# INTRODUCTION TO JAVASERVER PAGES

JavaServer Pages (JSP) is technology developed by Sun Microsystems that is used to create powerful and dynamic Web sites.

### Web Servers

You do not require a dedicated Web server to publish the JSP pages you create. You can simply install Web server software on your own computer. A popular example of Web server software that includes support for JavaServer Pages is Tomcat. The Tomcat Web server is commonly used by Web developers who create JSP pages.

You do not require any special development tools to create and view JSP pages. All you need is a text editor and a Web browser.

### Versions

JavaServer Pages 1.1 is the current version of JavaServer Pages, although the specification for a newer version 1.2 has been proposed. On average, a new version of JavaServer Pages is produced each year. Each new version offers more features than previous versions of the technology. The Web server you are running will determine the version of JavaServer Pages you can use and the tasks you can perform.

### Programming Languages

The Java programming language forms the basis of JavaServer Pages technology. The version of Java you use depends on the Web server you are running and the version of JavaServer Pages technology you are using.

### Servlet Technology

JavaServer Pages is based on servlet technology, which allows Web developers to use Java code to create dynamic Web pages. JavaServer Pages simplifies the process of creating dynamic pages using Java.

### Server-side Processing

JavaServer Pages uses a JSP engine that is part of the Web server, so the processing of JSP code takes place on the server. When a user requests a JSP page, the JSP engine processes the page and then sends the result as HTML code to the user's Web browser. This allows JSP pages to be viewed by every Web browser.

### JSP Implicit Objects

JSP includes implicit objects that can be used to perform specific tasks. For example, the `session` object can be used to store session information about a client computer as the client navigates a Web site. Other commonly used implicit objects include the `response` object, which sends information to a client, and the `request` object, which retrieves and controls information sent from a client to the Web server.





## FEATURES OF JAVASERVER PAGES

### Create Dynamic Web Sites

Dynamic Web sites contain Web pages that display constantly changing content. Using JavaServer Pages, you can determine the content a Web page displays, depending on many different factors. For example, you can have a page automatically present different content to users depending on the current date or the user's location. Dynamic Web pages are more useful to each individual user than static Web pages.

### Create Interactive Web Sites

Interactive Web sites contain Web pages that exchange information between the Web site and the user. JavaServer Pages allows Web developers to easily create Web pages that process information from a user and then generate content depending on the information submitted by the user. Interactive Web sites allow Web developers to tailor the content of Web pages to better appeal to the user.

### Increased Security

Because JavaServer Pages code is processed on the Web server, the user cannot access the code used to create a JSP page. This makes it safer to work with sensitive data, such as login names and passwords. If a user views the source code of a JSP page within a Web browser, all the user will see is the HTML code that was generated by the Web server to create the page, not the JavaServer Pages code itself.

### Work With Databases

An important feature of JavaServer Pages is the ability to connect to a database. JSP pages can be used to make information stored in a database available to the users who visit a Web site. Using databases to store information and JSP pages to access the information is an efficient method of displaying up-to-date information in a Web site.

JavaServer Pages can also allow users to manipulate the data in a database. For example, a JSP page can be used to add, delete or edit records in a database.

### Using JavaBeans

JavaBeans are re-usable components that allow Web developers to keep the Java code for a JSP page separate from the HTML code for the page. This helps prevent the code on a JSP page from becoming long and difficult to work with and allows Web developers to share and re-use Java code. JavaBeans also enable specialization when developing a Web site by allowing experts in Web page design to work with the HTML content for a page while programmers develop the Java code for the page.

### Using Custom Tags

JavaServer Pages technology allows Web developers to create their own custom tags that perform specific tasks. Like JavaBeans, tag libraries are re-usable components that help keep the Java code for a JSP page separate from the HTML code for the page. Once a tag library containing the code for a custom tag has been created, the custom tag can be used in JSP pages.

# INTRODUCTION TO JAVA

Java is a programming language used to create applications for the World Wide Web. Java was originally developed by Sun Microsystems in 1991 for use in consumer electronics such as handheld computers and television sets, but the language was later modified for use on the Web. Java is now a full-featured programming language that is easy to use and understand.

Java is the main programming language used in JavaServer Pages. While in-depth knowledge of Java is not required, in order to effectively use JavaServer Pages you need to understand the basics of the language. A thorough understanding of Java will enable you to create more sophisticated, versatile and efficient JSP pages.

You can also utilize your knowledge of Java to work with other Java-based technologies, such as JavaBeans.

The popularity of JavaServer Pages is partly due to the fact that people who are already familiar with the Java programming language do not need to learn a new programming language in order to use JavaServer Pages. Since JavaServer Pages uses Java code to create Web pages, programmers can use their existing knowledge of Java to create JSP pages. JavaServer Pages also uses programming code that is unique to JavaServer Pages and is not strictly Java code.

### Features

Java includes a number of features that make the language ideal for use on the Web. Java programs transfer quickly over the Web since the language was created to be portable and file sizes are small. In addition, Java is platform independent. This means that a Java program can be run on any computer that has a Java virtual machine, regardless of the operating system the computer uses. This feature is invaluable for use on the Web, where computers using various languages and environments must interact.

### Bytecode

When a Java program is compiled, the program is not immediately translated into machine code, which are instructions specific to a particular operating system. Instead, it is compiled into an intermediate language, called bytecode, that can be interpreted by a Java virtual machine. When the Java program is run on a computer that has the Java virtual machine, the Java interpreter translates the bytecode into code that the computer running the program can understand.

### Object-Oriented

Java is an object-oriented programming language, so if you understand the fundamentals of how Java works, you will understand the fundamental concepts of object-oriented programming. Object-oriented programming is a type of programming that treats separate pieces of code as distinct modules, or objects. It is often easier to learn object-oriented programming if you do not have vast experience with programming languages that are non-object-oriented. Despite its apparent initial complexity, object-oriented programming is easy to learn.

### Security

Java provides a number of advanced security features, such as access controls, which are not offered by many other programming languages. Java programs may contain viruses or code that can cause computer problems. Java's access controls allow programmers to use untrusted Java code in their programs without putting their systems at risk.





## PROGRAMS FOR CREATING JAVA CODE

The first step in creating Java programs is to select the method you want to use to create the Java programming code, or *source code*.

### Text Editors

Since all Java source code is plain text, you can use a simple text editor to create the source code.

#### For Non-Windows Operating Systems

There are many text editors that can be used to create Java source code on UNIX computers. Most UNIX computers have multiple text editors installed by default. Text-based editors such as vi and Emacs are very popular and can be configured to suit your needs. If your UNIX system has a graphical interface, like GNOME, then you should already have access to graphical text editors, such as gnotepad+.

On the Macintosh operating system, you can use the SimpleText text editor included with the system.

#### Notepad

Microsoft Notepad is a simple text editor available on all computers running the Windows operating system. Most operating systems contain a text editor similar to Notepad. While very basic, Notepad is more than adequate for creating source code and is widely used by programmers.

#### UltraEdit

UltraEdit is a sophisticated text editor popular with many programmers. UltraEdit's advanced features include syntax highlighting, which highlights the Java code to make the code easier to read, and the ability to save Web pages directly to a Web server. UltraEdit is a shareware program available at www.ultraedit.com.

### HTML Editors

HTML editors are programs specifically designed to help you create Web pages. Compared to text editors, HTML editors usually offer more advanced features to help you work with HTML and Java code.

#### BBEdit

BBEdit is a sophisticated HTML editor for the Macintosh operating system. BBEdit makes it easy to create JavaServer pages and is available at www.barebones.com.

#### HomeSite

Allaire's HomeSite is a comprehensive HTML editor designed for creating Web pages on the Windows operating system. HomeSite is suitable for beginners creating a small number of Web pages and for experienced Web masters producing complicated Web pages and Web sites. HomeSite includes syntax coloring for JavaServer Pages code and allows you to view the results generated by the code within HomeSite. HomeSite is available at www.allaire.com.

### Integrated Development Environments

Instead of a text editor, you can use a Java Integrated Development Environment (IDE). An IDE is a program that allows you to create, execute, test and organize your source code. IDEs often contain additional features such as sample code, reusable components and troubleshooting capabilities. IDEs are commonly used to create larger applications and to enable multiple programmers to work on a single project at the same time.

#### JBuilder

Borland's JBuilder is one of the more popular Java IDEs. JBuilder is a sophisticated, full-featured IDE that can be used to create JSP pages and complex Java applications. JBuilder is also available for various UNIX operating systems. JBuilder is available at www.borland.com/jbuilder.

# OBJECT-ORIENTED PROGRAMMING CONCEPTS

Java shares many concepts with other object-oriented programming languages, such as C++ and Perl. While object-oriented programming languages use the same concepts, the terminology and coding systems sometimes differ. For example, in Perl, a single value in an object is referred to as a property. In Java, this is referred to as a field.

The amount of object-oriented programming a Web site requires depends on the size and scope of the Web site. It also depends on where you store the Java code. Storing the Java code in the JSP pages themselves requires much less object-oriented programming than storing the Java code in external modules, referred to as *JavaBeans*.

## JAVA CONCEPTS

### Classes

A class is the Java code that serves as a template or plan for creating objects, which are the core features of object-oriented programming. A single class can be used to create many objects. For example, a class containing code for generating messages can be used to create an object that displays a welcome message at the top of each Web page. The same class can be used to create another object that displays copyright information at the bottom of a page. Classes can be used and shared by more than one Java program and therefore help programmers avoid having to constantly rewrite the same type of code.

### Objects

An object is a package of code that is composed of data and procedures that make use of the data. Objects have two primary functions–to store information and to perform tasks. Objects contain fields, which are used to store information, and methods, which are used to perform tasks. Objects can be created to perform a single task or a range of related tasks. Multiple objects can be created using the same class. When an object is created, it is said to be an instance of the class used to create the object.

### Fields

Fields, also known as data fields, are the properties or attributes associated with an object. In comparison to other programming languages, fields can be thought of as variables of the class. Fields can store different types of data, such as strings of text, integers and references to other objects.

Changing the value of an object's fields usually affects the behavior of the object. For example, in an object used to display a changing message on a Web page, a field may be used to specify how often the message changes. With a field value of 1, the message will be updated once every minute. When the field value is changed to 60, the message will be updated once an hour.

When multiple objects are created using the same class, it is typical for the objects to be the same except for the values held in the objects' fields.

### Methods

Methods are code that objects use to perform a specific task. A class used to create objects can contain multiple methods. The methods in a class usually perform related tasks. For example, in a class used to format text information on Web pages, one method may be used to generate the code needed to format the headers of paragraphs. Another method may be used to format information in a table. The behavior of methods may be influenced by the values stored in the fields of the object.

#### Arguments

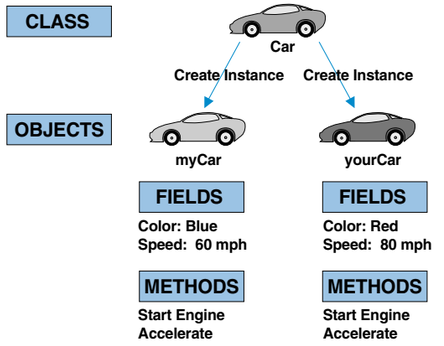
One or more values, called arguments, may be passed to a method to provide the method with input data or additional information about how to perform a task. For example, when using a method that creates tables on a Web page, you may need to pass the number of rows and columns for a table to the method. Some methods do not require any arguments.

#### Return Values

A method may return a value after performing a specific task. The return value may indicate the result of a calculation or it could indicate whether or not the task was performed successfully. For example, a method that writes information may return a true or false value, which the program can use to determine the next code that should be executed.

### Object Relationships

The following diagram shows how a single class can be used to create multiple objects, each with its own distinct fields and methods.




### Data Hiding

Data hiding makes classes easier to use by hiding the fields and methods of the classes from other parts of the program. The program then has to know only how to access the class, not the internal workings of the class. Data hiding is often used in programs to protect classes from tampering and to ensure that the methods of the classes are used as originally intended. A programmer can modify and maintain the code within the class without affecting the programs that use the class. This also helps ensure that objects developed by multiple people are compatible.

# THE JAVA CLASS LIBRARY

The Java class library is a collection of predefined classes that you can use in your programs. The Java class library is also known as the standard class library or the Java Applications Programming Interface (Java API).

### Classes

Some predefined classes are used often in Java programs, such as those used to display output, while other classes are used infrequently, such as the classes used to create Graphical User Interfaces (GUIs). The classes included in the Java class library are available to every Java program you create. Using the predefined classes in the Java class library saves you time and effort when creating programs.

### Java Class Library Installation

The Java class library is installed automatically when the Java Software Development Kit is installed on a computer. The Java class library is stored in a Java archive file named rt.jar in the lib subdirectory of the jre directory. The jre directory is located in the main Java SDK directory. You do not need to adjust any settings on your computer to specify the location of the Java class library before using a class from the library in your code.

### Packages

The classes that make up the Java class library are organized into packages. A package is a set of related classes stored in a separate directory. For example, classes that are used to generate output are stored in a different package than classes used to process data from a database. Generally, classes stored in the same package can easily access each other.

Package names are based on the directory structure that stores the classes in the package. For example, the classes in the `java.util` package are stored in the util subdirectory of the java directory.

### Import Packages

You can import a package from the Java class library into a Java program. This allows you to efficiently use all the classes in the package. The `java.lang` package is automatically imported into every Java program you create. For more information about importing a package, see page 52.

### Create Packages

In addition to using predefined classes from Java class library packages, you can author your own classes and store them in packages you create. For example, if you create three classes to work with a Web site, you could store these classes in a package named website. You could then use the classes from the package when creating other Java applications. For more information about creating packages, see page 50.

### Commonly Used Java Class Library Packages

The Java class library contains more than 70 packages. The following is a list of some of the most commonly used packages in the library.

`java.io`

Contains classes that allow Java programs to perform data input and output tasks.

`java.lang`

Contains the fundamental classes of the Java programming language and is automatically loaded by the Java compiler.

`java.math`

Contains classes that allow Java programs to perform arbitrary-precision arithmetic.

`java.lang.ref`

Contains classes that allow Java programs to interact with the garbage collector, which performs memory management tasks.

`java.lang.reflect`

Contains classes that allow Java programs to obtain information about the variables and methods of loaded classes.

`java.security`

Contains classes that allow Java programs to carry out security procedures, such as controlling access and encrypting data.

`java.sql`

Contains classes that allow Java programs to access and process data from a database.

`java.text`

Contains classes that allow a Java program to manipulate strings, dates, numbers and characters.

`java.util`

Contains utility classes that allow Java programs to perform various tasks such as date and time operations and random number generation.

`java.util.jar`

Contains utility classes that allow Java programs to read and write Java ARchive (JAR) files.

`java.util.zip`

Contains utility classes that allow Java programs to read and write ZIP files.

`javax.swing`

Contains classes for creating Swing Graphical User Interface (GUI) components. Swing GUI components can be used on all platforms.

# JAVA CONVENTIONS

To use the Java programming language effectively, there are several conventions you should know. For more information about the conventions used in Java, you can consult the Java SDK documentation.

### Semicolons

Most Java statements end with a semicolon (;). Java statements that include a block of code, known as the body of the statement, are the exception. Examples of these types of statements include methods, conditional statements and statements that create a loop. The Java compiler will stop compiling code and report an error if a required semicolon is missing or a semicolon is used where one is not needed. When an error occurs due to the omission or misplacement of a semicolon, the Java compiler may indicate that the error is in the statement following the actual location of the error. To avoid these types or errors, you should always review your Java code carefully before compiling the code.

### Braces

Java statements that include a body use braces { } to indicate the beginning and the end of the body. A body often contains several statements. If a statement block contains only one statement, braces are typically not required. There are two accepted formats that you can use when including braces in your Java code. You should choose one format and then use that format consistently throughout your code.

The most widely used format places the opening brace on the same line as the Java statement. The closing brace is placed on its own line and in the same column as the first character of the Java statement that uses the braces.

**Example:**

```
public static void main(String[] args) {
      System.out.println("Hello.");
      System.out.println("My name is Bob.");
}
```

The second format places each brace on its own line. The braces are in the same column as the first character of the Java statement that uses the braces. This format is easier to read, but adds more lines to your Java code.

**Example:**

```
public static void main(String[] args)
{
      System.out.println("Hello.");
      System.out.println("My name is Mary.");
}
```

### Indenting

When working with a Java statement that includes a body, you should always indent the code in the body. Indenting makes your code easier to read. Tabs or spaces can be used to indent code. To keep your Java programs consistent, you should use the same indenting style in all your code.

**Code without indents:**

```
public static void main(String[] args)
{
int counter = 1;
while (counter <= 5)
{
System.out.println(counter);
counter++;
}
}
```

**Code with indents:**

```
public static void main(String[] args)
{
      int counter = 1;
      while (counter <= 5)
      {
            System.out.println(counter);
            counter++;
      }
}
```

### White Space

White space is the term used to describe characters that are not displayed or printed, such as spaces, tabs and newlines. Using white space in your Java code can greatly improve the readability of your code. For example, x + 1 / age is easier to read than x+1/age. The Java compiler ignores white space. This means that using white space will not affect the speed at which your Java code is compiled.

### Comments

You can include comments in your Java code to explain important or difficult sections of code. Adding comments to your code is a good programming practice and can help make the code easier to understand. Comments are particularly useful if you or someone else will need to modify or troubleshoot the code in the future. For more information about adding comments to your Java code, see page 15. Using descriptive names for items such as classes, methods and variables can also make your code easier to understand.

### Keywords

The Java programming language includes many keywords. A keyword is a word reserved for use only by Java. You cannot use keywords as variable names or values in your code. If you use a Java keyword inappropriately, the Java compiler will usually detect the error and stop compiling the code. The following table displays a listing of Java keywords:

| | | | |
|---|---|---|---|
| abstract | else | interface | super |
| boolean | extends | long | switch |
| break | false | native | synchronized |
| byte | final | new | this |
| case | finally | null | throw |
| catch | float | package | throws |
| char | for | private | transient |
| class | goto | protected | true |
| const | if | public | try |
| continue | implements | return | void |
| default | import | short | volatile |
| do | instanceof | static | while |
| double | int | strictfp | |

# INSTALL THE JAVA SOFTWARE DEVELOPMENT KIT

The Java Software Development Kit (SDK) is a collection of programs used to compile and execute Java programs. You need to install the Java SDK in order to install the Tomcat Web server, which allows you to create and test JavaServer Pages.

The Java Software Development Kit is constantly being updated. A recent release of the Java SDK for Windows is included on the CD-ROM disc that accompanies this book, but you should make sure you use the latest release of the kit. More information about the latest release of the Java SDK is available on the Java Web site at java.sun.com. The Java SDK is also currently available for the Sun Solaris and Red Hat Linux platforms. Downloading and installation instructions are available at the Java Web site.

On the Windows platform, the Java SDK is installed using a standard Windows installation program. The Java SDK installation program selects a folder where the kit will be installed for you. It is recommended that you accept this folder. During the installation, you can select which components of the Java SDK you want to install, such as demos. It is recommended that you install all the available components.

The installation program allows you to choose to view a README file that contains information about the release of the Java Software Development Kit you installed and any last minute changes to the documentation. If you choose to display the file, it will open when the installation is complete. You should carefully review the README file for any new release of the Java SDK you install.

Once the Java SDK has been installed, you should restart your computer, particularly if you are upgrading from an older release of the Java SDK.

## Extra

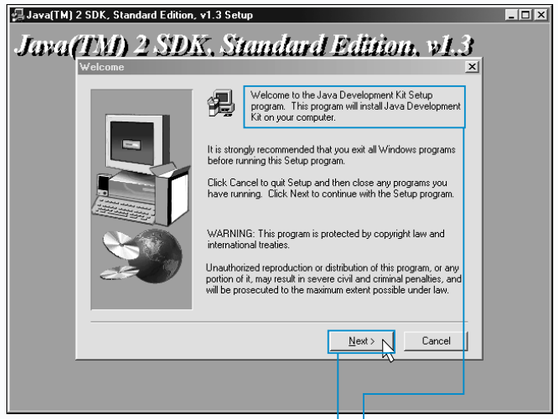
The Java SDK installation program is over 20 megabytes (MB) in size. If you are using a modem to connect to the Internet, the program can take a few hours to download. For convenience, you may want to start the download and let it continue through the night.

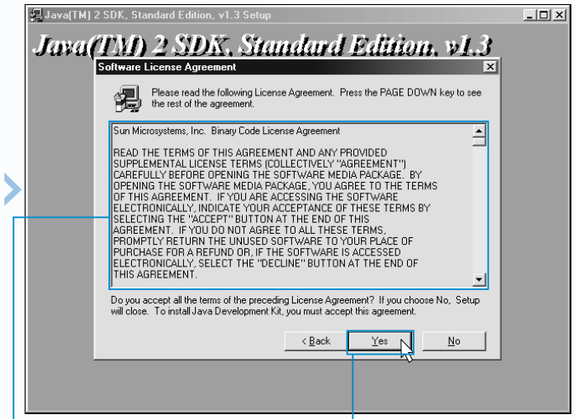
If you already have a previous release of the Java SDK installed on your computer, it is recommended that you uninstall the previous release before upgrading to the latest release of the Java SDK.

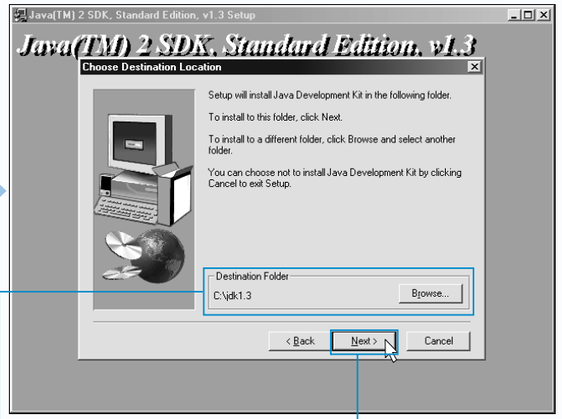
The Java SDK documentation can be downloaded separately from the Java Web site. It is recommended that you install and review the Java SDK documentation, particularly if you will be creating your own Java applications.

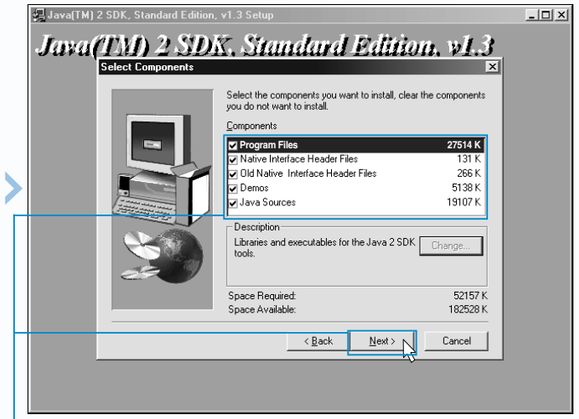
After installing the Java Software Development Kit, you may want to add the location of the Java SDK programs to the path variable of your computer's operating system. Setting the path variable will allow you to run your Java programs from any folder on the computer without having to type the full path to the Java compiler and interpreter. Refer to the documentation that came with the Java SDK and your operating system documentation for information about changing the path variable.

## INSTALL THE JAVA SOFTWARE DEVELOPMENT KIT




**1** Double-click the icon for the Java SDK installation program to start installing the kit.

- A setup window appears on the screen and a welcome dialog box is displayed.

- This area displays information about the Java SDK installation program.

**2** Click Next to continue.

- This area displays the license agreement you must read and accept before continuing.

**3** Click Yes to accept the agreement and continue.

- This area displays the folder where the Java SDK will be installed.

- You can click Browse to install the Java SDK in a different folder.

**4** Click Next to continue.

- Each component in this area that displays a check mark (☑) will be installed. You can click the box beside a component you do not want to install (☑ changes to ☐).

**5** Click Next to install the Java SDK components on your computer.

- A dialog box appears when the installation is complete. Click Finish to close the dialog box and then restart your computer.

# DECLARE A CLASS

After installing the Java Software Development Kit, you can begin creating Java programs. When creating Java programs, the first step is to declare a class. A class is the smallest unit of Java code that can be run and is the fundamental structure that Java applications use to group together related code. For example, a class called CheckText may contain all the code required to analyze and validate a string of text. The CheckText class can be used on its own in a program or used in conjunction with other classes. All Java applications must include at least one class.

Java classes are declared using the keyword `class` followed by the class name. The class name should be easy to understand and should indicate the purpose of the class. The class name is followed by a pair of braces { }. All methods and Java code in the class must be placed between the braces. The code between the braces is referred to as the body of the class and is made up of methods, which are structures that contain the Java code for specific actions. For more information about declaring a method, see page 16.

The class name you choose must be the same as the filename with which the program is saved. For example, if the class in your Java program is called DisplayText, the program must be saved with the filename DisplayText.java. It is also important to note that Java is a case-sensitive language. If the program is saved with the filename displaytext.java, an error may occur when you attempt to compile the program.

## Extra

Class names can begin with any letter, an underscore (_) or the symbol `$`, `£` or `¥`. Class names cannot begin with a number or contain any punctuation, such as a period or a comma. Class names also cannot be the same as any of the Java reserved words, such as `do`, `while` or `public`. These naming rules also apply to the naming of methods, fields and parameters in Java code.

You may want to add comments that span multiple lines to your Java code. To do so, type /* before the first line of the comment and */ after the last line of the comment.

**Example:**

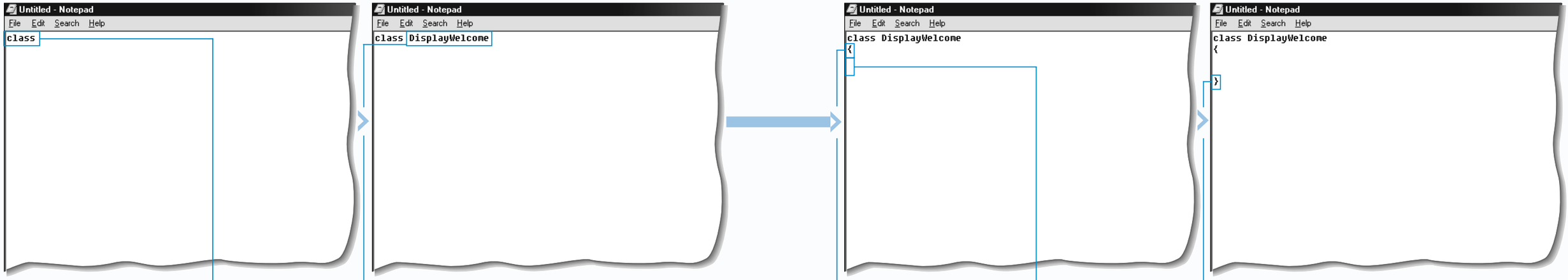
```
/*
This Java application
displays a welcome message when
the program is executed
*/
```

You should always include comments to make your Java code easier to understand. Comments are helpful if you or other people need to modify or troubleshoot the code. Any code you write should include comments that indicate the author's name and the main purpose of the program. Comments are preceded by // and can be included at the end of a line of code or on a separate line.

**Example:**

```
// Author: Martine Edwards
class DisplayWelcome  // A welcome message
{
     // The body of the class
}
```

## DECLARE A CLASS




**1** Start the text editor you will use to create a Java program.

**2** Type **class**.

**3** Type the name of the class you want to create.

**4** Type an opening brace to mark the beginning of the body of the class.

**5** Press Enter to create blank lines where you will type the body of the class.

**6** Type a closing brace to mark the end of the body of the class.

*Note: To declare methods for the class you created, see page 16.*

# DECLARE A METHOD

Once a class has been declared, methods can be declared for the class. Methods are similar to subroutines and functions that are found in other, non-object-oriented programming languages. Methods contain lines of code that perform a specific task, such as displaying an invoice or calculating the final total of an invoice.

Using methods makes it easy to re-use sections of code and allows you to group lines of code into smaller, more manageable sections. This makes it easier for people to understand and troubleshoot the code.

You can use method modifiers, such as `public` and `static`, to tell Java how a method is to be used. The `public` method modifier is an access modifier that indicates that this method can be used by other classes that you create. A `static` method modifier indicates that the method can be used by any program without having to create an object of the class that declares the method.

A method declaration should also include a return type. A return type specifies the type of value the method returns. If a method does not return a value to the code, the return type should be `void`. For more information about return values in methods, see page 38.

The name of a method is followed by parentheses, such as DisplayInvoice().

Every Java application must have a method called `main`, in which all the other methods required to run the program are called. The argument `String[] args` must be placed within the parentheses at the end of the method name for a `main` method. This argument indicates that the method can accept strings passed from the command line when the Java program is executed.

The method declaration ends with a pair of braces. The code that makes up the body of the method is placed inside the braces.

## Extra

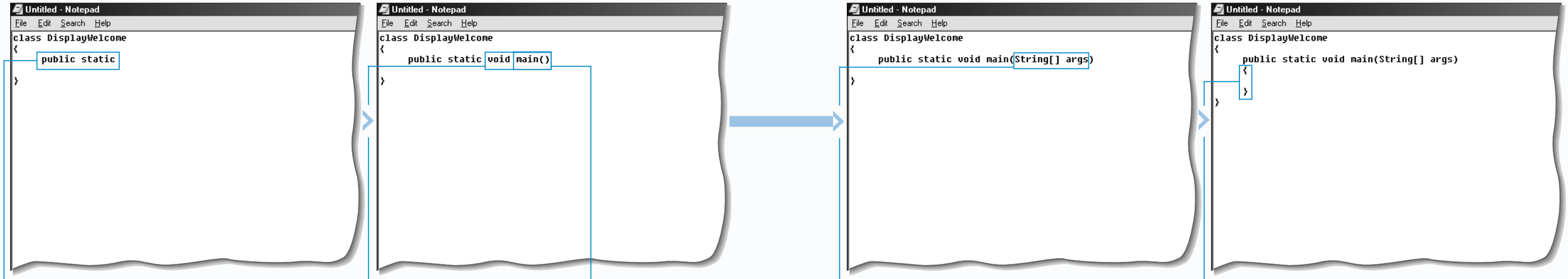
The name of the method should indicate the purpose of the method. A method name can consist of multiple words. To make the name easier to read, you can capitalize the first letter of each word, such as DisplayMyName.

You can use different access modifiers when declaring a method, depending on how the method will be accessed. The `public` access modifier indicates that the method can be accessed by any class and subclass within any *package*. The `protected` access modifier indicates the method can be accessed by any class within the same package and any subclass of the class that contains the method within a different package. The `private` access modifier indicates the method can be accessed only by the class that contains the method.

A method can generate a result which is returned to the code. The return type for a method that returns a value can be any valid data type in Java, such as `String`, `byte` or `boolean`. The body of a method that returns a value must also include a `return` statement. An error may occur if the data type of the value that is returned does not match the return type specified in the method declaration.

Every `main` method must include the `public`, `static` and `void` method modifiers. If one or more of the method modifiers are entered in a different order, the code may generate an error message.

## DECLARE A METHOD




**1** In the body of a class, type the method modifiers for the method you want to declare.

*Note: A `main` method must include the `public` and `static` method modifiers.*

**2** Type the return type of the method.

*Note: A method that does not return a value must include the `void` return type.*

**3** Type the name of the method followed by ().

**4** Between the parentheses, type any arguments the method requires.

*Note: The arguments of a `main` method must be `String[] args`.*

**5** Type the opening and closing braces that will contain the body of the method.

■ To create the body of the method, see page 18.

# CREATE THE METHOD BODY

The body of a method contains the Java code that is used to perform a task and must be created within the method's braces { }.

The code in the body of a method is often used to *call*, or access, another method. The called method can be declared in the same class or in a different class. Re-using methods saves you time and effort when writing Java programs. For example, if you create a method that displays your name and e-mail address, the same method can be used in any Java application you create.

The Java Software Development Kit includes many classes and methods that can be used to perform a wide variety of common tasks. For example, the Java SDK includes a class called `math`. The `math` class contains several methods that perform mathematical calculations. For example, to determine the square root of a number, you can simply call the `sqrt` method from the `math` class.

Methods can be used to display information on a user's screen. To display information, `System.out.print` can be used. The `System` object is included in the Java SDK and is created automatically when a Java program is executed. The `out` field is used to send information to the standard output device, typically the screen. The `print` member takes an argument that must be enclosed in parentheses. `System.out.print` can be used to display any type of data used in Java. When using `System.out.print` to display a string of text, the string must be enclosed in quotation marks.

Once you have finished creating the code for your Java program, save the code as a text file with the .java extension. The name of the file must be exactly the same as the name of the first class defined in the code.

## Extra

To start a new line at the end of a line of text, you can use the escape sequence `\n`. Using the escape sequence `\n` allows you to display text over multiple lines.

**TYPE THIS:**

```
class MyIntroduction
{
     public static void main(String[] args)
     {
          System.out.print("My name is Martine Edwards." + "\n");
          System.out.print("This is my first Java Program." + "\n");
     }
}
```

**RESULT:**

```
My name is Martine Edwards.
This is my first Java Program.
```

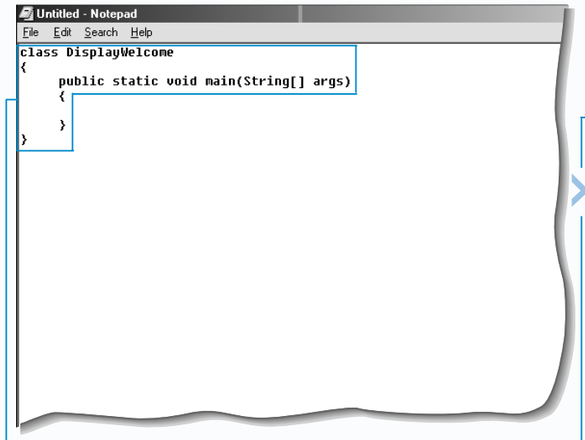
The classes and methods included with the Java Software Development Kit are collectively known as the *Java class library*, also called the Java Application Programming Interface or Java API. The Java SDK documentation describes all the classes and methods available in the Java class library. If you have not already installed the Java SDK documentation, you can obtain the documentation on the Web at java.sun.com.

`System.out.println` can also be used to start a new line.

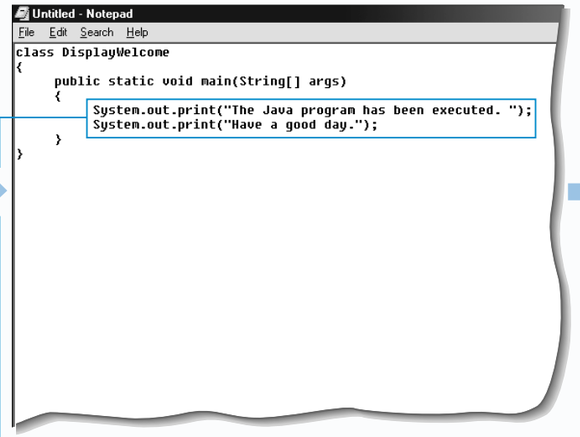
**Example:**

```
System.out.println("The Java program has been executed.");
System.out.println("Have a good day.");
```

## CREATE THE METHOD BODY

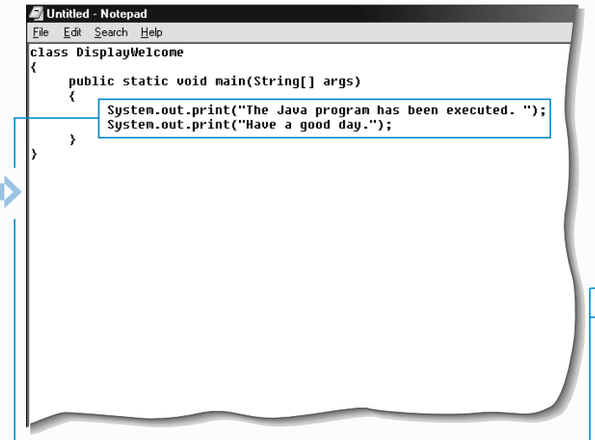



**1** Enter the code that declares the class and the method you want to use.



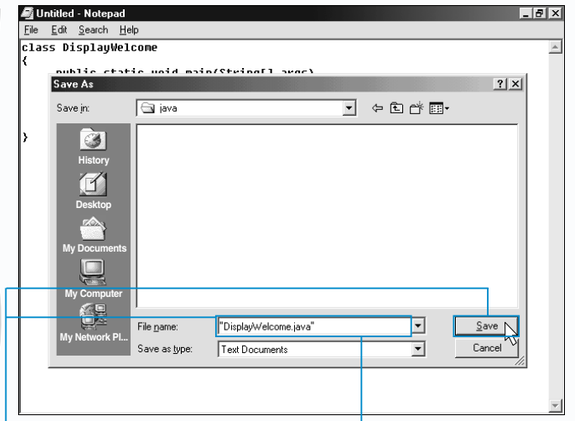

**2** In the body of the method, type the code for the task you want to perform.

- In this example, `System.out.print` is used to display output.




**3** Type any arguments the code requires.

- String arguments must be enclosed in quotation marks.




### SAVE JAVA CODE

**1** Save the Java code as a text file.

- The name of the file must be exactly the same as the name of the first class in the code. The filename must also have the .java extension.

- You may need to place quotation marks around the name of the file.

- You are now ready to compile the Java code. See page 20 to compile Java code.

# COMPILE JAVA CODE

Compiling Java code converts the source code into bytecode. Bytecode contains instructions that the *Java interpreter* executes.

A Java compiler is required to compile Java code. The Java Software Development Kit includes a Java compiler called javac. The javac compiler can only be executed from the command prompt. If you are using a Windows operating system, you will need to open an MS-DOS Prompt or Command Prompt window to use javac.

To compile Java source code, you enter the name of the Java compiler, such as javac, at the command prompt, followed by the name of the file that stores the code you want to compile. The filename must have the .java extension. Depending on whether you have added the location of the Java SDK programs to your operating system's path variable, you may need to specify the full path to the Java compiler, which is typically c:\jdk1.3\bin\javac. For information about setting the path variable, refer to the Java SDK installation instructions or your operating system's documentation.

Before compiling Java code, the Java compiler checks the code for errors. If an error is found, the code will not be compiled and an error message will be displayed.

If the Java code is successfully compiled, the resulting bytecode will be saved in a new file with the .class extension. The name of the new file is taken from the name of the file that stores the Java source code. For example, when the code in a file named Program.java is compiled, the bytecode is saved in a file called Program.class. The filenames of Java programs are case sensitive on most platforms.

Once Java source code has been compiled, the Java program is ready to be executed.

## Extra

When compiling Java source code, there are two main types of errors that can occur.

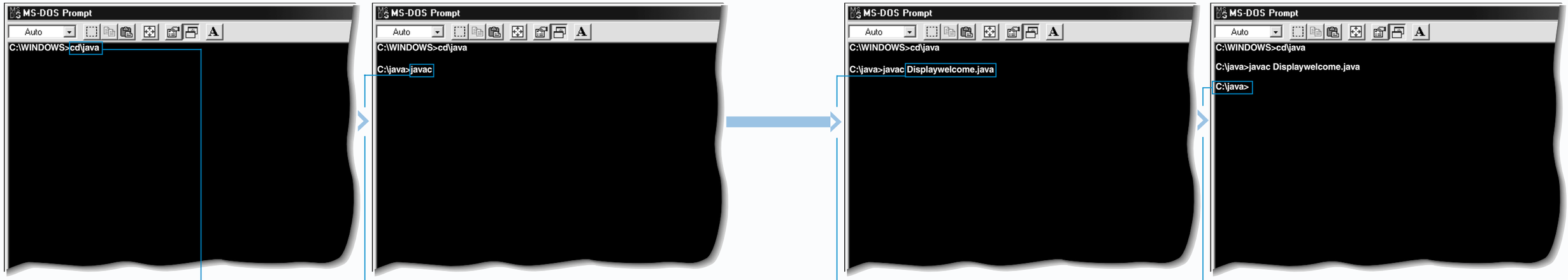
### Java SDK Errors

If your operating system cannot locate the Java compiler, a problem may have occurred when the Java SDK was installed. Java SDK errors usually result in an error message such as "bad command or file name." To correct this type of error, first determine the correct path to the compiler. If you cannot locate the Java compiler, try re-installing the Java SDK. If you were able to confirm the path to the compiler, ensure that you have not made any typing mistakes in the path.

### Source Code Errors

A wide variety of errors can occur in Java source code. When the Java compiler finds an error in source code, the compiler displays an error message that usually specifies the error type and where the error was detected. For example, the error "Program.java:5: invalid method declaration" indicates that an error involving a method declaration was generated at line 5 in the Program.java file. It is important to note that the line number indicates the line that the compiler was processing when the error was detected, which is not necessarily the line that contains the error.

## COMPILE JAVA CODE




**1** Open the window that allows you to work at the command prompt.

**2** Move to the directory that stores the Java code you want to compile.

**3** To compile the Java code using the javac compiler, type **javac**.

■ If you have not added the location of the javac compiler to your operating system's path variable, you will need to type the full path to the javac program.

**4** Type the name of the file that stores the Java code you want to compile, including the .java extension.

**5** Press Enter to compile the Java code.

■ If the Java code was successfully compiled, the command prompt re-appears.

■ The Java program is now ready to be executed. See page 22 to execute a Java program.

*Note: If an error message appears, the Java code was not successfully compiled.*

# EXECUTE A JAVA PROGRAM

Once the Java compiler has converted the source code for a Java program into bytecode, the program can be executed.

Bytecode must be processed by the Java interpreter before the code can be executed. When you execute a Java program, the Java interpreter first checks the bytecode to ensure the code is safe to execute and then it interprets and executes the instructions contained within the bytecode.

The instructions in the bytecode are executed by the Java interpreter in what is called the Java Virtual Machine, or JVM. The Java virtual machine enables Java programs to be executed in a controlled environment. This environment may also protect your computer from harmful code that may be included in Java programs.

The Java interpreter that comes with the Java SDK is called java and is typically stored in the c:\jdk1.3\bin directory.

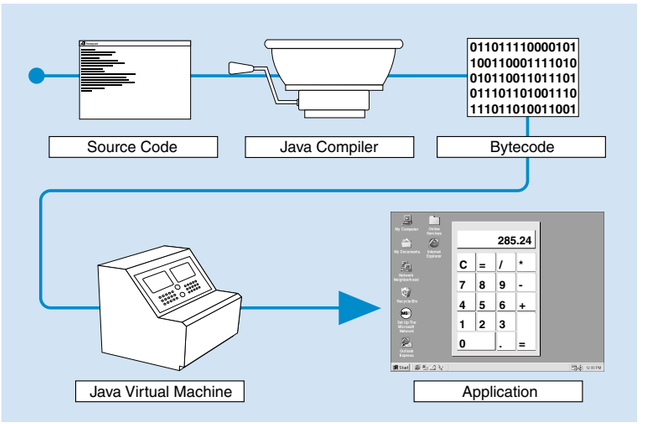
Like the Java compiler, the Java interpreter must be run at the command prompt. The Java interpreter is a stand-alone program, but the interpreter can also be integrated into other programs, such as Web browsers. This allows you to execute your Java programs on different platforms.

To evoke the Java interpreter, type the name of the interpreter followed by the name of the bytecode file. You should not type the .class extension. For example, to execute the instructions in the Program.class file, type **java Program**.

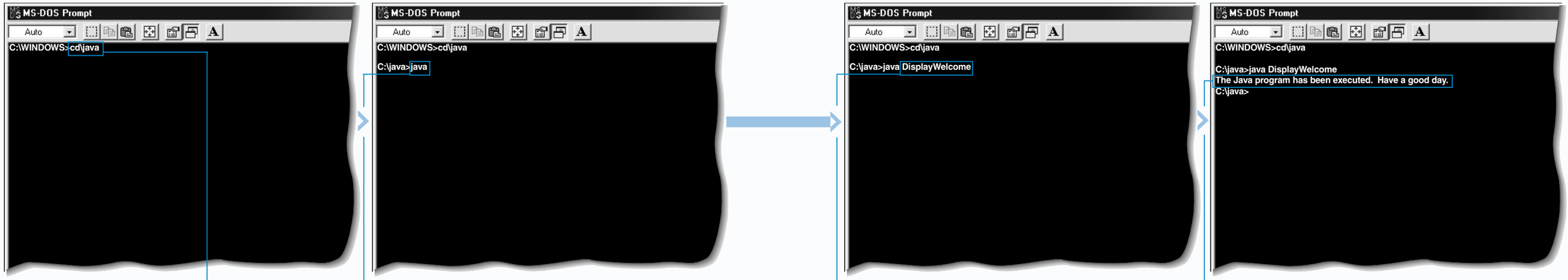
If the Java program executes successfully, the results of the program will be displayed. If the Java interpreter encounters any errors, it will stop executing the program. Most errors encountered at this stage are usually related to the use of incorrect filenames or paths.

## Extra

### How Java Code Is Processed




## EXECUTE A JAVA PROGRAM




**1** Open the window that allows you to work at the command prompt.

**2** Move to the directory that stores the bytecode for the Java program you want to execute.

**3** To execute the instructions in the bytecode using the Java interpreter, type **java.**

■ If you have not added the location of the Java interpreter to your operating system's path variable, you will need to type the full path to the Java interpreter.

**4** Type the name of the file that stores the bytecode for the Java program you want to execute.

**5** Press Enter to execute the program.

■ The results of the program are displayed on the screen.

# CREATE AN OBJECT

Objects are created using classes. An object usually contains at least one method that specifies the behavior of the object. Objects may also contain fields. For information about fields, see page 26.

The first step in creating an object is to create a class that will serve as a template for the object. In its simplest form, this type of class contains a class declaration, a method declaration and a method body. The method body contains code defining specifications for the object, such as the tasks the object will perform. This type of class is not executed as a stand-alone program and therefore does not need to use the `main` method. Before the class can be used to create an object, you must compile the code for the class.

Once you have created and compiled the class that serves as a template for an object, you can use a stand-alone program to create, or instantiate, the object. To instantiate an object, you must assign the object a name, which is used to access the object. You then use the `new` operator and the name of the class that defines the object to create the object. You can create multiple instances of an object within a program.

After creating an object, you can access a method of the object. This allows the object to take on the characteristics defined in the class. To access a method, you enter the name of the object and the name of the method, separated by a dot. For example, if you create an object named 'employee' that contains a method called DisplayName, you would access the method by entering employee.DisplayName.

## Extra

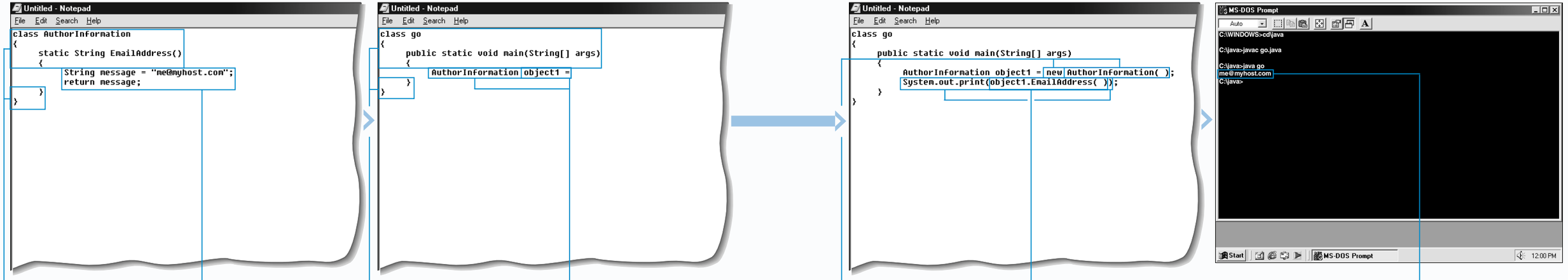
The directory you should use to store files you create for Java programs depends on the setup of your computer. In most cases, you should have a specific folder dedicated to Java program development. Regardless of the location you choose, you should always store the class file that defines an object and the file that instantiates the object in the same directory.

It is very rare that an object will be made up of only a single method. In most cases, objects are more complex, containing a wide range of related methods and fields that dictate the behavior of the object.

The fields and methods of an object are also referred to as members. Fields and methods that are available when an object is instantiated and are unique to that object are called instance members.

The ability to create objects is an important feature of JavaServer Pages. JavaServer Pages technology makes use of JavaBeans, which are a form of class file used to create objects. While it is possible to create JSP pages without knowing how to create objects, the flexibility and efficiency of your pages will be limited.

## CREATE AN OBJECT




### DEFINE THE OBJECT

**1** To create a class that will serve as a template for an object, enter the code that defines the class and method you want to use.

*Note: This class does not need to use the `main` method.*

**2** In the body of the method, type the code that defines the object you want to use.

**3** Save the class as a text file with the .java extension.

**4** Compile the Java code.

### INSTANTIATE THE OBJECT

**5** To create a stand-alone program that will instantiate the object, enter the code that defines the class and method you want to use.

**6** In the body of the method, type the name of the class you created in step 1.

**7** Type a name for the object, followed by **=**.

**8** Type **new**.

**9** Type the name of the class you created in step 1, followed by **( )**.

**10** To access the method of the object, type the name of the object followed by a period. Then type the name of the method you created in step 1, followed by **( )**.

**11** Type the code that uses the object.

**12** Compile the Java code and then execute the program.

The results of using the object are displayed.

# WORK WITH OBJECT FIELDS

You can create an object field, also referred to as data field, to hold information about an object. The information contained in an object's fields determines the properties and attributes of the object.

When objects of the same class are created, the objects have the same methods, but some or all of the object fields may hold different information. For example, each object created from the Employee class may have an object field called empNumber that stores the unique employee number for each object.

Object fields must be declared in the class body, outside of any methods. This allows the field to be used as soon as the object is created. You must specify an access modifier for an object field you create, as well as the data type that the field will store. For information about access modifiers, see page 16. For information about data types, see page 30.

Most object fields are created with an initial value. You can later change the value of an object field as you would change the value of a variable. Changing the value of an object field may change the way some of the methods of the object behave. Object fields may also hold constant data which cannot be changed.

You can use the dot operator (.) to access an object field in a program. When specifying the object field, the field name is separated from the object name by a dot, such as object.field. The object name is the name that was given to the object when it was created.

Unlike methods, object field names are not followed by parentheses. It is possible to have object fields and methods that share the same name in a program.

## Apply It

You can set a default value for an object field by using a constructor. A constructor is a special type of method that is always executed each time the class is accessed and an object is created. This makes constructors useful for performing initialization tasks for new objects, such as setting up a connection to a database. A constructor method must have the same name as the class for which it is the constructor.

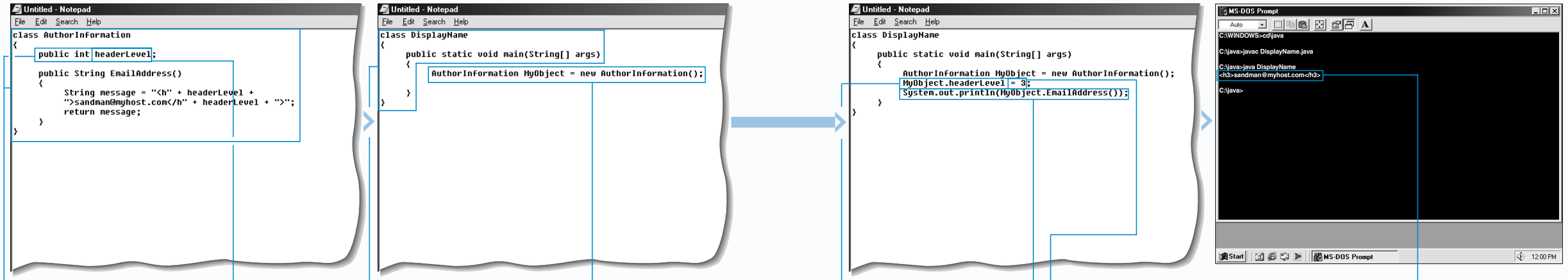
**Example:**

```
class AuthorInformation
{
      public int headerLevel;

      public AuthorInformation()
      {
            headerLevel = 3;
      }

      public String EmailAddress()
      {
            String message = "<h" + headerLevel +
            ">sandman@myhost.com</h" + headerLevel + ">";
            return message;
      }
}
```

## WORK WITH OBJECT FIELDS




### CREATE A FIELD

1 Create a class that will serve as a template for an object.

2 In the body of the class, type the access modifier and data type for the object field you want to create.

3 Type the name of the object field.

4 Save the class as a text file with the .java extension and then compile the Java code.

### USE AN OBJECT FIELD

5 To create a stand-alone Java program, enter the code that declares the class and `main` method.

6 In the body of the `main` method, type the code to create an object using the class you created in step 1.

7 To assign a value to an object field, type the name of the object followed by a dot. Then type the name of the field.

8 Type = followed by the value you want to assign to the object field.

9 Type the code that uses the object field.

10 Compile the Java code and then execute the program.

■ The results of using the object field are displayed.