| [Print](javascript:window.print()) | [Close](javascript:parent.window.close()) |

**Course Transcript**

Java SE7 Fundamentals: Introduction to Java

**Introduction to Java**

| [1. Introducing the Java Technology](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#t3) |

| [2. Working with IDEs and the Product Life Cycle](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#t5) |

| [3. Thinking in Objects](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#t7) |

**Java Language and Programs**

| [1. Java Language Fundamentals](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#t34) |

| [2. Examining and Modifying Java Programs](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#t35) |

| [3. Creating, Compiling, and Debugging a Java Program](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#t36) |

Introducing the Java Technology

Learning Objectives

*After completing this topic, you should be able to*

* *identify the key concepts of the Java programming language*
* *identify the Java technology groups*

**1. Key concepts of Java**

To put things in perspective, Java is the single most widely used development language in the  
world today, with over 9 million developers saying they spend at least some of their time  
developing in Java, according to a recent Evans Data study. That’s out of a world population of  
about 14 million developers.

Graphic

*The graphic shows a pie chart with the amount of people involved with software development. A section of it shows the amount of developers that use Java.*

Java is used in many different products:

**desktops**

Java is used in desktops worldwide. By the end of 2010, 1.1 billion desktops were running Java. Between August 2009 and 2010, 930 million JRE, or Java Runtime, downloads occurred. Between August 2009 and 2010, there were 9.5 million JDK, or Java Development Kit, downloads.

**mobile phones, and**

Java is also used extensively in mobile phones, with approximately three billion mobile phones using it. At the end of 2009, 31 times more Java phones shipped every year than Android and Apple combined. Also, all five of the top five manufacturers used Java.

**TV and cards**

Java is also used in Java TV and Cards. All the Blu-ray players that are made use Java. In 2010, 71.2 million people connected to the web using Java-powered devices. 1.4 billion Java cards were also manufactured every year by the end of 2010.

The Java programming language, formerly known as Oak, began in 1991 as part of a research project at Sun. The aim of the research project was to develop a programming language that would bridge the communication gap between many consumer devices, such as video cassette recorders, and televisions.   
  
A highly skilled team of software developers, known as the Green team, under the leadership of James Gosling, wanted to create a programming language that enabled consumer devices with different central processing units to share the same software enhancements.

This initial concept failed after several deals with consumer device companies were unsuccessful. The Green team was forced to find another market for their new programming language. Fortunately, the World Wide Web was becoming popular and the Green team recognized that the Oak language was perfect for developing web multimedia components to enhance web pages.   
  
These small applications, called applets, became the initial use of the Oak language, and programmers using the Internet adopted what became the Java programming language. The big break for Java came in 1995, when Netscape incorporated Java into its browser.

Graphic

*Multiple computers worldwide are connected to each other by the Internet.*

The key concepts of the Java programming language are that it's

* object orientated
* distributed
* simple
* multi-threaded
* secure, and
* platform independent

The Java programming language is an object-oriented programming, or OOP, language because one of the main goals of the Java programmer is to create objects – pieces of autonomous code – that can interact with other objects to solve a problem.  
  
OOP started with the SIMULA-67 programming language in 1967, and has led to popular programming languages such as C++. Java is loosely based on C++.

In contrast, procedural programming focuses on a sequence of steps.

Graphic

*The graphic shows five steps. The first step is connected to the second step. The second step is connected to the third step, and so on.*

Procedural programming places emphasis on the sequence of coding steps required to solve a problem, whereas OOP stresses the creation and interaction of objects.

OOP has several key characteristics, including

**modularity**

The source code for an object can be written and maintained independently of the source code for other objects. After it is created, an object can be easily passed around inside the system.

**information-hiding**

By interacting only with an object’s methods, the details of its internal implementation remain hidden from the outside world.

**code re-use, and**

If an object already exists – perhaps written by another software developer – you can use that object in your program. This allows specialists to implement, test, or debug complex, task-specific objects, which you can then trust to run in your own code.

**pluggability and debugging ease**

If a particular object has problems, you can remove it from your application and plug in a different object to replace it. This is similar to fixing mechanical problems in the real world. If a bolt breaks in a machine, you can replace the bolt in order to fix it.

The Java programming language is considered a distributed language because it provides support for distributed network technologies. These distributed network technologies include Remote Method Invocation, or RMI, Common Object Request Broker Architecture, or CORBA, and Universal Resource Locator, or URL.  
  
Additionally, the dynamic class loading capabilities of Java technology allow pieces of code to be downloaded over the Internet and executed on a computer.

Note

*The terms Java technology and Java programming language do not refer to the same thing. Java technology refers to a family of Java technology products of which the programming language is only one part.*

The Java programming language is simple because the designers removed some of the complex or obscure programming constructs that are found in other popular programming languages. As an example, the Java programming language does not allow programmers to directly manipulate pointers to memory locations – a complex feature of the C or C++ programming languages. Instead, the Java programming language only allows programmers to manipulate objects using object references.

The programming language also uses a feature called a garbage collector to monitor and to remove objects that are no longer referred to.

Another feature that makes the Java programming language simple is that a Java Boolean can only have a true or false value, unlike some other languages where Boolean is represented by zero and one.

The Java programming language supports multithreading, which allows several tasks to run  
concurrently, such as querying a database, performing long-running and complex calculations, and displaying a user interface. Multithreading allows a Java technology program to be very efficient in its use of system resources.

Question

Match each feature of object-oriented programming with its description.

**Options:**

1. Modularity
2. Information-hiding
3. Code re-use
4. Pluggability

**Targets:**

1. The source code for an object can be independent of the source code for other objects
2. The details of an object’s internal implementation remain invisible to the outside world
3. You can use existing objects in your program
4. If an object is problematic, you can replace it with another object

Answer

*The source code for an object is independent of the source code that is created and written for other objects. An object moves around the inside the system easily after it is created.*

*Information-hiding allows you to hide information by only allowing interaction with an object’s methods. Any details of its internal implementation are hidden to outside users.*

*You can use existing objects in your program. Specialists can use these objects to implement, test, or debug complex, task-specific objects. These objects can then be used in your own code.*

*If an object is causing problems, it can be removed and replaced with another object. This is very similar to the replacement of problematic parts in a machine in the real world.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

Java technology programs are secure because the Java programming language and the Java environment in which Java programs run, use security measures to protect programs from attacks. These measures include

Graphic

*An applet is prevented from sending a file to a desktop computer.*

* prohibiting distributed programs, such as applets, from reading and writing to a hard disk of a computer
* verifying that all Java technology programs contain valid code
* supporting digital signatures, which allows a company or person to sign code so that another person receiving the code can verify the legitimacy of the code, and
* prohibiting the manipulation of memory with pointers

Programs written in most languages usually require you to make numerous modifications to them in order to run on more than one type of computing platform. This platform dependence is because most languages require you to write code specific to the underlying platform.

Graphic

*C code uses the Solaris OS C compiler to create Solaris OS binary. C code uses the Linux C compiler to create Linux binary. C code uses the Microsoft Windows C compiler to create Microsoft Windows binary.*

Popular programming languages, like C and C++, require programmers to compile and link their programs, resulting in an executable program unique to a platform. A compiler is an application that converts a program that you write into a CPU-specific code called machine code.

These platform-specific files, or binary files, are often combined with other files, such as libraries of prewritten code, using a linker to create a platform-dependent program.

This platform-dependent program is called an executable and can be executed by an end user.

Unlike languages such as C and C++, the Java programming language is platform independent. A Java technology program can run on several different CPUs and operating system combinations, such as Solaris OS on a SPARC chip, MacOS X on an Intel chip, and Microsoft Windows on an Intel chip, usually with few or no modifications.

Java technology programs are compiled using a Java technology compiler. The resulting format of a compiled Java technology program is platform-independent Java technology bytecode instead of CPU-specific machine code.   
  
After the bytecode is created, it is interpreted by a bytecode interpreter called the virtual machine, or VM. A virtual machine is a platform-specific program that understands platform-independent bytecode and can execute it on a particular platform.   
  
For this reason, the Java programming language is often referred to as an interpreted language, and Java technology programs are said to be portable or executable on any platform. Other interpreted languages include Perl.

Graphic

*Java code uses a Java compiler to create Java bytecode.*

A virtual machine gets its name because it is a piece of software that runs code, a task usually  
accomplished by the CPU or hardware machine. For Java technology programs to be platform independent, a virtual machine called the Java Virtual Machine, or JVM, is required on every  
platform where your programming will run.   
  
The Java Virtual Machine is responsible for interpreting Java technology code, loading Java classes, and executing Java technology programs.

However, a Java technology program needs more than just a Java Virtual Machine to execute. A Java technology program also needs a set of standard Java class libraries for the platform. Java class libraries are libraries of prewritten code that can be combined with the code that you write to create robust applications. Combined, the JVM software and Java class libraries are referred to as the Java runtime environment, or JRE. Java runtime environments are available from Oracle for many common platforms.

Note

*Some modifications might be required to make a Java technology program platform independent. For example, directory names might need to be altered so that they use the appropriate delimiters, such as forward and backward slashes, for the underlying operating system.*

Question

Identify features of the Java programming language.

**Options:**

1. Complex programming constructs found in other languages have been removed
2. Multithreading allows for efficient use of system resources
3. Programs are written specific to the underlying platform
4. It uses a sequence of coding steps to solve a problem

Answer

***Option 1:****Correct. The designers of the Java programming language have removed some of the complex or obscure programming constructs found in other popular languages. For example, Java does not allow programmers to directly manipulate pointers to memory locations.*

***Option 2:****Correct. The Java programming language supports multithreading, which allows several tasks to run concurrently and uses system resources more efficiently.*

***Option 3:****Incorrect. Java is platform independent and does not require you to code specific to the underlying platform. A Java program can run on several different CPUs and OS combinations.*

***Option 4:****Incorrect. Java is an object-oriented programming or OOP language. OOP differs from procedural programming because procedural programming stresses the sequence of coding steps required to solve a problem, whereas OOP stresses the creation and interaction of objects.*

**Correct answer(s):**

1. Complex programming constructs found in other languages have been removed  
2. Multithreading allows for efficient use of system resources

**2. Java technology groups**

Oracle provides a complete line of Java technology products ranging from kits that create Java technology programs to emulation or testing environments for consumer devices, such as cellular phones. All Java technology products share the foundation of the Java language.

Java technologies, such as the Java Virtual Machine, are included in three different groups of products, each designed to fulfill the needs of a particular target market. These are the groups:

Graphic

*Java Technology Product Groups are servers, desktop, embedded, TV mobile and card. Each of these product groups use different key APIs, which are based upon different Java platforms. Servers use the Java EE API, which is based upon the Java SE platform. Desktops use the JavaFX API, which is based upon the Java SE platform. Embedded chips do not use a key API, but use technology from both the Java SE and Java ME platforms. TVs use BD-J, and Java TV, which are based upon the Java ME platform. Mobile phones use the MSA API, which is based upon the Java ME platform. Cards do not use a key API, but are based upon the Java card platform. All of these product groups are based upon the Java language.*

* Java SE
* Java ME, and
* Java Card

Note

*The JavaFX API is a rich client for creating user interfaces for your Java program. The MSA API is the mobile software application used to create user interfaces on portable devices.*

Each of Java edition includes a Java Development Kit, or JDK. The JDK allows programmers to create, compile, and execute Java technology programs on a particular platform.

Note

*A JDK is also known as a Software Development Kit, or SDK.*

The Java Platform Standard Edition, or Java SE, is used to develop applets that run within web browsers and applications that run on servers and desktop computers. For example, you can use the Java SE JDK to create a word-processing program for a personal computer.

Graphic

*A PC and web browser are each connected to the Java Platform. The web browser uses applets and the PC uses applications.*

Note

*Applets and applications have several differences. The main difference between applets and applications is that applets are launched inside a web browser, while applications are launched within an operating system.*

The Java Platform Enterprise Edition, or Java EE, is used to create large enterprise, server-side, and client-side distributed applications. For example, you can use the Java EE JDK to create a web shopping, or eCommerce, application for a retail company’s web site.

Java EE is built on top of the Java SE Platform, extending it with additional APIs supporting the  
needs of large-scale, high-performance enterprise software.   
  
The APIs are packaged and grouped to support different kinds of containers, such as a web container for web-based applications, a client container for thick clients, and the EJB container to run workhorse Java components. Some of the kinds of functionality supported by the different APIs include objects and UI in Java SE APIs and integration, persistence, transactions, and security in Java EE APIs.

Java Platform Micro Edition, or Java ME, is used to create applications for resource-constrained  
consumer devices. For example, you can use the Java ME JDK to create a game that runs on a  
cellular phone. Blu-ray Disc Java applications and Java TV use the same SDK as Java ME.

Java Card enables devices with a small memory -footprint to run applets. It is used for many different types of functionality including

* identity
* security
* transactions, and
* mobile phone SIMs

Question

Match the project to the Java technology product best suited to develop it.

**Options:**

1. Creating a web shopping application for a retail company’s web site
2. Creating a word-processing program for a personal computer
3. Creating a game that runs on a cellular phone

**Targets:**

1. Java EE
2. Java SE
3. Java ME

Answer

*Java EE is used to create large enterprise, server-side, and client-side distributed applications. For example, you can use the Java EE JDK to create a web shopping, or eCommerce, application for a retail company’s web site.*

*Java SE is used to develop applets  that run within web browsers and applications that run on servers and on desktop computers. For example, you can use the Java SE JDK to create a word-processing program for a personal computer.*

*Java ME is used to create applications for resource-constrained consumer devices. For example, you can use the Java ME JDK to create a game that runs on a cellular phone.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

**3. Summary**

In this topic, you've learnt the key concepts of Java. You've also learnt about Java technology groups.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#top)

Working with IDEs and the Product Life Cycle

Learning Objectives

*After completing this topic, you should be able to*

* *recognize how IDEs help speed development time*
* *sequence the stages in the product life cycle*

**1. IDEs and development time**

Setting up your Java development environment is a simple task. The JDK, or Java Development Kit, is available for free from the Oracle Java web site.

To install the Java JDK you need to

* go to "www.oracle.com/java"
* download the version for your platform
* follow the installation instructions
* set your Java PATH, and
* compile and run a sample Java application

When you download and install the Java SE Development Kit, these items are installed:

* Java runtime environment, or JRE
* a Java Virtual Machine, or JVM, for the platform you use
* Java class libraries for the relevant platform
* a Java technology compiler
* additional utilities, such as utilities for creating Java archive, or JAR, files, and for debugging Java technology programs, and
* examples of Java technology programs

Note

*The compiler is located in the /jdkversion/bin folder.*

An Integrated Development Environment, or IDE, is a tool that can assist you with developing your Java application. There are several IDEs that you can use, for example, NetBeans, JDeveloper, or Eclipse.  
  
These IDEs include features such as full integration, easy deployment, and easy project development. There is also built-in source control and a smart editor.  
  
Integrated development environments help speed up development time by automating simple tasks, using code auto-completion, and integrating debugging.

Graphic

*The NetBeans IDE Start screen is open. It consists of a menu bar and a toolbar.The Start Page tab has links in two sections: Learn and Discover as well as Demos and Tutorials.*

Note

*Smart editors in IDEs include features such as code completion and semantic coloring and highlights that are not supported when a file is opened in a text editor such as Notepad.*

The NetBeans IDE 7.0 is compatible with and supports Java SE 7. The IDE's installer requires   
a JDK to install the IDE on your system, because NetBeans is a Java application.  
  
However, you can add additional JDK versions after NetBeans is in place, and you can choose which JDK version to use when you create a NetBeans project. The IDE is available with specific downloads that support various Java technologies.   
  
For example, you might choose to download and install the Java SE bundle only to install NetBeans and Java SE, or you can choose the NetBeans All bundle.

Graphic

*NetBeans IDE has a variety of bundles. Each bundle supports different features. The Java SE bundle supports the NetBeans Platform SDK, and Java SE. The Java EE bundle supports the NetBeans Platform SDK, Java SE, Java EE, and the Glassfish and Apache Tomcat servers. The C/C++ bundle supports C/C++. The PHP bundle supports PHP. The All bundle supports all the previously mentioned features, as well as Java ME, Java Card 3 Connected, and Groovy.*

Note

*This course uses the NetBeans IDE for the course demonstrations and exercises. However, once you become familiar with an IDE, you can easily transfer your skills to any similar IDE of your choice.*

NetBeans is available for free from NetBeans.org. There are several different bundles available.

In NetBeans, you work within the context of a project. A project consists of

* an organized group of source files and associated metadata
* project-specific properties files
* an Ant build script and run settings, and
* the tools you'll need to write, compile, test, and debug your application

You can create a main project with subprojects, and you can link projects through dependencies. So getting started is as easy as giving your project a name.

Graphic

*The NetBeans IDE New Project Wizard is open. The wizard is divided into two sections. The first section displays the current step of the wizard. The second section displays the step details. In this example, the Choose Project step is displayed. You can select from a list of categories, and then select a project type.*

After you tell NetBeans the name of a  
new project, it then

* creates a source tree with an optional skeleton class inside
* creates a folder for unit tests
* sets classpaths for compiling, running, and testing
* sets the Java platform on which the project runs, and
* creates an Ant build script – build.xml – which contains instructions that the IDE uses when you perform commands on your project, such as compile or run

Question

Integrated development environments offer several features that can help to speed development time. Identify these features.

**Options:**

1. External debugging
2. Facilitating the deployment of applications
3. External source code control
4. Automating simple tasks

Answer

***Option 1:****Incorrect. IDEs offer integrated debugging tools to ease Java application development.*

***Option 2:****Correct. IDEs help speed development time by facilitating the compilation and deployment of applications.*

***Option 3:****Incorrect. IDE features include full integration, easy deployment, smart editors, easy project development, and built-in source code control.*

***Option 4:****Correct. IDEs help to speed development time by automating simple tasks and providing full integration.*

**Correct answer(s):**

2. Facilitating the deployment of applications  
4. Automating simple tasks

**2. Stages of the product life cycle**

The product life cycle, or PLC, is an iterative process used to develop new products by solving problems.

A PLC is split into several stages.

**Analysis**

Analysis is the process of investigating a problem that you want to solve with your product. Among other tasks, analysis consists of clearly defining the problem you want to solve, the market niche you want to fill, or the system you want to create. The boundary of a problem is also known as the scope of the project. Identifying the key subcomponents that compose your overall product is also a key part of analysis. Good analysis of the problem leads to a good design of the solution and to decreased development and testing time.

**Design**

Design is the process of applying the findings you made during the analysis stage to the actual design of your product. The primary task during the design stage is to develop blueprints or specifications for the products or components in your system.

**Development**

Development consists of using the blueprints created during the design stage to create actual components.

**Testing**

Testing consists of ensuring that the individual components, or the product as a whole, meet the requirements of the specification created during the design stage.  
  
Testing is usually performed by a team consisting of people other than those who actually developed the product. A team ensures that the product is tested without any bias on behalf of the developer.

**Implementation**

Implementation consists of making the product available to customers

**Maintenance**

Maintenance consists of fixing problems with the product and re-releasing the product as a  
new version or revision

**End-of-life**

Although the PLC does not have a separate stage for the start of a concept or project, it does have a stage for the end of a project. End-of-life, or EOL, consists of carrying out all of the necessary tasks to ensure that the customers and employees are aware that a product is no longer being sold and supported, and that a new product is available.

The PLC is an important part of product development because it helps to ensure that products are created and delivered so that time-to-market is reduced, the quality of a product is high, and the return on investment is maximized. Developers who do not follow the PLC often encounter problems with their products that are costly to fix and that could have been avoided.

Question

Rank the phases in the product life cycle.

**Options:**

1. Implementation
2. Analysis
3. Design
4. Maintenance
5. Testing
6. Development

Answer

**Correct answer(s):**

**Analysis is ranked**

Analysis is the first phase and involves investigating a problem that you want to solve with your product.

**Design is ranked**

Design is the second phase and consists of applying the findings you made during the analysis stage to the action design of your product.

**Development is ranked**

Development is the third phase and consists of using the blueprints created during the design stage to create actual components.

**Testing is ranked**

Testing is the fourth phase and consists of ensuring that the individual components, or the product as a whole, meet the requirements of the specification created during the design phase.

**Implementation is ranked**

Implementation is the fifth phase and consists of making the product available to customers.

**Maintenance is ranked**

Maintenance is the sixth phase and consists of fixing problems with the product and re-releasing it as a new version or revision.

**3. Run a Java program in the command line**

You need to compile and run a Java program at the command line. The Java program CalcAverage.java has already been created. Before you can compile the program, you must first set the PATH variable for the DOS session before running the program.  
  
To begin, from the Windows **Start** menu, you select **Start - Run**.

Graphic

*The Start menu is open, and the Run field selected.*

Then you enter cmd in the Open field and click **OK**.

Graphic

*The Run dialog box is open. The command cmd has been entered into the Open field.*

After you've opened the command prompt, you perform the following steps:

Graphic

*The command prompt is open.*

Code

Microsoft Windows 7 [Version 5.1.2600]                                       
(C) Copyright 1985-2001 Microsoft Corp.                                     
  
C:\Users\Admin\**\_**

* type cd D:\labs\les02 and press **Enter**   
    
  **Code**  
  Microsoft Windows 7 [Version 5.1.2600]                                       
  (C) Copyright 1985-2001 Microsoft Corp.                                         
    
  D:\winnit\Profiles\Administrator>cd D:\labs\les02
* type dir at the command prompt to check the directory contents   
    
  **Code**  
  D:\labs\les02>dir  
   Volume in drive D is WINNT  
   Volume Serial Number is FC5C-B059  
    
   Directory of D:\labs\les02  
    
  06/08/2012  02:27 PM    <DIR> .  
  06/08/2012  02:27 PM    <DIR> ..  
  04/22/2012  03:05 PM 641 CalcAverage.java  
        1 File(s) 641 bytes  
        2 Dir(s) 459, 428, 489, 216 bytes free
* append the location of the Java executables – the compiler and the runtime executable – to the System PATH variable, using the specified code   
  *The directory contents is visible.*   
    
  **Code**  
  D:\labs\les02>Path = %Path%;D:\Program Files\Java\jdk1.7.0\bin
* confirm that the PATH was changed correctly by typing PATH at the next prompt, and   
    
  **Code**  
  D:\labs\les02>PATH  
  PATH=D:\Program Files\Java\jdk1.7.0\bin;d:\winnt\system32;d:\winnt;d:\winnt\system32\wbem;c:dos;c:\ntinst.ad;c:utils;c:\  
  detect;c:\net;D:\Program Files\Java\jdk1.7.0\bin
* type javac CalcAverage.java to compile the file and press **Enter**   
    
  **Code**  
  D:\labs\les02>javac CalcAverage.java

After a slight delay, the prompt will return. You have compiled the file and you now want to run the CalcAverage.java program.  
  
Look for the compiled class, CalcAverage.class, by listing the contents of the directory again. To do this, type dir and press **Enter**.

Code

D:\labs\les02>dir  
 Voleume in drive D is WINNT  
 Volume Serial Number is FC5C-B059  
  
 Directory of D:\labs\les02  
  
06/08/2012  04:57 PM <DIR> .  
06/08/2012  04:57 PM <DIR> ..  
06/08/2012  04:57 PM    921 CalcAverage.class  
04/22/2012  03:05 PM    641 CalcAverage.java  
2 File(s)   1,562 bytes  
2 Dir(s)  459,428,415,488 bytes free

You run the CalcAverage program by invoking the java runtime executable. You do not need to use the .class extension of the class. Type java CalcAverage and press **Enter**. The program will prompt you to enter three integers. Type three integers separated by spaces and then press **Enter**.

Graphic

*The integers are 2, 46, and 88. The average is 45.*

Code

D:\labs\les02>java CalcAverage  
  
Enter 3 Integers separated only by spaces: (example 20 30 40)  
2 46 88  
  
Average = 45

**4. Run a Java program in NetBeans IDE**

Now you want to compile and execute a Java program using NetBeans IDE. In addition, you want to explore some features of an IDE that let you develop programs more quickly and easily than if you use a command line. To do this, you follow these steps:

* when NetBeans opens, clear the **Show On Startup** checkbox and close the Start Page, and   
  *The NetBeans IDE Start screen is open on the Start Page tab.*
* create a NetBeans project that includes the CalcAverage.java file in its project source folder, by selecting **File - New Project** from the main NetBeans menu   
  *From the File menu you can also access options such as Open Project, Project Properties, and Page Setup.*

Using the New Project wizard, you do the following:

* In the Choose Project step, select **Java** from the Categories column and **Java Project with Existing Source** from the Projects column, and click **Next**.   
  *The New Project Wizard is divided into two sections. The first section displays the steps of the wizard, and the second section displays the details of the step. In this case, the Categories and Projects options are listed. The Categories are Java, Maven, NetBeans Modules, and Samples. The Java category is selected. The Project options are Java Application, Java Desktop Application, Java Class Library, Java Project with Existing Sources, and Java Free-Form Project.*
* In the Name and Location step of the wizard, enter Practice02 for the Project Name, clear the **Set as Main Project**checkbox, and click **Next**.   
  *The second step, Name and Location, is open. The options available include Project Name, Project Folder, Build Script Name, Use Dedicated Folder for Storing Libraries, and Set as Main Project.*
* In the Existing Sources step of the wizard, add C:\labs\les02 to the Source Packages Folder panel by clicking **Add Folder**, and browsing to the desired directory. You then click **Finish**.   
  *The third step, Existing Sources, is open. You can add and remove the Source Package Folders. You can also add and remove the Test Package Folders.*

You are now prompted with the message "The specified package folders contain compiled class files."  
  
Click **Delete** to delete the CalcAverage.class file that was generated when you compiled the CalcAverage.java file from the DOS console. NetBeans will generate a new class file for you.

Graphic

*The New Java Project with Existing Sources dialog box is open. It informs you that the specified package folder contains compiled class files (\*.class), and because the IDE generates compiled class files in the "build" folder, it is recommended that you delete the existing compiled class files. You're prompted to confirm if you want to delete the class files now. Three buttons are available: Delete, Ignore, or Cancel.*

The contents of the project are now displayed in the Projects window in the upper left pane of NetBeans. Select the **Projects** tab if necessary to view the Projects window. The project name is at the root node. You can expand the nodes beneath that to find CalcAverage.java.

Graphic

*The Project is open. The interface contains a menu bar, and toolbar. On the left are the Projects, Files, Services tabs. The Projects tab is selected. Below these tabs is the Navigator. On the right is the main window, which contains the Start Page, and the Tasks window.*

Now you modify the properties of this project to set the Source/Binary Format property to JDK 7. This will allow you to use any new language features of Java SE 7 without getting an error message from NetBeans.  
  
To do this, you follow these steps:

* Select **File -** **Project Properties (Practice02)** from the main menu. You can also right-click the project name in the Projects window and select **Properties**.   
  *The File menu is open.*
* Select **Sources** in the Categories pane and then set the Source/Binary Format drop-down list to **JDK 7**.   
  *The Project Properties window is open. The Categories pane includes categories such as Sources, Libraries, Build, Run, Application, and Formatting. Two panes are available: Source Package Folders and Test Package Folders. Each pane has associated Add Folder, Remove, Move Up, and Move Down buttons. There is also a Source/Binary Format drop-down list box and an Includes/Excludes button.*

NetBeans allows you to specify the lowest Java platform version with which the generated code should be compatible. For instance, if you had not changed this setting to JDK 7, you would have seen error messages when using any of the core language changes included in JDK 7. NetBeans would warn you that the code would be incompatible with an earlier version.  
  
Remember that when you compiled and ran this java file from the command prompt, you had to manually set the PATH to point to the JDK 7 installation. When you use an IDE, it automatically sets a default JDK runtime environment for each NetBeans project.

You continue by performing the following steps:

* Confirm that the Java Platform setting for the Practice02 project is JDK 7.   
  *The Source/Binary Format drop-down list displays JDK 7.*
* Select the **Libraries** node in the Categories column. On the right, JDK 7 is listed as the Java Platform for this project. You could select a different platform, or JDK version, if you wished. This is assuming other platforms had been properly installed on this machine.   
  *The Java Platform drop-down list box displays JDK 7.*
* Click **OK** to save the change you made to the project properties.

Then you do the following three steps:

* To determine or change the default Java Platform for NetBeans, select **Tools -** **Java Platforms** from the main menu.   
  *Other options in the Tools menu include Add to Favorites, Analyze Javadoc, and Plugins.*
* Note the different versions of the JDK that have been installed on the PC using the Java Platform Manager. This window shows all versions of the JDK that have been properly installed on this machine. In our case, only the JDK 7, or JDK 1.7, has been installed so it is marked as the default platform in the Platforms column. On the right, the directory location for the JDK 7 installation is shown.   
  *The Java Platform Manager contains a Platforms pane that lists two options: JDK 1.7 and JDK 1.7 (Default) platforms. Two text boxes are also available: Platform Name and Platform Folder. There are Classes, Sources and Javadoc tabs. The Classes tab is selected and the JDK 1.7 (Default) classes are displayed.*
* Then close the Java Platform Manager window when you have finished examining it by clicking the **Close**button.

To view and edit the code for the CalcAverage.java file, you double-click it in the Project's window. It opens in the Editor pane. Notice the color coding used by the editor. For example, keywords are in blue, string literals are in red. This makes working with, and reading your code, much easier.

Graphic

*CalcAverage.Java is open in the Editor pane.*

You then do the following:

* In the Projects window, right-click the **CalcAverage.java**file, and choose **Compile File**.   
  *The shortcut menu also has options like Run File, Save as Template, and BeanInfo Editor.*
* Assuming you had no compilation errors, you can now find the .class file by clicking the **Files**tab and expanding **Practice02 -** **build - classes**. If you made changes to the java file, the **Save**button would have become enabled. By default, compilation occurs automatically with a save.
* Next you click the Projects window again. Right-click the file and choose **Run** **File**. The output from the program appears in the Output window. Enter the three integer values in the line beneath the output message and press **Enter**.   
  *The file has been run and an output has been created.*

Now you have seen how to run a simple Java program using both the DOS command prompt and the NetBeans IDE.

Graphic

*The file has been run and an output has been created.*

**5. Summary**

In this topic, you've learnt how IDEs help speed development time and you've learnt how to sequence the stages in the product life cycle.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#top)

Thinking in Objects

Learning Objectives

*After completing this topic, you should be able to*

* *identify the features of objects, attributes, and operations*
* *recognize factors to consider when designing and modeling a class*

**1. Objects, attributes, and operations**

When you want to build something, a house or a software application,  you generally determine the scope of the item first, such as the outer dimensions of the item – for instance in the case of a house, its height, width, depth – the way the item must "fit" into its environment, or lot size, and so on. After that, you can begin to divide the item into its major components, which are usually recognized as nouns or "things," such as floor, roof, or kitchen.

Graphic

*The graphic shows a room, with the dimensions 11 feet by 15 feet.*

When considering components that make up an item, its useful to group related objects. A taxonomy is a classification of related objects or organisms that have similar characteristics, or features, called *attributes*. For example, fins or gills, operations, the ability to swim, and the ability to walk on two feet.

Attributes are distinguishing characteristics or features of an organism from a similar taxonomy – for example, a dorsal fin is an attribute for a whale.  
  
Values represent the current state of an attribute. For example, one whale, the blue whale, has a  
small dorsal fin while another whale, the orca or killer whale, has a large dorsal fin. Large and  
small are values for the fin attribute in the whale taxonomy.

You may need to use Object-Orientated Analysis, or OOA, to analyze a problem for which you you need to provide a software solution. For example, Duke's Choice sells clothing from their catalog. Business is growing 30% per year, and they need a new order entry system.   
  
Duke's Choice produces an online catalog of clothing every three months and e-mails it to subscribers. Each shirt in the catalog has an item identifier, or ID, one or more colors, each with a color code, one or more sizes, a description, and a price.  
  
Duke's Choice accepts all credit cards. Customers can call Duke's Choice to order directly from a customer service representative, or CSR, Customers can also complete an online order form on the Duke's Choice web site.

Graphic

*A table displays, with the column headers Price, Description, Size, and Color code.*

As the customer enters the order online, each item's availability, or quantity-on-hand, is verified. If one or more items are not currently available in Duke's Choice's warehouse, then the item is marked as backordered until the items arrive at the warehouse.   
  
After all of the items are available, payment is verified and the order is submitted to the warehouse for assembly and for shipping to the customer’s address. When the order is received, the customer is given an order ID, which is used to track the order throughout the process. Orders that are phoned in are entered by a CSR.

Graphic

*The graphic shows the process that a customer must go through when ordering an item online.*

Note

*In a true analysis, you would work side-by-side with a company getting details about each aspect of how the company does its business. This case study outlines only a small portion of the information needed to create a system for Duke's Choice.*

Because Java is an object-oriented programming language, one of the main goals of the Java technology programmer is to create objects to build a system or, more specifically, to solve a problem. The scope of the problem you will solve is referred to as a problem domain. Most projects start by defining the problem domain, by gathering customer requirements, and by writing a statement of scope that briefly states what you, the developer, want to achieve.   
  
For example, a scope statement for the Duke's Choice project might be: "Create a system allowing the online order entry method to accept and verify payment for an order."  
  
After you have determined the scope of the project, you can begin to identify the objects that will interact to solve the problem.

To validate objects in a problem domain, you must first identify the properties of all objects:

**objects can be physical or conceptual**

A customer's account is an example of a conceptual object, because it is not something you can physically touch. An automated teller machine, or ATM, is something many people touch every day and is an example of a physical object.

**objects have attributes, and**

Objects have attributes, or characteristics, such as the size, name, or shape, that represent the state of the object. For example, an object might have a color attribute. The value of all of an object’s attributes is often referred to as the object’s current state. An object might have a color attribute with the value of red and a size attribute with a value of large.

**objects have operations**

Objects have operations, or things they can do, such as setting a value, displaying a screen, or increasing speed. These represent the behavior through which the state of the object can be altered. Operations usually affect an object's attributes. The operations that an object performs are often referred to as its behavior. For example, an object might have an operation allowing other objects to change the object's color attribute from one state to another, such as from red to blue.

Object names are often nouns, such as "account" or "shirt." Object attributes are often nouns too, such as "color" or "size", but in the context of its value, the attribute is an adjective of the noun describing the object.  
  
Object operations are usually verbs or noun-verb combinations, such as "deliver" or "submit order." Your ability to recognize objects in the world around you will help you to define objects better when you approach a problem using object-oriented analysis.

Graphic

*Examples of object names are mouse, screen, and keyboard. Examples of object operations are deliver and submit order.*

An attribute with only two states is referred to as a Boolean attribute. A clock can be an object. A clock has at least one attribute – current time – that has a value, which is the current hours, minutes, and seconds. There's also a dial that allows you to set the value of the current time – this is an operation.  
  
A chair can be an object. A chair has at least one attribute, such as height, that has a value, height in inches, and it can have a lever allowing another object, such as a person, to change the value of the height, which is an operation. An instructor can be an object. A student can be an object.

Graphic

*An image of a whale displays. Its attributes include color – blue – and size – large. Its operations include eat, migrate, dive, and communicate.*

Question

Identify the statements that accurately describe objects, attributes, and operations.

**Options:**

1. Objects have operations to represent the state of the object
2. Objects can be physical or conceptual
3. Objects have attributes to represent the behaviors of the object
4. Operations usually affect an object's attributes

Answer

***Option 1:****Incorrect. Objects have operations - the things they can do - such as setting a value, displaying a screen, or increasing speed, which represent the behavior through which the state of the object can be altered.*

***Option 2:****Correct. Objects can be physical or conceptual. A customer's account is an example of a conceptual object, because it is not something you can physically touch. An automated teller machine, or ATM, is something many people touch every day and is an example of a physical object.*

***Option 3:****Incorrect. Objects have attributes, or characteristics, such as size, name, shape, that represent the state of the object.*

***Option 4:****Correct. Operations usually affect an object's attributes. For example, an object might have an operation allowing other objects to change the object's color attribute from one state to another, such as from red to blue.*

**Correct answer(s):**

2. Objects can be physical or conceptual  
4. Operations usually affect an object's attributes

When looking for additional criteria for recognizing objects, you need to assess the relevance to the problem domain, and the independent existence. To determine if the object is relevant to the problem domain, you need to assess

* if the object exists within the boundaries of the problem domain
* if the object is required for the solution to be complete, and
* if the object is required as part of an interaction between a user and the solution

Note

*Some items in a problem domain can be attributes of objects or can be objects themselves. For example, temperature can be an attribute of an object in a medical system, or it can be an object in a scientific system that tracks weather patterns.*

For an item to be an object and not an attribute of another object, it must exist independently in the  
context of the problem domain. Objects can be connected and still have independent existence. In the Duke's Choice case study, a customer and an order are connected but are independent of each other, so both would be objects.

When evaluating potential objects, ask yourself if the object needs to exist independently, rather  
than being an attribute of another object. Identifying objects in a problem domain is an art, not a science. Any object could be a valid object if it has relevance to the domain of a problem and has the characteristics of an object, but this does not mean that it is a good object. Regardless, the person who models the system or solution must understand the entire system.

After performing an analysis on the system, some nouns that are probably not appropriate objects are identified. These could include a fax, verification, and payment.

After you identify the objects, you then specify their attributes and operations. Attributes define the state of an object. They can be data, such as order ID and customer ID for an Order object, or they can be another object, such as the customer having an entire Order object as an attribute rather than just the order ID.   
  
Operations are behaviors that usually modify the state of an attribute. For example, an order can be printed, can have an item added or deleted, and so on. The customer or CSR would be initiating those actions in real life, but the operations belong to the Order object.

Graphic

*Examples of attributes are ID and order object. Examples of operations are delete item and change ID.*

An attribute can be a reference to another object. For example, the Customer object can have an  
attribute that is an Order object. This association might or might not be necessary, depending on  
the problem you are trying to solve.  
  
You can use attribute and operation names that clearly describe the attribute or operation. For example, the Customer object contains an Order attribute. The asterisks denote attributes that are other objects.

Graphic

*The Customer object has these attributes: customer ID, name, address, phone number and email address, and order. The Customer object has the operation assign a Customer ID. The Order object has these attributes: order ID, data, shirt(s), total price, form of payment, CSR, and status. The operations are calculate order ID, calculate the total price, add shirt to order, remove shirt from order, and submit the order. The order attribute of the Customer object is connected to the Order object. These operations have asterisks: order, shirt(s), form of payment, CSR, and status.*

For example, Duke's Choice Case Study include Order, Shirt, and Customer objects and some possible attributes and operations are displayed here.

Graphic

*The three objects are Order, Shirt, and Customer. The attributes for Order are order ID, data, \*Shirt(s), total price, \*Form of payment, \*CSR, and status. The operations for Order are calculate order ID, calculate the total price, add shirt to order, remove shirt from order, and submit the order.  
  
The attributes for Shirt are shirt ID, description, size, and color code. The operations for Shirt are calculate shirt ID, and display shirt information.   
  
The attributes for Customer are customer ID, name, address, phone number, email address, and \*Order. The operation for Customer is assign a customer ID.*

Supplement

*Selecting the link title opens the resource in a new browser window.*

**Learning Aid**

Use the learning aid [Case Study Solutions](javascript:doWindow('./la_jsef_a01_T7_001_frame.html')) to learn more about objects, attributes, and behaviors in the Duke's Choice case study.

Question

After you identify objects, you then specify their attributes and operations. For an Order object, identify the operations.

**Options:**

1. Order ID
2. Delete item
3. Print order
4. Customer ID

Answer

***Option 1:****Incorrect. Order ID is an example of an object attribute.*

***Option 2:****Correct. Operations are actions, such as deleting an item or printing an order.*

***Option 3:****Correct. Printing an order or deleting an item are examples of operations.*

***Option 4:****Incorrect. Customer ID is an example of an object attribute.*

**Correct answer(s):**

2. Delete item  
3. Print order

**2. Designing and modeling a class**

Identifying objects helps you design the class or blueprint for each of the objects in a system. For example, window manufacturers often create a single blueprint for each of the styles of windows they create. These blueprints define a range of colors and styles that can be selected when the  
window is purchased.  
  
These blueprints are then the basis for any number of windows with any number of combinations of color and style. In object-oriented design terms, each object or window created using the class – generic blueprint – is called an instance of a class. More specifically, each object created from a class can have a specific state – or value – for each of its attributes, but will have the same attributes and operations.

Graphic

*Whale attributes include Dorsal fin, Color, and Size. The attributes for each of these whales are different. For two different whales, these have different values.*

Note

*The American Heritage Dictionary defines the word class as "a group whose members have certain attributes in common."*

Classes and objects are often used in the field of biology. For example, a marine biologist studying sea creatures is often asked to categorize sea creatures in a family, or class, of sea creatures. In OOA terms, each animal, such as a blue whale, in a family, such as whales, can be considered an object instance of the whale class.

Objects differ from classes.

**Class**

A class is the way you define an object. Classes are descriptive categories, templates, or blueprints. A Shirt could be a class defining all shirts to have a shirt ID, size, color code, description, and a price.  
*For example, the Shirt class has these attributes: shirt ID, price, description, size and color code. These are its operations: calculateShirt ID () and displayShirtInformation ().A class is the way you define an object.*

**Object**

Objects are unique instances of classes. The large blue polo shirt that costs $29.99 with shirt ID 62467-B is an instance of the Shirt class, as is the small green shirt with the same price and shirt ID 66889-C, or the patterned shirt for $39.99, ID 09988-A. You can also have two Shirt objects in memory with exactly the same attribute values.  
*A red shirt and a blue shirt are two examples of Shirt objects.*

In the Java programming language, attributes are represented using variables, and operations are represented using methods. Variables are the Java programming language mechanism for holding data. Methods are the Java programming language mechanism for performing an operation.

The first phase of the design stage consists of visually organizing or modeling a program and its classes. Each class in a design should be modeled so that it is enclosed in a box with the class name at the top, followed by a list of the attribute variables, including the range of possible values, and a list of methods.

This is the syntax for modeling a class:

* the *ClassName* is the name of the class
* the *attributeVariableName* is the name of the variable for an attribute
* the *range of values* is an optional range of values that the attribute can contain, and
* the *methodName* is the name of a method

This example contains a modeled Shirt object. The modeling technique is loosely based on a light version of the Unified Modeling Language, or UML, which is a tool to aid in the modeling process.

Code

shirtID  
price  
description  
size  
colorCode R=Red, B=Blue, G=Green  
calculateShirtID()  
displayInformation()

Variable and method names are written in a special short-hand called *camel case*. Camel case specifies that a variable or method, representing any multiple-word attribute or operation, starts with a lowercase letter, and subsequent words are capitalized. For example, this is how an operation such as "calculate the total price" is written. Furthermore, a set of closed parentheses indicates a method.

Code

calcTotalPrice()

Modeling classes are similar to modeling database structures. In fact, your object data can be stored in a database using the Java Database Connectivity, or JDBC, API. The JDBC API allows you to read and write records by using structured query language, or SQL, statements within your Java technology programs.

UML is used to model the objects, attributes, operations and relationships in object-oriented programs and model the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects.  
  
Using UML, you can choose nouns for all your objects, choose verbs for all your methods, or choose adjectives for all your attributes.

Question

Identify the correct statement regarding objects and classes.

**Options:**

1. An object is a blueprint for a class
2. An object and a class are exactly the same
3. An object is an instance of a class
4. An attribute cannot be a reference to another object

Answer

***Option 1:****Incorrect. A class is a blueprint for an object.*

***Option 2:****Incorrect. An object is simply an instantiation of a class, and a class serves as a blueprint for the object.*

***Option 3:****Correct. In object-oriented design terms, each object created using the class is called an instance of a class.*

***Option 4:****Incorrect. An attribute can be a reference to another object.*

**Correct answer(s):**

3. An object is an instance of a class

**3. Using object-orientated analysis**

You set out to analyze a case study and use object-oriented analysis to list the objects, attributes, and operations in the case study.  
  
Suppose a soccer league needs a system to track team and player standings. At any moment they want to be able to report a list of games played with results, a list of teams ranked by wins, and a list of players on each team ranked by goals scored.

You need to produce an object-oriented analysis for a Java technology application that tracks soccer scores. The program will track

* the list of players on each team ranked by goals scored
* the list of games played with results, and
* the list of teams in the league ranked by wins

Note

*You can think of the objects as nouns, attributes as adjectives, and operations as verbs. As an example, a Player is a noun, the player's name is an adjective that describes that noun, and add goal is a verb.*

The application should be able to generate statistics for teams, players, and seasons. To do this, follow these steps:

* open Notepad
* save the file as C:\labs\les03\oo-analysis.txt, and
* list the high-level classes that are included in this problem, using dashed lines to separate the objects, attributes, and operations.   
    
  **Code**  
  Player  
  -----------  
  id  
  name  
  number  
  \*Team  
    
  Team  
  -----------  
  id  
  name  
  Players\*  
  -----------  
  get ranked player

Note

*Instead of using a text editor, you can use a UML tool such as UMLet which can be downloaded for free from UMLet.com.*

**4. Designing a programming solution**

You continue designing the solution by using UML-like notation to represent the classes you previously identified.  
  
You now want to produce a design for each of the classes in the earlier system for tracking soccer scores.

When producing the design you need to remember to

* use camel case to name your classes, attribute variables, and methods
* identify a valid range of values for each attribute, where a range is known
* use square brackets to indicate an attribute that represents a collection of values (players[ ] ), and
* use parentheses to identify methods

Then you follow these steps:

Code

Player  
-----------  
id  
name  
number  
team  
  
Team  
-----------  
id  
name  
players[ ]  
-----------  
getRankedPlayers()  
  
Game  
-----------  
id  
team one score  
team two score  
goals [ ]  
-----------  
getResults()

* open the file that you created to document the high-level object-oriented analysis of the soccer program, and
* use the classes, variables, and operations that you previously identified to develop method names for the operations

Finally, your design lists all the classes, attributes, and behaviors you'll need for the program.

Graphic

*The objects are Player, Team, Game, League, and Goals.  
  
The attributes for Player are id, name, number and team. The attributes for Team are id name, and players[ ].  
  
The operation for Team is getRankedPlayers().  
  
The attributes for Game are id, team one score, team two score, and goals[ ]. The operation for Game is getResults().  
  
The attributes for League are teams[ ] and games[ ]. The operations for League are getGameResults(), and getRankedTeams().  
  
The attributes for Goals are id, team, player and time.*

Note

*You will still need to add or remove methods for each collection attribute and get/set methods for all other attributes.*

**5. Summary**

In this topic, you've learnt about the features of objects, attributes, and operations. You've also learnt about the factors you need to consider when designing and modeling a class.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#top)

Java Language Fundamentals

Learning Objective

*After completing this topic, you should be able to*

* *identify how to define a class using Java*

**1. Java programming classes**

Java technology programs consist of multiple classes made up of various components. You need to understand what developing and testing classes is all about. There are several ways to test a program. For example, you can test different components, known as a unit test, or you can test the entire item by seeing if it fits into its environment.

Classes are the blueprints that you create to define the objects in a program. For example, these are objects that might exist in the order entry program for a clothing catalog.  
  
A desktop application usually consists of one object, often called the controller object, main object, or test object, that is the starting point for your program. For example, an OrderEntry object might interact with one or more Window objects, Customer objects, Order objects, and so on while your program runs.

Graphic

*The Order Entry program comprises a series of objects, including Customer, Order, Shirt, Date, Window, and Button.*

Each object is an instance of a blueprint or class. For example, all Window objects are instances of Window classes. Some classes, such as the Window class used for creating graphical user interface, or GUI, windows, are general-purpose classes. These classes are provided to you as part of the Java technology API.

Other classes, such as the Shirt Class, might be unique to your particular program, so you must create them yourself.

Classes are composed of the Java technology code necessary to instantiate objects, such as Shirt objects. A Java class file can be divided into four separate sections:

* the class declaration
* field declarations
* methods, which are optional, and
* comments, which are also optional

Note

*Class attributes are called fields.*

Variables hold values, and the values can change during the course of the application. Fields are one type of variable. Local variables are another type of variable. Variables may also be initialized at declaration time.

Syntax

*variablename = value;*

Question

When defining a Java class, which code sections are required?

**Options:**

1. The class declaration
2. Field declarations
3. Methods
4. Comments

Answer

***Option 1:****Correct. When structuring a Java class, the class declaration is required.*

***Option 2:****Incorrect. When structuring a Java class, the use of field declarations is optional.*

***Option 3:****Incorrect. When structuring a Java class, the use of methods is optional.*

***Option 4:****Incorrect. When structuring a Java class, the use of comments is optional.*

**Correct answer(s):**

1. The class declaration

The programming code for a class is contained within a text file that must adhere to a certain structure. For example, the Shirt class has several fields and one method, displayInformation, for printing the values of the fields.

Graphic

*The relevant code is:  
  
Shirt  
  
and  
  
displayinformation*

Code

public class Shirt {  
  
    public int shirtID = 0; // Default ID for the shirt  
    public String description =   
        "-description required-"; // default  
    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = ’U’;  
    public double price = 0.0; // Default price for all shirts  
    // This method displays the values for an item  
    public void displayInformation() {  
        System.out.println("Shirt ID: " + shirtID);  
        System.out.println("Shirt description:" + description);  
        System.out.println("Color Code: " + colorCode);  
        System.out.println("Shirt price: " + price);  
    } // end of display method  
} // end of class

You can define a Java source using these symbols:

**braces**

Braces signify a block of code. The curly braces enclose the code for a specific method, or the code for an entire class, for example.

**parentheses**

Parentheses are used to indicate input data, also called *arguments*, that can be passed into a method.

**semicolons**

Semicolons signify the end of a statement.

**commas**

Commas can separate multiple arguments and values.

**single quotation marks**

Single quotation marks define single characters.

**double quotation marks, and**

Double quotation marks define a string of multiple characters.

**double forward slashes**

Double forward slashes indicate a single-line comment.

You must declare a class for each class that you've designed for the problem domain. For each class, you must write a class declaration.

Syntax

[*modifiers*] class *class\_identifier*

A class declaration is made up of three components.

Syntax

[*modifiers*] class *class\_identifier*

*The [modifiers] variable determines the accessibility that other classes have to this class. The modifiers variable is optional, as indicated by the square brackets, and can be public, abstract, or final.*

*The class keyword tells the compiler that the code block is a class declaration. Keywords are words that are reserved by the Java programming language for certain constructs.*

*The class\_identifier is the name that you give to the class. Class names should be nouns, in mixed case, with the first letter of each word capitalized – for example, MyClass. Class names should contain whole words. Avoid acronyms and abbreviations unless the abbreviation is much more widely used than the long form, such as JVM or UML.*

For example, the Shirt class uses a class modifier of public, followed by the class keyword, followed by the class name of Shirt.

Graphic

*The relevant code is:  
  
public class Shirt*

Code

public class Shirt{  
    public double price;  
  
    public void setPrice(double priceArg){  
        price = priceArg;  
    }  
}

Question

Which class declaration adheres to the class-naming guidelines?

**Options:**

1. class Shirt
2. public Class 501Pants
3. public Shirt
4. public Class Pants

Answer

***Option 1:****Correct. The class definition is followed by an open curly brace indicating the beginning of the class\_body, the attribute variables, and the methods that compose the class. The braces around the class\_body define where the class starts and ends.*

***Option 2:****Incorrect. This word class should not be capitalized.*

***Option 3:****Incorrect. The keyword class should be included in the class name.*

***Option 4:****Incorrect. The word class must not be capitalized.*

**Correct answer(s):**

1. class Shirt

Curly braces are used to enclose the entire body of code for the Shirt class and to enclose the body of code for the setPrice method. Parentheses are used to enclose the argument passed into the setPrice method. A semicolon is used at the end of the declaration of the price field.

Code

public class Shirt{  
    public double price;  
  
    public void setPrice(double priceArg){  
        price = priceArg;  
    }  
}

You can develop your classes so that the Java technology programming code that you write for each class is in its own text file or source code file.   
  
In the Java programming language, source code file names must match the public class name in the source code file, and must have a .java extension. For example, the Shirt class must be saved in a file called Shirt.java.

Syntax

*file\_name*.java

The field declarations and assignments block follows the first open curly brace. Generally, you  
set up all of the attribute variables for the class after this curly brace. You use a semicolon at the  
end of each line of code.

Code

public int shirtID = 0;  
public String description = “-description required-”;  
public char colorCode = ‘U’;  
public double price = 0.0;  
public int quantityInStock = 0;

You should put comments in every class that you create to make it easier to determine what the program is doing. Commenting is particularly important in longer programs developed by large teams where several programmers need to read the code.   
  
Commenting also helps with the maintenance of a program when new programmers need to determine what the code is doing.

Graphic

*The comments in the code are as follows:  
  
// Default ID for the shirt  
  
// Default price for all shirts  
  
and  
  
// The color codes are R=Red, B=Blue, G=Green*

Code

public int shirtID = 0; // Default ID for the shirt  
public double price = 0.0; // Default price for all shirts  
// The color codes are R=Red, B=Blue, G=Green

You can use one of the two main styles of comments.

**Single-line comments**

Single-line comments have a double forward-slash marker, which tells the compiler to ignore everything until the end of the current line. Many programmers also make their programs easier to read by using single-line comments to comment the first and last line of every class and method. For example, the Shirt class contains an end-of-line comment to indicate the end of the display method.   
  
**Code**  
public int shirtID = 0; // Default ID for the shirt  
public double price = 0.0; // Default price for all shirts  
// The color codes are R=Red, B=Blue, G=Green

**Traditional comments**

Traditional comments have a forward slash - asterisk character combination, which tells the compiler to ignore everything on all lines up to, and including, a comment termination marker of asterisk - forward slash.  
  
Programmers often use traditional comments to provide details for a large block of code. In long programs, it can be very difficult to find the ending braces of the class. Commenting the structure that each ending brace belongs to makes reading and fixing errors much easier.   
  
**Code**  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* Attribute Variable Declaration Section \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

There is a third type of comment called a documentation comment. You can use a Java technology tool, the Javadoc tool, to create documentation for any of your classes that will be used by other programmers. In fact, all of the class library documentation that comes with the Java SE JDK was created using the Javadoc tool.  
  
Documentation comments must begin with a forward slash and two asterisks and end with an asterisk and a forward slash. This traditional comment is also a valid documentation comment.

Code

/\*\*  
Attribute Variable Declaration Section  
\*/

Methods follow the attribute variable declarations in a class.

Code

public void displayInformation() {  
    System.out.println("Shirt ID: " + shirtID);  
    System.out.println("Shirt description:" +   
        description);  
    System.out.println("Color Code: " + colorCode);  
    System.out.println("Shirt price: " + price);  
    System.out.println("Quantity in stock: " +   
        quantityInStock);  
  
} // end of display method

Syntax

[*modifiers*] *return\_type* *method\_identifier* ([*arguments*]) {  
    *method\_code\_block*}

*The [modifiers] represent several unique Java technology keywords that modify the way  
methods are accessed. Modifiers are optional as indicated by the square brackets.*

*The return type indicates the type of value, if any, that the method returns. If the method returns a value, the type of the value must be declared. Returned values can be used by the calling method. Any method can return at most one value. If the method returns nothing, the keyword void must be used as the return type.*

*The method\_identifier is the name of the method.*

*The ([arguments]) represent a list of variables whose values are passed to the method for use by the method. Arguments are optional as indicated by the square brackets, because methods are not required to accept arguments. The parentheses are not optional. A method that does not accept arguments is declared with an empty set of parentheses.*

*The method\_code\_block is a sequence of statements that the method performs. A wide variety of tasks can take place in the code block or body of the method. For example, the Shirt class contains one method, the displayInformation method, which displays the values for the attributes of a shirt. Within the displayInformation method, several lines of code invoke the System.out.println method. This method is used to print out a specific string of data.*

Keywords are special, reserved words in the Java programming language that give instructions to the compiler. Keywords must not be used as identifiers for classes, methods, variables, and so on.  
  
The terms true, false, and null might seem like keywords, but they are actually literals. You can't use them as identifiers in your programs.

Graphic

*Some examples of keywords are abstract, break, byte, class, continue, enum, final, if, imports, ints, new, package, return, switch, throw,  void, and while.*

Supplement

*Selecting the link title opens the resource in a new browser window.*

**Learning Aid**

Access the learning aid [Java Keywords](javascript:doWindow('./la_jsef_a01_T3401_frame.html')) to review the list of Java technology keywords.

**2. Summary**

In this topic, you've learned how to define a class using Java.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#top)

Examining and Modifying Java Programs

Learning Objectives

*After completing this topic, you should be able to*

* *recognize how to test and execute a simple Java program*
* *use the features of the IDE debugger*

**1. Testing and executing a Java program**

Most of the classes that you create can't be executed or tested by themselves. Instead, you need to execute another class to create an object instance of your class before your class can be tested. You can use a test or main class to test each of your classes.  
  
Every test class should be named so that it can be recognized as the test class for a particular class you have written. Specifically, each test class name should consist of the name of the class you are testing, followed by the word Test. For example, the class designed to test the Shirt class is called ShirtTest.

Graphic

*The relevant code is:  
  
ShirtTest*

Code

class ShirtTest {  
    public static void main (String args[]) {  
    Shirt myShirt;  
    myShirt= new Shirt();  
    myShirt.displayInformation();  
    }  
}

Test classes have two distinct tasks to perform:

Code

class ShirtTest {  
    public static void main (String args[]) {  
    Shirt myShirt;  
    myShirt= new Shirt();  
    myShirt.displayInformation();  
    }  
}

* providing a starting point, called the main method, for your program, and
* creating an object instance of your class and test its methods

The main method is a special method that the Java Virtual Machine, or JVM, recognizes as the starting point for every Java technology program that runs from the command line or from a prompt.   
  
Any program that you want to run from a command line or a prompt must have a main method.  
  
Many of the Java technology classes that engineers create don't run within an operating system. For example, applets run within a web browser, and have their own unique starting method.

Syntax

public static void main (String [] args)

The main method adheres to the syntax for all methods described earlier.

Syntax

public static void main (String args[])

*The main method contains two required modifiers – public and static.*

*The main method does not return any values, so it has a return type of void.*

*The main method has a method identifier, or name, of main.*

*The main method accepts zero or more objects of type String (String args[]). This syntax allows you to type in values on the command line to be used by your program while it's running.*

Compiling converts the class files that you write into bytecode that can be executed by a Java Virtual Machine. Remember the rules for naming your Java source files. If a source file contains a public class, the source file must use the same name as the public class, with a .java extension. For example, the class Shirt must be saved in a file called Shirt.java.  
  
To compile the Shirt and ShirtTest source code files, you go to the directory where the source code files are stored, and enter the command for each .java file that you want to compile.

Code

javac Shirt.java

Syntax

javac *filename*

After the compilation has finished, and assuming no compilation errors have occurred, you should have a new file called classname.class in your directory for each source code file that you compiled.  
  
If you compile a class that references other objects, the classes for those objects are also compiled, if they haven't been compiled already. For example, if you compiled the ShirtTest.java file, which references a Shirt object, you have a Shirt.class and ShirtTest.class file.

Syntax

*classname*.class

When you've successfully compiled your source code files, you can execute and test them  
using the Java Virtual Machine. To execute and test your program, you go to the directory where the class files are stored. Then you enter the command for the class file that contains the main method, java ShirtTest, for example.

Code

java ShirtTest

Syntax

java *classname*

This command runs the ShirtTest class. As mentioned previously, the ShirtTest class  
creates an instance of the Shirt object, using the Shirt class. All Shirt objects have one  
method, the display method, which prints the values of their attribute variables.

Code

Shirt ID: 0  
Shirt description:-description required-  
Color Code: U  
Shirt price: 0.0  
Quantity in stock: 0

You can instead compile and run a program using an IDE. The **Save** option is the equivalent of using javac and a classname for all .java files in the project. You can use either **Run File** to run a single file, or the **Run** button to run the main class for the entire project, to do the equivalent of using the java command and specifying a classname.

Graphic

*The NetBeans IDE interface displays.*

Most Java editors check the code syntax and show alerts by using icons and red underlines where there are errors in the code. If you hover the cursor over the red bubble, the editor offers a suggestion for fixing the error.

To avoid syntax problems, you do the following:

* observe any red bubble indicators in the code editor to locate syntax errors
* have a semicolon at the end of every line where one is required, and
* have an even number of symbols such as curly braces, brackets, and quotation marks

Question

Identify the method syntax elements that the main method must adhere to.

**Options:**

1. It must contain the public and static modifiers
2. It must return a value
3. It must have a method identifier of main
4. It must accept one argument of type String

Answer

***Option 1:****Correct. The main method contains two required modifiers – public and static. Any program that you want to run from a command line or a prompt must have a mainmethod.*

***Option 2:****Incorrect. The main method does not return any values, so it has a return type of void.*

***Option 3:****Correct. The main method has a method identifier of main. The main method is a special method that the JVM recognizes as the starting point for every Java technology program that runs from the command line or a prompt.*

***Option 4:****Incorrect. The main method accepts zero or more objects of type String (String args[]). This syntax allows you to type in values on the command line to be used by your program while it is running.*

**Correct answer(s):**

1. It must contain the public and static modifiers  
3. It must have a method identifier of main

**2. Adding code to an existing Java program**

You want to create a new project from an existing Java source in NetBeans. To begin, you follow these steps:

Graphic

*The NetBeans IDE 7.0.1 toolbar has a menu bar and a toolbar. The File menu is open and includes options such as New Project, New File, Open Project, Open Recent Project, Close Project, and Open File.*

* select **File - New Project** from the main NetBeans menu to open the New Project wizard
* in the Choose Project step of the wizard, select **Java** from the Categories column, select **Java project with Existing Sources** from the Projects column, and click **Next**   
  *The New Project window is displayed at the Choose Project step of the wizard. The Categories column includes Java, Maven, NetBeans Modules, andSamples. Java is selected. The Projects column lists the available Java project options, which are Java Application, Java Desktop Application, Java Class Library, Java Project with Existing Sources, and Java Free-Form Project. A brief description of the selected project is also displayed.*
* in the Name and Location step of the wizard, enter the project name, deselect the **Set as Main Project** checkbox, and click **Next**   
  *In the Name and Location step of the wizard are options to specify the name and location of the new project. The Project Name is Practice04. The Project Folder is C:\Users\Admin\Documents\NetBeansProjects\Practice04, and the Build Script Name is build.xml. There is a Browse button alongside the Project Folder field. There are two checkboxes –Use Dedicated Folder for Storing Libraries and Set as Main Project.*
* in the Existing Sources step of the wizard, click **Add Folder**   
  *The New Java Project with Existing Sources dialog box has two panels – Source Package Folders and Test Package Folders. Each panel has an Add Folder button and a Remove button next to it.*
* browse to the desired directory of the existing source. Note that this can be any directory of your choosing so long as it contains your source. Then click **Open** to add it to the Source Packages Folder panel, and   
  *In the Browse Source Packages Folder dialog box, the folder les04 is selected in the Look In drop-down list. The File name field displays C:\labs\les04. There are Close, Open, and Cancel buttons.*
* click **Finish**   
  *C:\labs\les04 is listed under Source Package Folders in the Existing Sources section of the wizard.*

The contents of the project are now displayed in the Projects window within the upper left pane of NetBeans. You can select the **Projects** tab if necessary to view the Projects window. The project name is displayed at the root node.  
  
Now you can modify the properties of this project to set the Source/Binary Format property to JDK 7. This will allow you to use any new language features of Java SE 7 without getting an error message from NetBeans.

Graphic

*The root of the project displayed is named Practice04, which contains two folders called Source Packages and Libraries. The Source Packages folder contains the following files: Quotation.java, QuotationTest.java, Shirt.java, and ShirtTest.java. There are also two other tabs – Files and Services. Below this section is a Navigator pane.*

To set the Source/Binary Format property to JDK 7, you follow these steps:

* Right click the project node in the Projects window and select **Properties**. Alternatively, select **File - Project Properties (Practice04)** from the main menu.   
  *The right-click shortcut menu contains a variety of options to run, debug, and test your files, configure different options, move, rename, copy, delete and locate files.*
* In the Project Properties window, select **Sources** in the Categories column. Then set the Source/Binary Format drop-down list to JDK 7.   
  *Project Properties categories include Sources, Libraries, Build, Run, Application, and Formatting.*

You can now double-click the Quotation.java file in the Projects window to open it for editing.

Code

public class **Quotation** {  
  
  public String quote = "Welcome to Oracle, the new home of Java!";  
      
  public void **display**() {  
  
        // display the member variable here  
    System.*out*.println(quote);  
  }  
}

**display ()**

*You identify the method contained within this class as display.*

**quote**

*You identify the field contained within this class as quote.*

**System. *out* .println( quote );**

*In the display method, you use this statement to display the quote field.*

As you type the code, NetBeans’ code assist feature provides feedback and help whenever you pause in your typing. For instance, if you stop at some point at which the code, as is, would not compile successfully, it displays a red exclamation mark in the left margin.  
  
If you pause after typing the period following System or out, it gives you context-sensitive help in the form of a list of methods and fields that would be valid for the particular class to the left of the period. You can select from the list instead of typing.

Graphic

*The list of methods and fields includes append (char c), append (Charsequence cs, int i, int i1), CheckError (), Close (), equals (Object o), flush (), format (String string, Object... os), format (Locale locale, String string, Object... os), getClass (), hashCode (), notify (), notifyAll (), print (Object o), print (String string), print (boolean bin), and print (char c).*

With the statement added, you can save and compile Quotation.java using the **Save** button.

Graphic

*The Save button is located on the toolbar below the various menu options.*

Next you open the QuotationTest.java file in the editor and examine its main method. It creates an instance of the Quotation class and then calls its display method.

Code

 public class **QuotationTest** {  
  
  
    public static void ***main*** (String args[]) {  
  
      Quotation myQuotation = new Quotation();  
      
      myQuotation.display();  
    }  
}

You can run the QuotationTest class by right-clicking QuotationTest.java in the Projects window and selecting **Run File**.

Graphic

*The right- click shortcut menu includes these options: Open, Cut, Copy, Paste, Compile File, and Run File.*

The output from the display method appears in the Output window.

Graphic

*The Output for practice 04 (run) displays the updated property file, which contains the build, the removal of the build directory, and a notification that the build was a success.*

Note

*You were able to skip the Compile step because when you select****Run File****, NetBeans first compiles not only the class you selected to run, but also any referenced classes within that class, for example Quotation.java.*

Next, you follow these steps:

Graphic

*Quotation.java is selected.*

* edit the Quotation.java file to change the default value of the quote field, and
* run QuotationTest again to verify the output before closing Quotation.java and QuotationTest.java in the Editor pane   
  *The output displays the quote Jumping Java!, and a build successful message.*

**3. Creating and compiling a Java class**

You now want to create a Java class and compile it. You then want to create another Java class to test the previous class.

You can create a new Java class using the NetBeans wizard. Follow these steps:

Graphic

*The Practice04 project is open in NetBeans. It contains a Source Packages folder ,which contains the files Quotation.java, and QuotationTest.java. There is also a Libraries folder. There are three tabs: Projects, Files and Services.*

* From the main menu, select **File** - **New File**.
* The New File Wizard opens on step 1 – Choose File Type. Select **Java** in the Category column, select **Java class** in the File Types column, and click **Next**.   
  *Some options that you can choose from the Categories column are Java, Swing GUI Forms, JavaBeans Objects, AWT GUI Forms, JUnit, and Persistence. With Java selected, the File Types column includes options such as Java Class, Java Interface, Java Enum, Java Annotation Type, Java Exception, Java Package Info, and JApplet,. A description of the selected option is displayed below the columns.*
* In step 2 – Name and Location – enter Shirt as the Class Name and click **Finish**.   
  *In the Name and Location window, the fields contain the following information: the Class Name field displays Shirt; the Project field displays Practice04; the Location field displays Source Packages; the Package field is empty, and the Created File field displays C:\labs\les04\Shirt.java.*

The Java source file for the new class now appears in the editor ready for you to fill in the details. You can then enter the Java code syntax for the Shirt class.  
  
Once entered, you click the **Save** button to save and compile the Shirt class. Any red error icons in the left margin disappear after saving if there were no compilation errors. However, if necessary, fix any errors that appear in the output window and save again.

Code

public class **Shirt** {  
    public int shirtID = 0; // default ID for the shirt  
    public String description = "-description required-"; // default  
      
    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = 'U';  
    public double price = 0.0;  // default price for all shirts  
    public int quantityInStock = 0;  // default  
      
    // This method displays the values for an item  
    public void **displayShirtInformation**() {  
        System.*out*.println("Shirt ID: " + shirtID);  
        System.*out*.println("Shirt description: " + description);  
        System.*out*.println("Color Code: " + colorCode);  
        System.*out*.println("Shirt price: " + price);  
        System.*out*.println("Quantity in stock: " + quantityInStock);  
    } // end of display method  
}

The Navigator pane for the Shirt class now shows the Members view of the class. Color coding distinguishes between fields and methods. Both of these are considered members of the class.

Graphic

*The Members view for shirt.java in the navigation pane displays the method displayShirtInformation() for the Shirt class. Fields listed for the Shirt class include colorCode, description, price, quantityInStock, and shirtID.*

You need to repeat the previous steps to create another new class. Since this will be a Test class, it will need a main method. To create the class, you go through the New Class wizard, performing these steps:

Graphic

*The Practice04 project is open in NetBeans. It contains a Source Packages folder which contains the files Quotation.java, QuotationTest.java, and Shirt.java. There is also a Libraries folder present.*

* select **Java Main Class** instead of **Java Class** in the File Type step, and
* in the Name and Location step, enter ShirtTest as the name   
  *In the Name and Location step, the fields contain the following information: the Class Name is ShirtTest, the Project is Practice04, the Location is Source Packages, the Package field is empty, and the Created File is C:\labs\les04\ShirtTest.java.*

You replace the To Do: comment in the main method with code, and then click **Save** to save and compile the code.

Code

/\*  
 \* To change this template, choose Tools | Templates  
 \* and open the template in the editor.  
 \*/  
  
/\*\*  
 \*  
 \* @**author** Administrator  
 \*/  
public class **ShirtTest** {  
    /\*\*  
     \* @**param** args the command line arguments  
     \*/  
    public static void ***main***(String[] args) {  
        Shirt myShirt;  
        myShirt = **new** Shirt();  
        myShirt.displayShirtInformation();  
   }  
}

You confirm that the Shirt and ShirtTest classes have been successfully compiled by clicking the **Files** tab in the upper left pane of NetBeans to open the Files window and expand Practice04\build\classes. Shirt.class and ShirtTest.class are within the classes folder.

Graphic

*Under the Files tab, the tree structure for Practice04 contains folders for Practice04 - Source Packages, Practice04, and nbproject, as well as the build.xml file which contains the manifest.mf file. The Practice04 folder is expanded to display the build folder, which contains the classes folder. The classes folder contains the following files: Quotation.class, QuotationTest.class, Shirt.class, and ShirtTest.class.*

You return to the Projects window and run the ShirtTest class. You can find the output of the  
displayShirtInformation method in the Output window.

Code

run:  
Shirt ID : 0  
Shirt description: -description required-  
Color Code: U  
Shirt price: 0.0  
Quantity in stock: 0  
BUILD SUCCESSFUL (total time: 0 seconds)

You can now modify the values of ShirtID and price. For example, you enter 29.99 for the price and 5 for the shirtID.

Graphic

*The relevant code is  
  
shirtID = 5  
  
and  
  
price = 29.99*

Code

/\*\*  
 \*  
 \* @**author** Administrator  
 \*/  
public class **Shirt** {  
    public int shirtID = 5; // default ID for the shirt  
    public String description = "-description required-"; // default  
      
    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = 'U';  
    public double price = 29.99;  // default price for all shirts  
    public int quantityInStock = 0;  // default  
      
    // This method displays the values for an item  
    public void **displayShirtInformation**() {  
        System.*out*.println("Shirt ID: " + shirtID);  
        System.*out*.println("Shirt description: " + description);  
        System.*out*.println("Color Code: " + colorCode);  
        System.*out*.println("Shirt price: " + price);  
        System.*out*.println("Quantity in stock: " + quantityInStock);  
    } // end of display method  
} // end of class

Finally, you run the ShirtTest class again to verify that the modified values are shown in the Output window.

Code

run:  
Shirt ID : 5  
Shirt description: -description required-  
Color Code: U  
Shirt price: 29.99  
Quantity in stock: 0  
BUILD SUCCESSFUL (total time: 0 seconds)

**4. Using the IDE debugger features**

A debugger lets you place breakpoints in your source code, add field watches, step through your code, run into methods, take snapshots, and monitor execution as it occurs. You can also attach the NetBeans debugger to an already running process.

The debugger has several features:

**a configurable debugger**

The configurable debugger allows you to configure breaking/suspending behavior, specify Variable Formatters, and skip methods and packages by using Step Filters.

**a debugging window**

The debugging window integrates the Sessions, Threads, and Call Stack views.

**configurable breakpoints**

You can use configurable breakpoints to configure custom breakpoints to be triggered by conditions and events such as uncaught exceptions, class loading, or variable access.

**expression evaluation**

Expression evaluation allows you to evaluate Java-syntax expressions assigned to watches and conditional breakpoints live while stepping through your code.

**expression stepping**

You can use expression stepping to step over individual expressions within a statement.

**multi-session debugging, and**

You can use multi-session debugging to debug several processes at the same time.

**HeapWalker**

The HeapWalker allows you to watch references to objects while debugging a program.

For example, you have a program in the middle of a debug session. A breakpoint has been set on the highlighted line and the execution has stopped at that line. In the Variables window at the bottom of the screen, the fields of the class that is currently being executed are display. The class itself is referenced by the keyword this. During a debug session, you can change the values of these fields to try out different potential scenarios. This is helpful in solving logic problems.

Graphic

*The relevant code is  
  
shirtID = 5  
  
price = 29.99  
  
and  
  
System.out.println("Shirt ID: " + shirtID);  
  
The Variable window displays the following fields referenced by this for the Shirt class: colorCode, quantityInStock, price, shirtID, and description. There are also tabs for Breakpoints and Output.*

Code

    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = 'U';  
    public double price = 29.99;  // default price for all shirts  
    public int quantityInStock = 0;  // default  
      
    // This method displays the values for an item  
    public void **displayShirtInformation**() {  
        System.*out*.println("Shirt ID: " + shirtID);  
        System.*out*.println("Shirt description: " + description);  
        System.*out*.println("Color Code: " + colorCode);  
        System.*out*.println("Shirt price: " + price);  
        System.*out*.println("Quantity in stock: " + quantityInStock);

Virtually every Java IDE provides a debugger. They tend to offer the same core features and work very similarly. You can debug the ShirtTest program using the NetBeans debugger, for example. You'll need to set breakpoints, examine field values, and modify them as you step through each line of code.

Graphic

*The Practice04 project is open in NetBeans. It contains a Source Packages folder, which contains the files Quotation.java, QuotationTest.java, Shirt.java and ShirtTest.java. There is also a Libraries folder.*

To set a breakpoint in the ShirtTest class, you click in the left margin of the editor, next to a line of code. A pink square appears in the margin, indicating a breakpoint.

Graphic

*The square is in line 17 opposite the code  
  
myShirt = new Shirt();*

Code

/\*  
 \* To change this template, choose Tools | Templates  
 \* and open the template in the editor.  
 \*/  
  
/\*\*  
 \*  
 \* @**author** Administrator  
 \*/  
public class **ShirtTest** {  
    /\*\*  
     \* @**param** args the command line arguments  
     \*/  
    public static void ***main***(String[] args) {  
        Shirt myShirt;  
        myShirt = **new** Shirt();  
        myShirt.displayShirtInformation();  
   }  
}

You run the debugger by right-clicking on the ShirtTest file in the Projects window and selecting **Debug File**.  
  
The debugger starts the program and stops at the breakpoint. In the Editor panel, you can see a different icon that points with a green arrow to the line of code. This line of code has not yet been executed.

Graphic

*The arrow points to the line  
  
myShirt = new Shirt();*

Code

/\*  
 \* To change this template, choose Tools | Templates  
 \* and open the template in the editor.  
 \*/  
  
/\*\*  
 \*  
 \* @**author** Administrator  
 \*/  
public class **ShirtTest** {  
    /\*\*  
     \* @**param** args the command line arguments  
     \*/  
    public static void ***main***(String[] args) {  
        Shirt myShirt;  
        myShirt = **new** Shirt();  
        myShirt.displayShirtInformation();  
   }  
}

Several other changes display in the NetBeans window. A new toolbar appears, containing buttons that you use when debugging.  
  
You can move your cursor over each of the toolbar buttons to read the toolbar tip explaining what each button does.

Graphic

*The new toolbar buttons above the Editor panel are Stop, Pause, Continue the Execution, Step Over, Step Over Expression, Step Into, Step Out, and Run to Cursor.*

The toolbar buttons have different functions.

**Stop**

The **Stop** button stops the debugging session.

**Pause**

The **Pause** button pauses the execution of the debugger.

**Continue the Execution**

The **Continue the Execution** button continues either to the next breakpoint, or to the end of the program.

**Step Over**

The **Step Over** button moves the program forward to the next line of code in the current class – for example, the ShirtTest class.

**Step Over Expression**

The **Step Over Expression** button allows you to step over an entire expression to the next line of code in the current class

**Step Into**

The **Step Into** button allows you to step into another class referenced in this current line of code.

**Step Out**

The **Step Out** button allows you to step back out of a class that you stepped into.

**Run to Cursor**

The **Run to Cursor** button takes execution to the line of code where the cursor appears.

The panel at the bottom of the window changes to show debugging output and variables and other useful information during a debug session.  
  
In the Variables panel, you see all variables that are visible to the current class. Remember that the execution was stopped before the Shirt class object has been instantiated. Consequently, you do not see the myShirt variable in this panel.

You click the **Step Over** button to move to the next line of code.  
  
The arrow now points to the line of code that calls the displayShirtInformation method on the myShirt object. In the Values window, you can see the myShirt variable. You can expand it to see all of the fields of this Shirt object.  
  
At this point, the displayShirtInformation method has not yet been executed. You could change the values of the object’s fields right now, using the Variables window if you wanted to. However, instead, you will step into the myShirt object and change the values during the execution of the displayShirtInformation method.

Graphic

*The arrow points to the line  
  
myShirt.displayShirtInformation();*

To change the values during the execution of the displayShirtInformation method, you follow these steps:

* Click the **Step Into** button to step into the displayShirtInformation method. The arrow icon is pointing to the first executable line of code within the displayShirtInformation of the Shirt class.   
  *The relevant code is   
    
  System.out.println("Shirt ID: " + shirtID);*   
    
  **Code**  
  public class **Shirt** {  
    public int shirtID = 0; // default ID for the shirt  
    public String description = "-description required-"; // default  
    
    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = 'U';  
    public double price = 0.0; // default price for all shirts  
    public int quantityInStock = 0; // default  
      
    // This method displays the values for an item  
    public void **displayShirtInformation**() {  
      System.*out*.println("Shirt ID: " + shirtID);  
      System.*out*.println("Shirt description: " + description);  
      System.*out*.println("Color Code: " + colorCode);  
      System.*out*.println("Shirt price: " + price);  
      System.*out*.println("Quantity in stock: " + quantityInStock);
* You can expand this  to see the fields of this object in the Variables window.   
    
  **Code**  
  public class **Shirt** {  
    public int shirtID = 0; // default ID for the shirt  
    public String description = "-description required-"; // default  
    
    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = 'U';  
    public double price = 0.0; // default price for all shirts  
    public int quantityInStock = 0; // default  
      
    // This method displays the values for an item  
    public void **displayShirtInformation**() {  
      System.*out*.println("Shirt ID: " + shirtID);  
      System.*out*.println("Shirt description: " + description);  
      System.*out*.println("Color Code: " + colorCode);  
      System.*out*.println("Shirt price: " + price);  
      System.*out*.println("Quantity in stock: " + quantityInStock);
* In the Value column, click each field’s value and edit it to change the value. Ensure that you use the correct value for the data type expected and enclose any character data types with the type of quote mark indicated. After editing the final field, click the **tab** button so that the text you typed into the edit buffer is accepted.   
  *Values for the variables are as follows: description has a value of Tshirt, quantityInStock has a value of 3, price has a value of 29.99, and shirtID has a value of 5.*   
    
  **Code**  
    public int shirtID = 5; // default ID for the shirt  
    public String description = "Tshirt"; // default    
    
    // The color codes are R=Red, B=Blue, G=Green, U=Unset  
    public char colorCode = 'U';  
    public double price = 29.99; // default price for all shirts  
    public int quantityInStock = 3; // default  
      
    // This method displays the values for an item  
    public void **displayShirtInformation**() {  
      System.*out*.println("Shirt ID: " + shirtID);  
      System.*out*.println("Shirt description: " + description);  
      System.*out*.println("Color Code: " + colorCode);  
      System.*out*.println("Shirt price: " + price);  
      System.*out*.println("Quantity in stock: " + quantityInStock);
* Click the **Step Out** button to return to the next line of code in the ShirtTest class. The  
  displayShirtInformation method will have completed.   
  *The arrow now points to the line after the displayShirtInformation method, which is a closing brace }.*   
    
  **Code**  
  /\*  
   \* To change this template, choose Tools | Templates  
   \* and open the template in the editor.  
   \*/  
    
  /\*\*  
   \*  
   \* @author Administrator  
   \*/  
  public class **ShirtTest** {  
      /\*\*  
       \* @param args the command line arguments  
       \*/  
      public static void ***main***(String[] args) {  
          Shirt myShirt;  
          myShirt = new Shirt();  
          myShirt.displayShirtInformation();  
      }  
  }

The myShirt object field variables reflect the changes you made while in the method. You can click the **Continue** button to finish execution and end the debug session.

Graphic

*The variables for the myShirt object include description, colorCode, price, quantityInStock, and shirtID.*

Finally, you click the **Output** tab to view the output.

Code

debug:  
Shirt ID: 5  
Shirt description: -Tshirt-  
Color Code : U  
Shirt price: 29.99  
Quantity in stock: 3  
BUILD SUCCESSFUL (total time: 2 minutes 26 seconds)

Question

What debugger feature is triggered by conditions and events such as uncaught exceptions?

**Options:**

1. Debugging window
2. Configurable breakpoints
3. Expression stepping
4. Multi-session debugging

Answer

***Option 1:****Incorrect. The Debugging window integrates the Sessions, Threads, and Call Stack views of the debugger.*

***Option 2:****Correct. You can configure custom breakpoints to be triggers for conditions and events, such as uncaught exceptions, class loading, and variable access.*

***Option 3:****Incorrect. Expression stepping allows you to evaluate Java-syntax expressions assigned to watches and conditional breakpoints live while stepping through your code.*

***Option 4:****Incorrect. Multi-session debugging allows you to debug several processes at the same time.*

**Correct answer(s):**

2. Configurable breakpoints

**5. Summary**

In this topic, you've learned to recognize how to test and execute a simple Java program. You've also learned how to use the features of the IDE debugger.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#top)

Creating, Compiling, and Debugging a Java Program

Learning Objectives

*After completing this topic, you should be able to*

* *create a project in the NetBeans IDE*
* *recognize how to use NetBeans to create and compile a Java class and to create a test program*
* *recognize the features of the NetBeans debugger*

**1. Exercise overview**

In this exercise you’re required to examine and modify existing Java programs and also run them to test the program.

This involves the following tasks:

* creating a project from an existing source
* recognizing how to use NetBeans to create and compile a Java class and to create a test program, and
* recognizing the features of the NetBeans debugger

**2. Creating a project in the NetBeans IDE**

You're busy designing a program to help maintain stock controls in a bookstore.

Question

You want to create a new project using some files you created for a previous application. Which NetBeans IDE project type would you use?

**Options:**

1. Java Application
2. Java Project with Existing Sources
3. Java Class Library
4. Java Desktop Application

Answer

***Option 1:****Incorrect. This project type creates a new Java SE application including a main class for the project. You cannot use files that were created for a previous application with this option.*

***Option 2:****Correct. This project type imports existing files into a standard IDE project that you can then use as you need.*

***Option 3:****Incorrect. This creates a new Java SE library in a standard IDE project.*

***Option 4:****Incorrect. A Java Desktop Application creates a skeleton of a desktop application based on the Swing Application Framework.*

**Correct answer(s):**

2. Java Project with Existing Sources

Question

You want to add a method to the Student class to display basic information about the Student. Which code sample would you use to display firstName and lastName?

**Options:**

1. public void displayStudentName() {          
      System.out.println("Name: " +  
      firstName + " " + lastName);  
   }
2. public int displayStudentName() {          
      System.out.println("Name: " +  
      firstName + " " + lastName);  
   }
3. public void displayStudentName() {          
      System.out.println("Name: " +  
      firstName + " " + lastName)            
   }
4. public void displayStudentName() {          
      System.out.println(Name: " +  
      firstName + " " + lastName);            
   }

Answer

***Option 1:****Correct. This sample contains all the necessary code to print out the name of the student. It uses the System.out.println method to print the required details.*

***Option 2:****Incorrect. As the method returns nothing, the void keyword must be used in the method declaration.*

***Option 3:****Incorrect. Statements in Java must end with a semi-colon. Since this one doesn't, it would produce an error when you attempt to compile it.*

***Option 4:****Incorrect. This version is missing an opening quotation for the string passed to the println method.*

**Correct answer(s):**

1. public void displayStudentName() {          
   System.out.println("Name: " +  
   firstName + " " + lastName);  
}

**3. Compiling a class, and testing a program**

You want to create and compile a Java class for your stock control program. You then need to test your new class.

Question

You want to create a new class to represent a book. Which basic class dentition is correct?

**Options:**

1. public class Book [  
      public String author;  
      public String title;   
   }
2. public Class Book {  
      public String author;  
      public string title;   
   }
3. public class Book {  
      public String author;  
      public String title;   
   }
4. public class Book {  
      public String author  
      public String title;   
   }

Answer

***Option 1:****Incorrect. The body of the class definition must be contained inside of a set of curly braces.*

***Option 2:****Incorrect. The class keyword is lowercase and Java is case sensitive.*

***Option 3:****Correct. This snippet of code correctly defines a class Book along with two field variables.*

***Option 4:****Incorrect. The first statement that declares the field variable author is missing a semi-colon.*

**Correct answer(s):**

3. public class Book {  
   public String author;  
   public String title;   
}

Question

You want to add two more fields to the class to represent the price and quantity in stock of a book as integers. Which statements are valid?

**Options:**

1. public int price;  
   public int quantityInStock = 0;
2. public Char price;  
   public int quantityInStock = 0;
3. int price, quantityInStock = 0;
4. public int price; quantityInStock = 0;

Answer

***Option 1:****Correct. These statements declare the required fields and initialize the quantityInStock value to zero.*

***Option 2:****Incorrect. The data type char is not an appropriate choice as you want to use integers. In addition, char should be lowercase*

***Option 3:****Correct. You can declare variables of the same type on one line if separated by a comma.*

***Option 4:****Incorrect. You declare variables of the same type on one line only if separated by a comma, not a semi-colon.*

**Correct answer(s):**

1. public int price;  
public int quantityInStock = 0;  
  
3. int price, quantityInStock = 0;

Question

You want to create another class to test the Book class. The class should include a mainmethod to create a Book object and print its details. Which code segment completes the mainmethod declaration?

**Code**  
public static void main(<INSERT THE MISSING CODE>) {  
         
    }

**Options:**

1. String[] args
2. String[] args;
3. String() args
4. Stringv[] args

Answer

***Option 1:****Correct. This syntax allows you to type in values on the command line to be used by your program while it's running, and it correctly completes the main method declaration.*

***Option 2:****Incorrect. The semicolon indicates the end of a statement, not the end of a parameter.*

***Option 3:****Incorrect. The main method accepts zero or more objects of type String as indicated by the square brackets.*

***Option 4:****Incorrect. The main method accepts parameters of type String.*

**Correct answer(s):**

1. String[] args

Question

You've completed the code for the test class and saved it as BookTest.java. Which statement will compile the file?

**Options:**

1. javac BookTest.java
2. java BookTest
3. javac BookTest
4. javadoc BookTest.java

Answer

***Option 1:****Correct. The javac command instructs the compiler to compile the BookTest.java file.*

***Option 2:****Incorrect. This statement uses the java command that would run the compiled file.*

***Option 3:****Incorrect. When invoking the compile, you must include the .java file extension.*

***Option 4:****Incorrect. The javadoc command is used to generate documentation based on your classes for other programmers to use.*

**Correct answer(s):**

1. javac BookTest.java

**4. Recognizing NetBeans debugger features**

You now want to check your program for any potential problems using the NetBeans IDE debugger.

Question

Identify features of the NetBeans IDE debugger.

**Options:**

1. Configurable debugger
2. Command line debugging window
3. Configurable breakpoints
4. Single-session debugging

Answer

***Option 1:****Correct. This feature lets you configure breaking or suspending behavior, specify Variable Formatters, and skip methods and packages by using Step Filters.*

***Option 2:****Incorrect. The Debugging window integrates the Sessions, Threads, and Call Stack views. It's part of the interface of the NetBeans IDE – not the command line.*

***Option 3:****Correct. You can configure custom breakpoints to be triggered by conditions and events.*

***Option 4:****Incorrect. The debugger allows you to perform multi-session debugging and debug several processes at the same time.*

**Correct answer(s):**

1. Configurable debugger  
3. Configurable breakpoints

Question

You have set a breakpoint in the test program for the Book class. You start the debugger, which starts the program and stops at the breakpoint you have set. How would you move the program forward to the next line of code in the current class?

**Options:**

1. Use the **Run to Cursor** button
2. Select the **Step Into** button
3. Select the **Step Over** button
4. Press the **Continue** button

Answer

***Option 1:****Incorrect. The****Run to Cursor****button takes execution to the line of code where the cursor appears.*

***Option 2:****Incorrect. The****Step Into****button allows you to step into another class referenced in the current line of code.*

***Option 3:****Correct. The****Step Over****button moves the program forward to the next line of code in the current class.*

***Option 4:****Incorrect. The****Continue****button continues the execution, either to the next breakpoint or to the end of the program.*

**Correct answer(s):**

3. Select the **Step Over** button

You've created a project in the NetBeans IDE, recognized how to use NetBeans to create and compile a Java class and create a test program, and recognized the features of the NetBeans debugger.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/jl_jsef_a01_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited