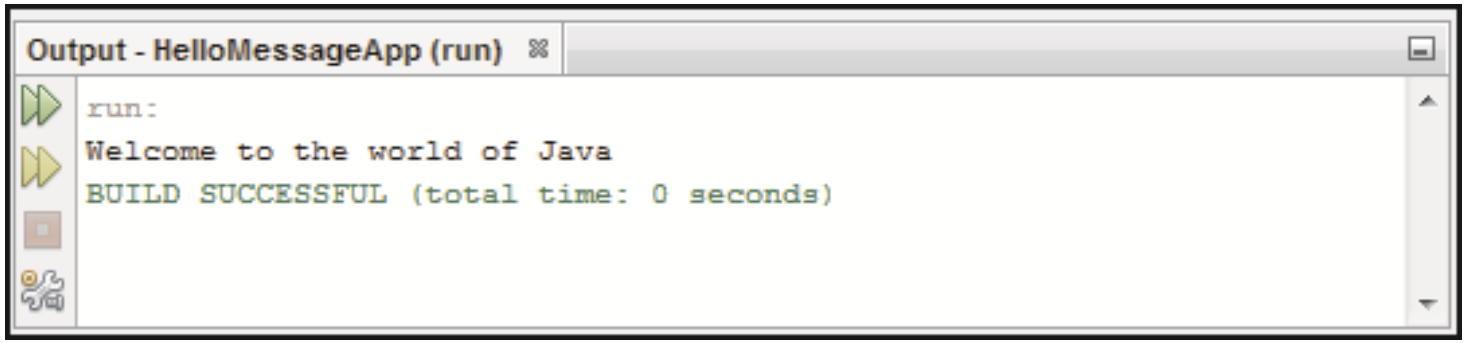
Build and Execute Java Program in NetBeans IDE 1-2

- To compile the source file, **`HelloMessageApp.java`**, click **Run→Build Main Project** in the NetBeans IDE menu bar.
- Following figure shows **Files** window that shows the generated bytecode file, **`HelloMessageApp.class`** after the project is build successfully:



Build and Execute Java Program in NetBeans IDE 2-2

- To execute the program, click **Run→Run Main Project.**
- Following figure shows the **Output** window that displays the output of the **`HelloMessageApp`** program.



Comments in Java

- Are placed in a Java program source file.
- Are used to document the Java program and are not compiled by the compiler.
- Are added as remarks to make the program more readable for the user.
- Are of three types:
  - Single-line comments
  - Multi-line comments
  - Javadoc comments

Single-line Comments 1-2

- A single-line comment is used to document the functionality of a single line of code.

- There are two ways of using single-line comments that are as follows:

Beginning-of-line comment

- This type of comment can be placed before the code (on a different line).

End-of-line comment

- This type of comment is placed at the end of the code (on the same line).

- The syntax for applying the comments is as follows:

Syntax

```
// Comment text
```

Single-line Comments 2-2

- Following code snippet shows the different ways of using single-line comments in a Java program:

```
...
// Declare a variable
int a = 32;
int b // Declare a variable
...
```

- Conventions for using single-line comments are as follows:
  - Insert a space after the forward slashes.
  - Capitalize the first letter of the first word.

Multi-line Comments

- Is a comment that spans multiple lines.
- Starts with a forward slash and an asterisk (/*).
- Ends with an asterisk and a forward slash (*/).
- Anything that appears between these delimiters is considered to be a comment.
- Following code snippet shows a Java program that uses multi-line comments:

```
...
/*
 * This code performs mathematical
 * operation of adding two numbers.
 */
int a = 20;
int b = 30;
int c;
c = a + b;
...
```

Javadoc Comments 1-2

- Is used to document public or protected classes, attributes, and methods.
- Starts with /** and ends with */.
- Everything between the delimiters is a comment.
- The `javadoc` command can be used for generating Javadoc comments.
- Following code snippet demonstrates the use of **Javadoc** comments in the Java program:

```
/**
 * The program prints the welcome message
 * using the println() method.
 */
package hellomessageapp;
```

Javadoc Comments 2-2

```
/**
 *
 * @author vincent
 */
public class HelloMessageApp {

  /**
   * @param args the command line arguments
   */
   public static void main(String[] args) {

// The println() method displays a message on the screen
System.out.println("Welcome to the world of Java");

  }
}
```

Summary

- The Java programming language is designed around object-oriented features and begins with a class design.
- A Java class structure contains the following components namely, package, import, class name, variables, and methods.
- Java programs can be written in a text editor, such as Notepad or in an IDE, such as NetBeans.
- The entry point for a Java console application is the main() method.
- The javac.exe compiles the source code into a .class file. Similarly, the java.exe command is used to interpret and run the Java bytecodes.
- NetBeans is a free and robust IDE that helps developers to create cross-platform desktop, Web, and mobile applications using Java.
- Comments are used to document the Java program and are not compiled by the compiler. There are three styles of comments supported by Java namely, single-line, multi-line, and Javadoc.