Nested selection statements:

Similarly, you can nest two (or more) loops. Trace the code in listing 5.7 in Chapter 5.9 to see how it works.

The example below shows how nested loops execute. Note that the inner loop completes all its iterations for each iteration of the outer loop.

```
for (int i=1; i<=5; i++) {                        // start of outer loop <------+
   System.out.println("\n i is now " + i);
   for(int j=1; j<=4; j++) {                      // start of inner loop
      System.out.print("  j = " + j);
   }                                             // end of inner loop
}                                                // end of outer loop <--------+
```

We will trace this code in class, but if you're trying this on your own, you can match your manual tracing's output with the actual output below:

```
i is now 1
  j = 1  j = 2  j = 3  j = 4
i is now 2
  j = 1  j = 2  j = 3  j = 4
i is now 3
  j = 1  j = 2  j = 3  j = 4
i is now 4
  j = 1  j = 2  j = 3  j = 4
i is now 5
  j = 1  j = 2  j = 3  j = 4
```

Nested loops are often used to handle or display tabular information. The trick to making this work is to remember that usually **the outer loop prints a row, and the inner loop prints each column in a row**.

Exercises

1. What's the output of each of the following programs:

Example a:

```

public class ExerciseA {

    public static void main(String[] args) {

        for (int i=0; i<=5; i++) {
            System.out.print(i + ":  ");
            for (int j=1; j<3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}

```

Solution

```
0:  0 0
1:  1 2
2:  2 4
3:  3 6
4:  4 8
5:  5 10
```

Example b:

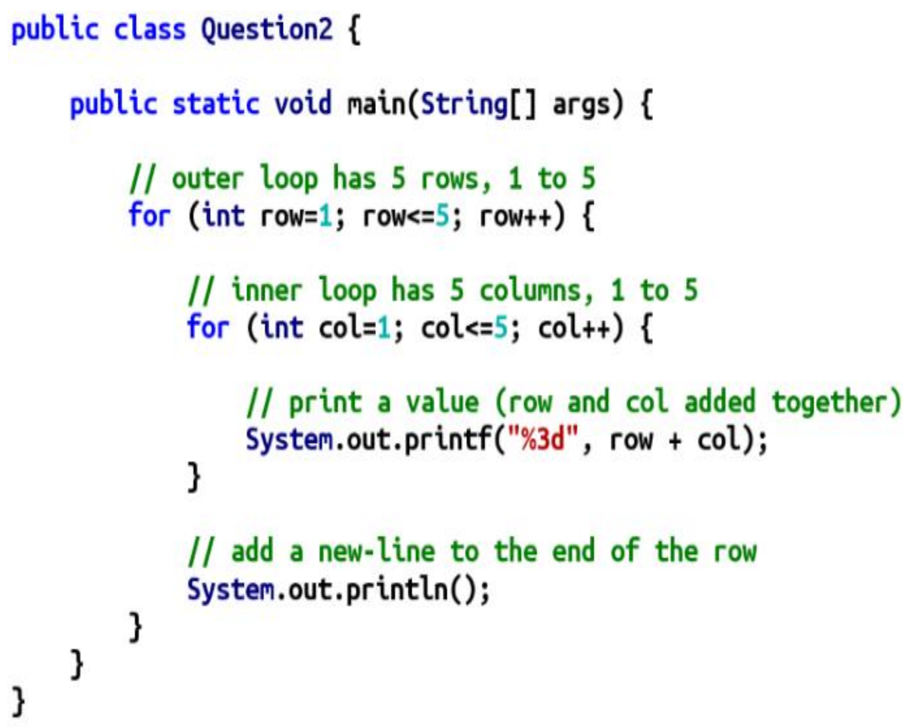
```
public class ExerciseA {

    public static void main(String[] args) {

        for (int i=0; i<=5; i++) {
            System.out.print(i + ":  ");
            for (int j=3; j>0; j--) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

Solution

```
0:  0 0 0
1:  3 2 1
2:  6 4 2
3:  9 6 3
4:  12 8 4
5:  15 10 5
```

2. Use nested loops to print the following table:

```
2   3   4   5   6
3   4   5   6   7
4   5   6   7   8
5   6   7   8   9
6   7   8   9  10
```



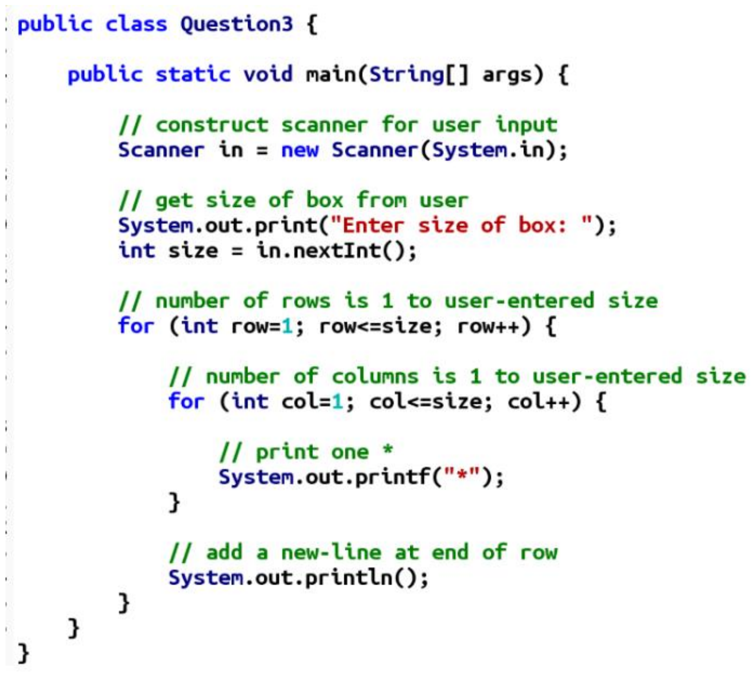
```java
public class Question2 {

    public static void main(String[] args) {

        // outer loop has 5 rows, 1 to 5
        for (int row=1; row<=5; row++) {

            // inner loop has 5 columns, 1 to 5
            for (int col=1; col<=5; col++) {

                // print a value (row and col added together)
                System.out.printf("%3d", row + col);
            }

            // add a new-line to the end of the row
            System.out.println();
        }
    }
}
```

3. Use nested loops to print a square of asterisks (*) with a given number of rows and columns. E.g. if the user enters 3, it would print:

```
***
***
***
```

If the user enters 5, it would print:

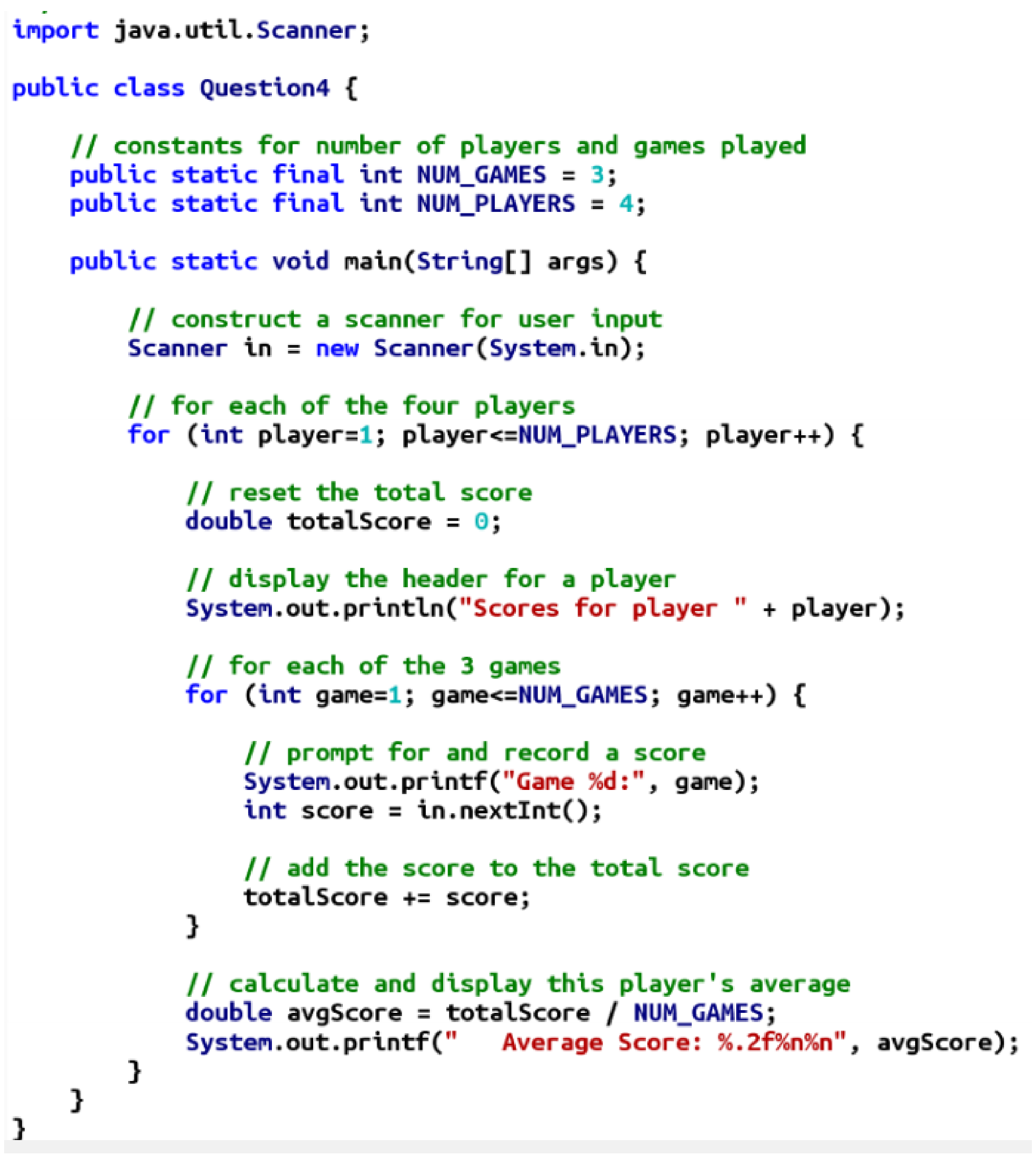
```
*****
*****
*****
*****
*****
```



```java
public class Question3 {

    public static void main(String[] args) {

        // construct scanner for user input
        Scanner in = new Scanner(System.in);

        // get size of box from user
        System.out.print("Enter size of box: ");
        int size = in.nextInt();

        // number of rows is 1 to user-entered size
        for (int row=1; row<=size; row++) {

            // number of columns is 1 to user-entered size
            for (int col=1; col<=size; col++) {

                // print one *
                System.out.printf("*");
            }

            // add a new-line at end of row
            System.out.println();
        }
    }
}
```

4. You and three of your classmates go bowling and want to track and display your results for three games. Write a program that prompts the user to enter the score for each player for each of the three games, and then displays the average score for each player. Sample output is shown below (prompts in navy blue, inputs in red, output in black):

```
Scores for Player 1
Game 1: 200
Game 2: 205
Game 3: 210
  Average Score: 205.00

Scores for Player 2
Game 1: 198
Game 2: 182
Game 3: 201
  Average Score: 193.67

Scores for Player 3
Game 1: 252
Game 2: 267
Game 3: 281
  Average Score: 266.67

Scores for Player 4
Game 1: 207
Game 2: 243
Game 3: 199
  Average Score: 216.33
```



```java
import java.util.Scanner;

public class Question4 {

    // constants for number of players and games played
    public static final int NUM_GAMES = 3;
    public static final int NUM_PLAYERS = 4;

    public static void main(String[] args) {

        // construct a scanner for user input
        Scanner in = new Scanner(System.in);

        // for each of the four players
        for (int player=1; player<=NUM_PLAYERS; player++) {

            // reset the total score
            double totalScore = 0;

            // display the header for a player
            System.out.println("Scores for player " + player);

            // for each of the 3 games
            for (int game=1; game<=NUM_GAMES; game++) {

                // prompt for and record a score
                System.out.printf("Game %d:", game);
                int score = in.nextInt();

                // add the score to the total score
                totalScore += score;
            }

            // calculate and display this player's average
            double avgScore = totalScore / NUM_GAMES;
            System.out.printf("   Average Score: %.2f%n%n", avgScore);
        }
    }
}
```

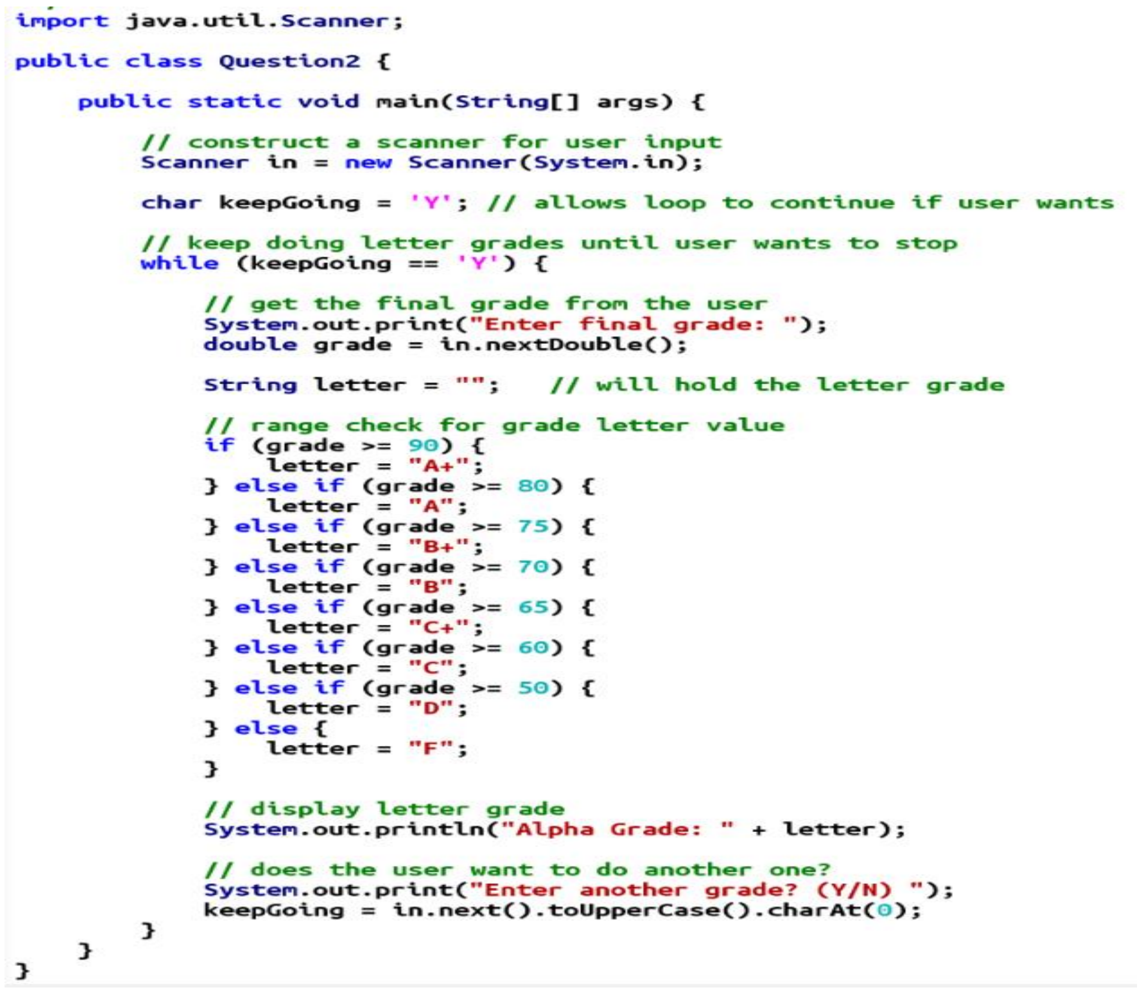
Nesting Other Structures

You can also nest selections inside loops, and loops inside selections. For example, determine the output of each of the following code segments:

```
// first code segment
for (int i=1; i<=10; i++)
{
    if (i%2 == 0)
        System.out.println(i);
}


// second code segment

System.out.println("Enter the high number:");
int high = Integer.parseInt(keysIn.readLine());
System.out.println("Enter the low number:");
int low = Integer.parseInt(keysIn.readLine());


if (high > low)
{
    for (int i=low; i<=high; i++)
        System.out.println(i);
} else
    System.out.println("Your low value must be smaller " +
        "than your high value!");
```

Exercises

A user wants to find out the grade points for each of their final grades this term. Write a program that allows the user to enter any number of final grades that they want, and display the grade points according to Sheridan's Grading System.



```java
import java.util.Scanner;

public class Question2 {

    public static void main(String[] args) {

        // construct a scanner for user input
        Scanner in = new Scanner(System.in);

        char keepGoing = 'Y'; // allows loop to continue if user wants

        // keep doing letter grades until user wants to stop
        while (keepGoing == 'Y') {

            // get the final grade from the user
            System.out.print("Enter final grade: ");
            double grade = in.nextDouble();

            String letter = "";   // will hold the letter grade

            // range check for grade letter value
            if (grade >= 90) {
                letter = "A+";
            } else if (grade >= 80) {
                letter = "A";
            } else if (grade >= 75) {
                letter = "B+";
            } else if (grade >= 70) {
                letter = "B";
            } else if (grade >= 65) {
                letter = "C+";
            } else if (grade >= 60) {
                letter = "C";
            } else if (grade >= 50) {
                letter = "D";
            } else {
                letter = "F";
            }

            // display letter grade
            System.out.println("Alpha Grade: " + letter);

            // does the user want to do another one?
            System.out.print("Enter another grade? (Y/N) ");
            keepGoing = in.next().toUpperCase().charAt(0);
        }
    }
}
```

Additional Exercises

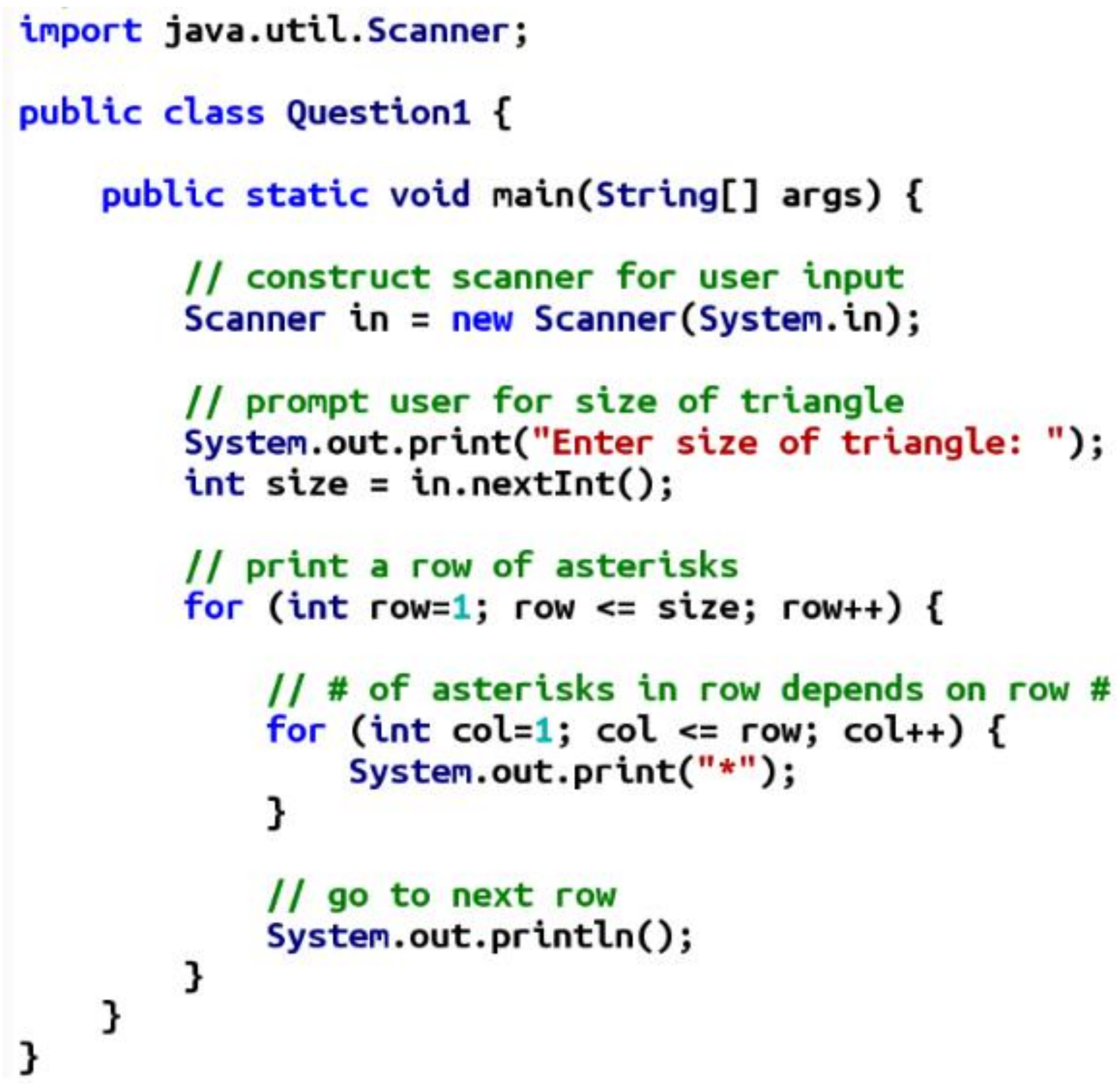
[solutions]

1. Use nested loops to print a triangle with a given number of rows. E.g. if the user enters 3, it would print:

```
*
**
***
```

If the user enters 5, it would print:

```
*
**
***
****
*****
```



```java
import java.util.Scanner;

public class Question1 {

    public static void main(String[] args) {

        // construct scanner for user input
        Scanner in = new Scanner(System.in);

        // prompt user for size of triangle
        System.out.print("Enter size of triangle: ");
        int size = in.nextInt();

        // print a row of asterisks
        for (int row=1; row <= size; row++) {

            // # of asterisks in row depends on row #
            for (int col=1; col <= row; col++) {
                System.out.print("*");
            }

            // go to next row
            System.out.println();
        }
    }
}
```

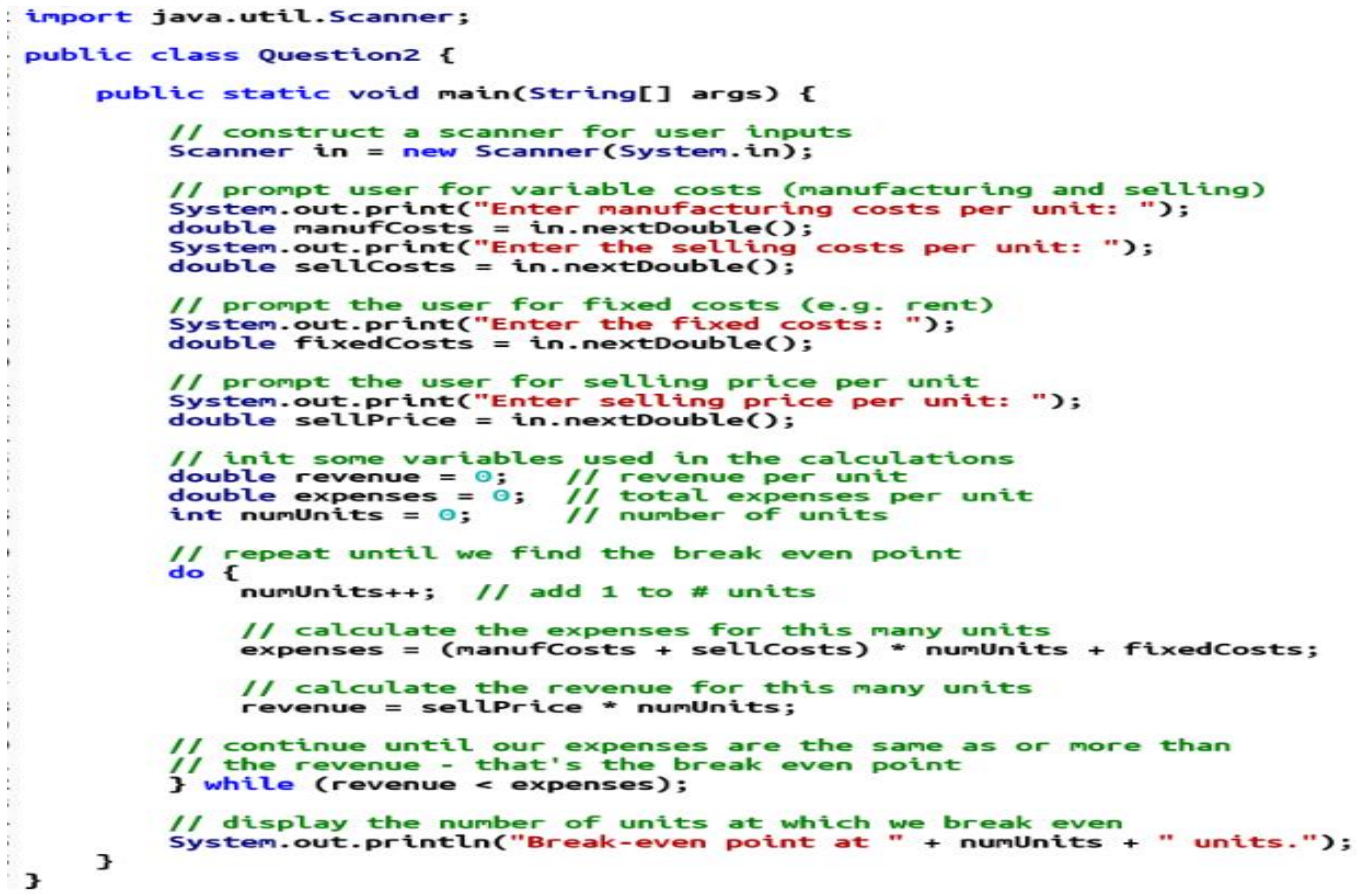
2. Break-even Analysis: When you are manufacturing a product, you need to know how many units you should produce in order to break even. For example, say you are making widgets. You pay $1000 a week to rent your manufacturing building, and it costs you $50 to manufacture one widget (raw materials and labour). It also costs an average of $10 per widget to sell and market your widgets. The rent is what we call a "fixed" cost, because it's always the same from week to week (or whatever period of time for which you are analyzing). Whether you make 1 widget or 1000 widgets, you are still going to pay $1000 a week in rent. Costs such as the manufacturing and selling costs are "variable" costs, because they vary, depending on how many widgets you make. If you make one widget, it will cost you $50 + $10 = $60 in variable costs, but if you make 1000 widgets, it will cost you (50 + 10) * 1000 = $60,000 in variable costs.

Which is better, to make 1 widget or to make 1000 widgets? It looks like it will cost a lot ot make 1000 widgets! Once you have a selling price for your product, you can calculate how much revenue you will gain (or lose!). For example, if we sell one widget for $200, will we make a profit selling only one widget or selling 1000 widgets?

Since it costs $60 + $1000 = $1060 to make one widget, and we will make $200 selling that widget, we'll end up with a loss of $1860! Not very good! Will we make a profit selling 1000 widgets? It costs $60,000 + $1000 = $61,000 to make 1000 widgets, and we will make $200 * 1000 = $200,000 in revenue from 1000 widgets. That means we've made a profit of $140,000! Not bad!

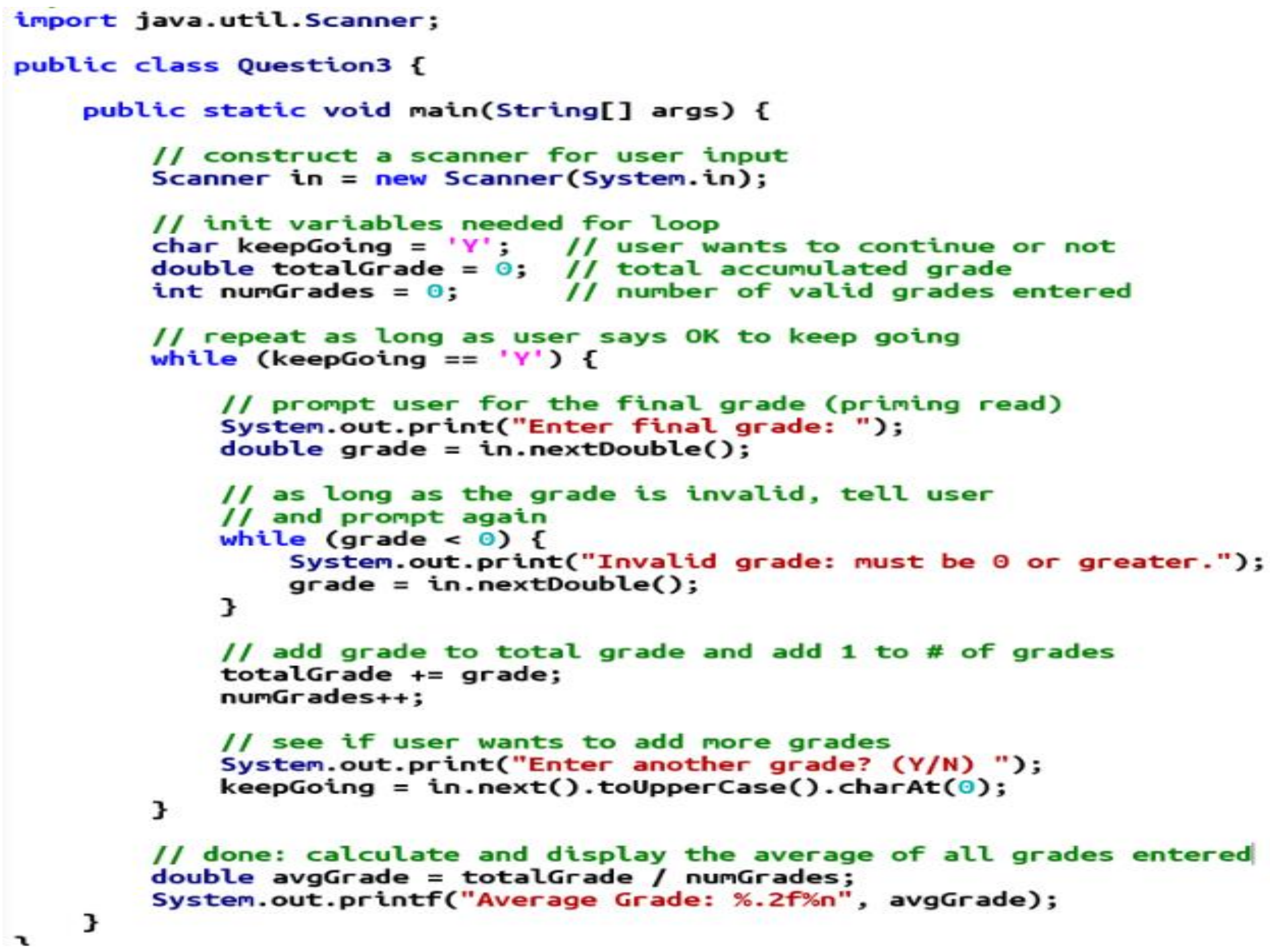
Do we make money or lose money if we manufacture 10 widgets? It will cost (50 + 10) * 10 + 1000 = $1600 to make 10 widgets and we will make $200 * 10 = $2000 in revenue. This means a profit of $400.

An important part of this process is knowing your break-even point. This is the point at which you stop making a loss and start making a profit. Write a program that prompts the user to enter the manufacturing costs, selling costs, fixed costs, and selling price. Your program should then calculate how many units it will take to reach the break-even point.



```java
import java.util.Scanner;

public class Question2 {

    public static void main(String[] args) {

        // construct a scanner for user inputs
        Scanner in = new Scanner(System.in);

        // prompt user for variable costs (manufacturing and selling)
        System.out.print("Enter manufacturing costs per unit: ");
        double manufCosts = in.nextDouble();
        System.out.print("Enter the selling costs per unit: ");
        double sellCosts = in.nextDouble();

        // prompt the user for fixed costs (e.g. rent)
        System.out.print("Enter the fixed costs: ");
        double fixedCosts = in.nextDouble();

        // prompt the user for selling price per unit
        System.out.print("Enter selling price per unit: ");
        double sellPrice = in.nextDouble();

        // init some variables used in the calculations
        double revenue = 0;    // revenue per unit
        double expenses = 0;   // total expenses per unit
        int numUnits = 0;      // number of units

        // repeat until we find the break even point
        do {
            numUnits++;  // add 1 to # units

            // calculate the expenses for this many units
            expenses = (manufCosts + sellCosts) * numUnits + fixedCosts;

            // calculate the revenue for this many units
            revenue = sellPrice * numUnits;

        // continue until our expenses are the same as or more than
        // the revenue - that's the break even point
        } while (revenue < expenses);

        // display the number of units at which we break even
        System.out.println("Break-even point at " + numUnits + " units.");
    }
}
```

3. Write a program that calculates the average final grade for a student who's taken some unknown number of courses. The final grade entered must be greater than or equal 0: if the user enters a negative grade, display an error message and ask again.



```java
import java.util.Scanner;

public class Question3 {

    public static void main(String[] args) {

        // construct a scanner for user input
        Scanner in = new Scanner(System.in);

        // init variables needed for loop
        char keepGoing = 'Y';    // user wants to continue or not
        double totalGrade = 0;   // total accumulated grade
        int numGrades = 0;       // number of valid grades entered

        // repeat as long as user says OK to keep going
        while (keepGoing == 'Y') {

            // prompt user for the final grade (priming read)
            System.out.print("Enter final grade: ");
            double grade = in.nextDouble();

            // as long as the grade is invalid, tell user
            // and prompt again
            while (grade < 0) {
                System.out.print("Invalid grade: must be 0 or greater.");
                grade = in.nextDouble();
            }

            // add grade to total grade and add 1 to # of grades
            totalGrade += grade;
            numGrades++;

            // see if user wants to add more grades
            System.out.print("Enter another grade? (Y/N) ");
            keepGoing = in.next().toUpperCase().charAt(0);
        }

        // done: calculate and display the average of all grades entered
        double avgGrade = totalGrade / numGrades;
        System.out.printf("Average Grade: %.2f%n", avgGrade);
    }
}
```