Week 1: Review of Java Programming Basics

Sources:
• Chapter 2 in Supplementary Book (Murach’s Java Programming)
• Appendix A in Textbook (Carrano)
Outline

• Objectives
• A simple Java Program
• Data-types
• Control Structures
Objectives

Applied

- Given the specifications for an application *that requires only the language elements presented in this chapter*, write, test, and debug the application.

- Given the Java code for an application that uses any of the language elements presented in this chapter, *explain what each statement in the application does*.

- Given the name of a package and a class, look it up in the documentation for the Java API.
Objectives (cont.)

Knowledge

- Name two types of comments that are provided by Java and explain how to code them.
- Given a list of names, identify the valid identifiers for Java classes and variables.
- Given a list of names, identify the ones that follow the naming recommendations for classes presented in this chapter.
- Given a list of names, identify the ones that follow the naming recommendations for variables presented in this chapter.
- Describe the difference between a main method and other methods.
- Name three things you can assign to a numeric variable.
- Distinguish between the int and double data types.
Objectives (cont.)

- Explain what happens when an arithmetic expression uses both int and double values.
- Name three things you can assign to a String variable.
- Explain what an escape sequence is and when you would use one.
- Explain what a static method is and how it differs from other methods.
- Explain what importing a class means and when you typically do that.
- Explain what the System.out object can be used for.
- Explain what a Scanner object can be used for.
- Explain what a Boolean expression is and when you might use one.
- Explain how an if/else statement works and what it allows you to do.
Objectives (cont.)

- Explain what it means for a variable to have block scope.
- Explain how a while loop works and what it allows you to do.
- Describe the difference between testing an application and debugging an application.
- Describe the difference between a compile-time error, a runtime error, and a logical error.
Outline
• Objectives
• A simple Java Program
  • Comments, identifiers, keywords
  • Specifying a class with a main method, …
• Data-types
• Control Structures
A sample application

```java
import java.util.Scanner;
public class InvoiceApp {
    public static void main(String[] args) {
        // display a welcome message
        System.out.println("Welcome to Invoice Calculator");
        // get the input from the user
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter subtotal: ");
        double subtotal = sc.nextDouble();
        // calculate the discount amount and total
        double discountPercent = .2;
        double discountAmount = subtotal * discountPercent;
        double invoiceTotal = subtotal - discountAmount;
        // format and display the result
        String message = "Discount percent: \n" +
                         discountPercent + "\n" +
                         "Discount amount:  " + discountAmount + "\n" +
                         "Invoice total:    " + invoiceTotal + "\n";
        System.out.println(message);
    }
}
```
A block comment

/*
 * Author:  J. Murach
 * Purpose: This program uses the console to get a subtotal
 * from the user, and it calculates the discount amount and
 * total and displays them.
 */
Valid identifiers

<table>
<thead>
<tr>
<th>Identifier1</th>
<th>Identifier2</th>
<th>Identifier3</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvoiceApp</td>
<td>$orderTotal</td>
<td>i</td>
</tr>
<tr>
<td>Invoice</td>
<td>_orderTotal</td>
<td>x</td>
</tr>
<tr>
<td>InvoiceApp2</td>
<td>input_string</td>
<td>TITLE</td>
</tr>
<tr>
<td>subtotal</td>
<td>_get_total</td>
<td>MONTHS_PER_YEAR</td>
</tr>
<tr>
<td>discountPercent</td>
<td>$_64_Valid</td>
<td></td>
</tr>
</tbody>
</table>

The rules for naming an identifier

- Start each identifier with a letter, underscore, or dollar sign. Use letters, dollar signs, underscores, or digits for subsequent characters.
- Use up to 255 characters.
- Don’t use Java keywords.

Exercise: Which of the following are not valid identifiers?
1. Point
2. final
3. #students
4. order_total$
## Keywords

<table>
<thead>
<tr>
<th>boolean</th>
<th>if</th>
<th>interface</th>
<th>class</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>else</td>
<td>package</td>
<td>volatile</td>
<td>false</td>
</tr>
<tr>
<td>byte</td>
<td>final</td>
<td>switch</td>
<td>while</td>
<td>throws</td>
</tr>
<tr>
<td>float</td>
<td>private</td>
<td>case</td>
<td>return</td>
<td>native</td>
</tr>
<tr>
<td>void</td>
<td>protected</td>
<td>break</td>
<td>throw</td>
<td>implements</td>
</tr>
<tr>
<td>short</td>
<td>public</td>
<td>default</td>
<td>try</td>
<td>import</td>
</tr>
<tr>
<td>double</td>
<td>static</td>
<td>for</td>
<td>catch</td>
<td>synchronized</td>
</tr>
<tr>
<td>int</td>
<td>new</td>
<td>continue</td>
<td>finally</td>
<td>const</td>
</tr>
<tr>
<td>long</td>
<td>this</td>
<td>do</td>
<td>transient</td>
<td>goto</td>
</tr>
<tr>
<td>abstract</td>
<td>super</td>
<td>extends</td>
<td>instanceof</td>
<td>null</td>
</tr>
</tbody>
</table>
Classification of keywords

• Values:
  • true, false, null, void

• Primitive Data-Types (8):
  • boolean, char, byte, short, int, float, long, double

• Control Structures:
  • if, else, switch, case, default,
  • for, while, do,
  • break, continue, return, goto

• Exception Handling
  • throw, try, catch, finally

• Abstract Data Types
  • package, import
  • class, interface, extends, implements, throws, instanceof,
  • new, this, super
  • private, protected, public,
  • const, final, static, abstract
  • synchronized, volatile, transient
  • native
The syntax for declaring a class

```java
public|private class ClassName
{
    statements
}
```

The syntax for declaring a main method

```java
public static void main(String[] args)
{
    statements
}
```
A public class that contains a main method

```java
public class InvoiceApp // declare the class
{
    // begin the class
    public static void main(String[] args)
    {
        System.out.println(
            "Welcome to the Invoice Total Calculator");
    }
} // end the class
```

The same class with different brace placement

```java
public class InvoiceApp { // declare and begin the class
    public static void main(String[] args){
        System.out.println(
            "Welcome to the Invoice Total Calculator");
    }
} // end the class
```
Recommended rules for naming a class

- Start the name with a capital letter.
- Use letters and digits only.
- Follow the other rules for naming an identifier.

Recommendations for naming a class

- Start every word within a class name with an initial cap.
- Each class name should be a noun or a noun that’s preceded by one or more adjectives.

Exercise: Which class names follow recommendations?
1. Point_2_Dimensional
2. array
3. Array
4. boolean
5. Boolean
Outline
• Objectives
• A simple Java Program
• **Data-types**
  • Primitive data-types
    • numbers, arithmetic
  • Strings
  • Objects, Classes, Packages
• Control Structures
Two of the eight primitive data types

- `int`
- `double`
How to declare and initialize a variable in one statement

Syntax

type variableName = value;

Examples

int scoreCounter = 1;  // initialize an integer variable
double unitPrice = 14.95;  // initialize a double variable
How to code assignment statements

```java
int quantity = 0;                  // initialize an integer variable
int maxQuantity = 100;             // initialize another integer variable

// two assignment statements
quantity = 10;                     // quantity is now 10
quantity = maxQuantity;            // quantity is now 100
```
**Naming recommendations for variables**

- Start variable names with a lowercase letter and capitalize the first letter in all words after the first word.
- Each variable name should be a noun or a noun preceded by one or more adjectives.
- Try to use meaningful names that are easy to remember.
The basic operators for arithmetic expressions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division (or Quotient)</td>
</tr>
<tr>
<td>%</td>
<td>Remainder</td>
</tr>
</tbody>
</table>
Statements that use simple arithmetic expressions

// integer arithmetic
int x = 14;
int y = 8;
int result1 = x + y;          // result1 = 22
int result2 = x - y;          // result2 = 6
int result3 = x * y;          // result3 = 112
int result4 = x / y;          // result4 = 1

// double arithmetic
double a = 8.5;
double b = 3.4;
double result5 = a + b;       // result5 = 11.9
double result6 = a - b;       // result6 = 5.1
double result7 = a * b;        // result7 = 28.9
double result8 = a / b;       // result8 = 2.5
Statements that increment a counter variable

```java
int invoiceCount = 0;
invoiceCount = invoiceCount + 1;      // invoiceCount = 1
invoiceCount = invoiceCount + 1;      // invoiceCount = 2
```

Statements that add amounts to a total

```java
double invoiceAmount1 = 150.25;
double invoiceAmount2 = 100.75;
double invoiceTotal = 0.0;
invoiceTotal = invoiceTotal + invoiceAmount1;
    // invoiceTotal = 150.25
invoiceTotal = invoiceTotal + invoiceAmount2;
    // invoiceTotal = 251.00
```

Statements that mix int and double variables

```java
int result9 = invoiceTotal / invoiceCount ;

double result10 = invoiceTotal / invoiceCount ;
    // Leading Q? Which statement is legal?
```
Type Conversion or Casting

• Source: (Carrano, Append A, Pages A-9,A-11)

• Implicit:
  • byte $\rightarrow$ short $\rightarrow$ int $\rightarrow$ long $\rightarrow$ float $\rightarrow$ double
  int wholeRate = 7;
  double interestRate = wholeRate;

• Explicit
  int rate = interestRate ; // Illegal
  int rate = (int) interestRate ; // Value truncated, not rounded!

• Exercise: What are values of result1 and result2 after:
  int a = 11; int b = 3; float result1 = a / b ;
  float c = 11.0; float d = 3.0; float result result2 = c /d ;
Math class: Representative Methods

• Source: (Carrano, Append A, Figure A-2, page A-16)

• Methods with double valued argument and result:
  • sqrt(x), cbrt(x), hypot(x, y)
  • cos(x), sin(x), tan(x), toDegrees(x), toRadians(x)
  • ceil(x), floor(x),
  • pow(x, y), exp(x), log(x), log10(x)
  • random(x)

• Methods for int, long, float or double
  • abs(x)
  • max(x, y)
  • min(x, y)

• Method
  • int round(float x)
  • long round(double x)
Outline

• Objectives
• A simple Java Program
• Data-types
  • Primitive data-types
    • numbers, arithmetic
  • Strings
    • Objects, Classes, Packages
• Control Structures
The syntax for declaring and initializing a string variable

```
String variableName = value;
```

Statements that declare and initialize a string

```
String message1 = "Invalid data entry.";
String message2 = "";
String message3 = null;
```

Exercise: Categorize “String” into one of the following:
1. Primitive data-type
2. Object
3. Class
4. Package
How to join strings

```java
String firstName = "Bob";            // firstName is Bob
String lastName = "Smith";           // lastName is Smith
String name = firstName + " " + lastName;
                // name is Bob Smith
```

How to join a string and a number

```java
double price = 14.95;
String priceString = "Price: " + price;
```
How to append one string to another with the + operator

```java
firstName = "Bob";       // firstName is Bob
lastName = "Smith";      // lastName is Smith
name = firstName + " ";  // name is Bob followed by a space
name += lastName;       // name is Bob Smith
```

How to append one string to another with the += operator

```java
firstName = "Bob";       // firstName is Bob
lastName = "Smith";      // lastName is Smith
name = firstName + " ";  // name is Bob followed by a space
name += lastName;       // name is Bob Smith
```
Common escape sequences

- `\n`
- `\t`
- `\r`
- `\"`
- `\\`
A string with a new line

String
"Code: JSPS
Price: $49.50"

Result
Code: JSPS
Price: $49.50

A string with tabs and returns

String
"Joe\tSmith\rKate\tLewis"

Result
Joe      Smith
Kate     Lewis
A string with quotation marks

String
"Type \"x\" to exit"

Result
Type "x" to exit

A string with a backslash

String
"C:\\java\\files"

Result
C:\java\files
More on Strings

• Source: Textbook (Cerrano) Appendix A (pp. A-38 to A-41)

• Methods in String class
  • length(), concat(), trim()
  • charAt( index ), indexOf( subString )
  • toLowerCase(), toUpperCase()
  • compareTo( anotherString), compareToIgnoreCase( anotherString)

• Related Classes
  • StringBuilder
    • methods of append, delete, insert, replace, setCharAt, …
  • Scanner – to extract a piece (e.g., a number) from a String
    • Discussed in later slides

Exercise: Compare and contrast the following pairs:
1. compareTo() method vs. == operator
2. String vs. StringBuilder
3. String vs. Scanner
Outline

- Objectives
- A simple Java Program
- Data-types
  - Primitive data-types
  - String
  - Objects, Classes, Packages
    - Overview
      - (java.lang.)System, System.in, System.out
      - java.util.Scanner, System.in
- Control Structures
How to create an object from a class

Syntax

```
ClassName objectName = new ClassName(arguments);
```

Examples

```
Scanner sc = new Scanner(System.in);
// creates a Scanner object named sc

Date now = new Date();
// creates a Date object named now
```

How to call a method from an object

Syntax:

```
objectName.methodName(arguments)
```

Examples

```
double subtotal = sc.nextDouble();
// get a double entry from the console

String currentDate = now.toString();
// convert the date to a string
```
How to call a static method from a class

Syntax

```
ClassName.methodName(arguments)
```

Examples

```java
String sPrice = Double.toString(price);
// convert a double to a string
double total = Double.parseDouble(userEntry);
// convert a string to a double
```

Note:

- Class-name (e.g., `Double`) prefixed class method-names
- Leading Question: How does “Double” relate to primitive data-type “double”?
- `Double` is a wrapper class
  - Provides additional methods for manipulating “double” valued variables
Leading Question:
  Are wrapper classes available for other primitive data-types (e.g., int)?
  Where are wrapper classes defined?
Common packages

- `java.lang` - classes for String, System, Math, ...
- `java.text` - classes for text, date, ...
- `java.util` - `Scanner`, classes for collections, ...
- `java.io` - classes for file read/write
- `java.sql` - classes for database management systems
- `java.applet`, `java.awt`, `java.awt.event`, `javax.swing`
- ... 2000+ packages with 20,000+ classes ...

Note: `java.lang` is implicitly imported in all Java programs. It provides

- Classes:
  - String, StringBuffer, StringBuilder, ...
  - System with automatically created objects named `in`, `out`, ...
  - Number, `Math`, and wrapper classes, e.g., Int, `Double`, `Boolean`, ...
  - Throwable, StackTraceElement, Thread, Process, ...
- Interfaces: Comparable, ...
- Ref.: http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/package-summary.html
The syntax of the import statement

```java
import packagename.ClassName;
or
import packagename.*;
```

Examples

```java
import java.text.NumberFormat;
import java.util.Scanner;
import java.util.*;
import javax.swing.*;
```
Outline

• Objectives
• A simple Java Program
• Data-types
  • Primitive data-types
  • String
  • Objects, Classes, Packages
    • Overview
    • (java.lang.)System.out
    • java.util.Scanner
• Control Structures
Two methods of the System.out object

- `println(data)` // print data followed by a newline
- `print(data)`

Code that uses the println method

```java
System.out.println("Welcome to the Invoice Total Calculator");
System.out.println("Total: " + total);
System.out.println(message);
System.out.println();  // print a blank line
```

Code that uses the print method

```java
System.out.print("Total: ");
System.out.print(total);
System.out.print("\n");
```

Question: What is `System.out`?
- It represents the standard output, e.g., screen

Question: Where is `System.out` defined?
- Recall `java.lang` is implicitly imported in all java programs
- It has a class named `System`
- With automatically created objects named `in`, `out`, ...
An application that prints data to the console

```java
public class InvoiceApp {
    public static void main(String[] args) {
        double subtotal = 100;
        double discountPercent = .2;
        // compute discount and invoice total
        double discountAmount = subtotal * discountPercent;
        double invoiceTotal = subtotal - discountAmount;
        // print the data to the console
        System.out.println("Welcome to the Invoice Calculator");
        System.out.println();
        System.out.println("Subtotal: 
" + subtotal);
        System.out.println("Discount percent: 
" + discountPercent);
        System.out.println("Discount amount: 
" + discountAmount);
        System.out.println("Total: 
" + invoiceTotal);
    }
}
```

The console output

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welcome to the Invoice Calculator</td>
<td></td>
</tr>
<tr>
<td>Subtotal:</td>
<td>100.0</td>
</tr>
<tr>
<td>Discount percent:</td>
<td>0.2</td>
</tr>
<tr>
<td>Discount amount:</td>
<td>20.0</td>
</tr>
<tr>
<td>Total:</td>
<td>80.0</td>
</tr>
</tbody>
</table>
Outline

• Objectives
• A simple Java Program
• Data-types
  • Primitive data-types
  • Objects, Classes, Packages
    • Overview
    • (java.lang.)System.out
    • java.util.Scanner, (java.lang.)System.in

• Control Structures
How to use the Scanner class to create an object

With an import statement
Scanner sc = new Scanner(System.in);

Without an import statement
java.util.Scanner sc = new java.util.Scanner(System.in);

Question: Name packages and classes in the import statement.
Question: What is System.in?
  • It represents the standard input, e.g., keyboard
  • It allows Scanner object sc to read from standard input

Question: Categorize System and System.in
  (a) System is a package, System.in is a class
  (b) System is a class and System.in is an object

Question: Why didn’t we use java.lang.System.in? Where is System.in defined?
  • Recall java.lang is implicitly imported in all java programs
  • It has a class named System
  • With automatically created objects named in, out, …
The documentation for the Scanner class

public final class Scanner
    extends Object
    implements Iterator<String>, Closeable

A simple text scanner which can parse primitive types and strings using regular expressions.

A scanner breaks its input into tokens using a delimeter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

For example, this code allows a user to read a number from System.in:

    Scanner sc = new Scanner(System.in);
    int i = sc.nextInt();

As another example, this code allows long types to be assigned from entries in a file myNumbers:

    Scanner sc = new Scanner(new File("myNumbers"));
The Scanner class
java.util.Scanner

How to create a Scanner object
Scanner sc = new Scanner(System.in);

Common methods of a Scanner object
- next()
- nextInt()
- nextDouble()
- nextLine()

How to use the methods of a Scanner object
String name = sc.next();
int count = sc.nextInt();
double subtotal = sc.nextDouble();
String cityName = sc.nextLine();

Note
- The Scanner class was introduced in version 1.5 of the JDK.
Code that gets three values from the user

```java
Scanner sc = new Scanner(System.in); // create a Scanner object

// read a string, a double value, and an int
System.out.print("Enter product code: ");
String productCode = sc.next();
System.out.print("Enter price: ");
double price = sc.nextDouble();
System.out.print("Enter quantity: ");
int quantity = sc.nextInt();

// perform a calculation and display the result
double total = price * quantity;
System.out.println();
System.out.println(quantity + " " + productCode + " @ " + price + " = " + total);
System.out.println();
```

The console after the program finishes

```
Enter product code: cshp
Enter price: 49.50
Enter quantity: 2

2 cshp @ 49.5 = 99.0
```
Code that reads three values from one line

```java
// read three int values
System.out.print("Enter three integer values: ");
int i1 = sc.nextInt();
int i2 = sc.nextInt();
int i3 = sc.nextInt();

// calculate the average and display the result
int total = i1 + i2 + i3;
int avg = total / 3;
System.out.println("Average: "+ avg);
System.out.println();
```

The console after the program finishes

```
Enter three integer values: 99 88 92
Average: 93
```
More on Scanner

• Source: Textbook (Cerrano) Appendix A (pp. A-42 to A-44)

• Other Methods in Scanner class
  • `useDelimiter(aString)` - change default of whitespace for
  • `next()`, `nextInt()`, …

• Example in A.82 (pp. A-42)
  • `useDelimiter(`"`,")`  

• Figure A-6 (pp. A-43) shows common delimiters
  • digits (\d), whitespace (\s), letter_digit_underscore (\w), any character (_)
  • Regular expression: ?, *, +, {n}, {n,}, …

Exercise: Specify delimiters to read next element on current line for
1. A Spreadsheet (.csv format)
2. A Unix password file
3. Non-comment parts of a Java program
Outline

• Objectives
• A simple Java Program
• Data-types
• **Control Structures**
  • Tests
  • Conditionals
  • Loops
Examples of conditional expressions

discountPercent == 2.3  // equal to a numeric literal
subtotal != 0           // not equal to a numeric literal
years > 0               // greater than a numeric literal
i < months              // less than a numeric variable
subtotal >= 500         // greater than or equal to a numeric literal
quantity <= reorderPoint // less than or equal to a numeric variable
Relational operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Equality</td>
</tr>
<tr>
<td>!=</td>
<td>Inequality</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater Than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less Than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater Than Or Equal</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less Than Or Equal</td>
</tr>
</tbody>
</table>

Comparison with Mathematics

• Source: Textbook (Cerrano) Appendix A, Fig. A-3 (pp. A-21)

• **Leading Question?** Which data-types do relational operators compare?
  • primitive data types, e.g., int, boolean, float
  • objects, e.g. String, Date, …
Two methods of the String class

- `equals(String)`
- `equalsIgnoreCase(String)`

Examples

- `userEntry.equals("Y")`  // equal to a string literal
- `userEntry.equalsIgnoreCase("Y")`  // equal to a string literal
- `(!lastName.equals("Jones"))`  // not equal to a string literal
- `code.equalsIgnoreCase(productCode)`  // equal to another string variable

Leading Question? Consider following Strings in Java:

```
String s1 = “abcd”;
String s2 = “abcd”;
String s3 = s1;
```

Which of the following are true?

(a) s1.equals(s2)  
(b) s1 == s2  
(c) s1.equals(s3)  
(d) s1 == s2  
(e) s2.equals(s3)  
(f) s2 == s3
Outline
• Objectives
• A simple Java Program
• Data-types
• Control Structures
  • Tests
  • Conditionals
  • Loops
The syntax of the if/else statement

```
if (booleanExpression) {statements}
[else if (booleanExpression) {statements}] ...
[else {statements}]
```

If statements without else if or else clauses

With a single statement
```
if (subtotal >= 100)
    discountPercent = .2;
```

With a block of statements
```
if (subtotal >= 100)
{
    discountPercent = .2;
    status = "Bulk rate";
}
```
An if statement with an else clause

if (subtotal >= 100)
    discountPercent = .2;
else
    discountPercent = .1;

An if statement with else if and else clauses

if (customerType.equals("T"))
    discountPercent = .4;
else if (customerType.equals("C"))
    discountPercent = .2;
else if (subtotal >= 100)
    discountPercent = .2;
else
    discountPercent = .1;
Outline

• Objectives
• A simple Java Program
• Data-types
• **Control Structures**
  • Tests
  • Conditionals
  • **Loops**
The syntax of the while loop

```java
while (booleanExpression) {
    statements
}
```

A loop that calculates the sum of the numbers 1 through 4

```java
int i = 1;
int sum = 0;
while (i < 5) {
    sum = sum + i;
    i = i + 1;
}
```
A loop that continues while choice is “y” or “Y”

String choice = "y";
while (choice.equalsIgnoreCase("y"))
{
    // get the invoice subtotal from the user
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter subtotal: ");
    double subtotal = sc.nextDouble();

    // the code that processes the user's entry goes here

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

- Objectives
- A simple Java Program
- Data-types
- Control Structures
  - Tests
  - Conditionals
  - Loops
- Refined Java Program
The code for the Invoice application

```java
import java.util.Scanner; // Q? Why didn’t we import System?

public class InvoiceApp {
    public static void main(String[] args) {
        System.out.println(
            "Welcome to the Invoice Calculator");
        System.out.println(); // print a blank line

        Scanner sc = new Scanner(System.in);

        // perform invoice calculations until choice isn't
        // equal to "y" or "Y"
        String choice = "y";
        while (choice.equalsIgnoreCase("y")) {
            // get the invoice subtotal from the user
            System.out.print("Enter subtotal: ");
            double subtotal = sc.nextDouble();
        }
    }
}
```
The code for the application (cont.)

```java
// calculate the discount amount and total
double discountPercent = 0.0;
if (subtotal >= 200)   discountPercent = .2;
else if (subtotal >= 100) discountPercent = .1;
else                  discountPercent = 0.0;
double discountAmount = subtotal * discountPercent;
double total = subtotal - discountAmount;

// display the discount amount and total
String message = "Discount percent: " + discountPercent + "\n" + "Discount amount: " + discountAmount + "\n" + "Invoice total: " + total + "\n";
System.out.println(message);

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

The console input and output for a test run of the Invoice application

Welcome to the Invoice Calculator

Enter subtotal: 150
Discount percent: 0.1
Discount amount: 15.0
Invoice total: 135.0

Continue? (y/n):
Review Quiz
1. Which of the following can you assign to a String variable?
   (a.) an empty string  
   (b.) null  
   (c.) a string literal  
   (d.) all of these
2. To refer to a Java class without qualification, you need to import the class unless that class
   (a.) contains only static methods  
   (b.) is stored in a package  
   (c.) is stored in a package that has already been imported
3. You can use a Scanner object to
   (a.) get the next token in an input line  
   (b.) display a prompt on the console ,  
   (c.) display a blank line on a console  
   (d.) check your Java code for exceptions
4. You can use relational operators (e.g., ==) to
   (a.) compare String variables  
   (b.) compare numeric variable  
   (c.) both a and b  
   (d.) neither a nor b
5. How many times will the while loop that follows be executed if months has a value of 5?
   ```java
   int i = 1; double v = 100; while (i < months) {
      v = v * (1 + i);
      i = i+1;
   }
   ```
   (a.) 0  
   (b.) 4  
   (c.) 5  
   (d.) 6
6. Which two keywords should always be used on the declaration of the main method?
   (a.) private and static  
   (b.) public and static  
   (c.) private and int  
   (d.) public and static
7. Which package is automatically available to all Java programs?
   (a.) java.lang  
   (b.) java.util  
   (c.) java.text  
   (d.) java.basic
8. What happens when you use both integer and double values in an arithmetic expression?
   (a.) an exception occurs  
   (b.) the integer values are cast to double values  
   (c.) the double values are cast to integer values  
   (d.) all values are cast to the type of the result variable