Chap07Yourname.java

Loops

```
import java.util.Scanner;

public class Chap07 {
	private static Scanner in;

	public static void main(String[] args) {
		in = new Scanner(System.in);




	}
}
```

7.1 The while statement

```
System.out.println("7.1 The while statement");
//A while loop executes statements repeatedly while the condition is true.
countDown(5);
System.out.println();
```

7.1 The while statement

```
public static void countDown(int n) {
        if (n == 0) {
                System.out.println("GO!");
        } else {
                System.out.println(n);
                countDown(n - 1);
        }
}
```

7.1 The while statement

```
//      public static void countDown(int n) {
//              if (n == 0) {
//                      System.out.println("GO!");
//              } else {
//                      System.out.println(n);
//                      countDown(n - 1);
//              }
//      }
//      //Recursive version of countDown(int n)
```

7.1 The while statement

```
public static void countDown(int n) {
        while (n > 0) {
                System.out.println(n);
                n = n - 1;
        }
        System.out.println("GO!");
}
```

7.3 Generating the Multiplication Table

```
System.out.println("7.3 Generating the Multiplication Table");
int i = 1;
while (i <= 9) {
        System.out.printf("%4d", 1 * i);
        i = i + 1;
}
//The preceding code gave row 1 of the table. Then we encapsulate
//the code in a new method: printRow(), and generalize it to row n.
System.out.println();
printRow(2);
System.out.println();
```

7.3 Generating the Multiplication Table

```
public static void printRow(int n) {
        int i = 1;//initializer
        while (i <= 9) {//condition
                System.out.printf("%4d", ____);
                i = i + 1;//update
        }
}
//The format specifier %4d causes the output to align vertically, regardless
//of whether the numbers are one or two digits.
```

7.3 Generating the Multiplication Table

```
public static void printRow(int n) {
        int i = 1;//initializer
        while (i <= 9) {//condition
                System.out.printf("%4d", n * i);
                i = i + 1;//update
        }
}
//The format specifier %4d causes the output to align vertically, regardless
//of whether the numbers are one or two digits.
```

7.3 Generating the Multiplication Table

```
//Now we have row 2. Write a method multiTable to generate a 9 * 9 table.

System.out.println();
multiTable();
System.out.println();
```

7.3 Generating the Multiplication Table

```
public static void multiTable() {



}
```

7.3 Generating the Multiplication Table

```
public static void multiTable() {
        int i = 1;
        while (i <= 9) {
                printRow(i);
                System.out.println();
                i = i + 1;
        }
}
```

7.3 Generating the Multiplication Table

//Write a new method printRow(int r, int c) generating the rth row of a
//multiplication table with c columns.

//Having more than one method with the same name is called overloading.
//It is legal in Java as long as each method takes different parameters.
//method (int n) and method (int m) are not different.
//method (int n) and method (double x) are different.

7.3 Generating the Multiplication Table

```
public static void printRow(int r, int c) {



}
```

7.3 Generating the Multiplication Table

```
public static void printRow(int r, int c) {
        int i = 1;
        while (i <= c) {
                System.out.printf("%4d", r * i);
                i = i + 1;
        }
}
```

7.3 Generating the Multiplication Table

```
//Write an method multiTable(n) generating an n * n multiplication table,
//taking an int n <= 30 as parameter. You can invoke printRow(r, c).

System.out.println();
System.out.print("Let me generate the n * n multiplication table."
                        + " n = ");
int n = in.nextInt();
multiTable(n);
System.out.println();
```

7.3 Generating the Multiplication Table

```
public static void multiTable(int n) {

}
```

7.3 Generating the Multiplication Table

```
public static void multiTable(int n) {
        int i = 1;
        if (n > 30) {
                System.out.println("ERROR: n should not be larger than 30!");
                return;
        }
        while (i <= n) {
                printRow(i, n);
                System.out.println();
                i = i + 1;
        }
}
```

7.3 Generating the Multiplication Table

```
//What's wrong is it to invoke the following multiTable(1) to get a 9 * 9 multi-table?
public static void multiTable(int i) {
        int r = 1;
        while (r <= 9) {
                while(i <= 9) {
                        System.out.printf("%4d", r * i);
                        i = i + 1;
                }
                System.out.println();
                r = r + 1;
        }
}
```

7.3 Generating the Multiplication Table

```
public static void multiTable() {
        int r = 1;
        while (r <= 9) {
                int i = 1;
                while(i <= 9) {
                        System.out.printf("%4d", r * i);
                        i = i + 1;
                }
                System.out.println();
                r = r + 1;
        }
}
```

7.5 The for statement

```
System.out.println("7.5 The for statement");
//A for loop has a concise syntax for writing loops.
//Rewrite the printRow(int n), multiTable(), printRow(int r, int c) and
//multiTable(int n).
System.out.println();
```

7.5 The for statement

```
//      public static void printRow(int n) {
//              int i = 1;//initializer
//              while (i <= 9) {//condition
//                      System.out.printf("%4d", n * i);
//                      i = i + 1;//update
//              }
//      }
//      //The format specifier %4d causes the output to align vertically, regardless
//      //of whether the numbers are one or two digits.
```

7.5 The for statement

```
public static void printRow(int n) {
        for (int i = 1; i <= 9; i++) {
                System.out.printf("%4d", n * i);
        }
}
//i++ means i = i + 1;
//i-- means i = i - 1;
//i += n means i = i + n;
//i -= n means i = i - n;
```

7.5 The for statement

```
//      public static void multiTable() {
//              int i = 1;
//              while (i <= 9) {
//                      printRow(i);
//                      System.out.println();
//                      i = i + 1;
//              }
//      }
```

7.5 The for statement

```
public static void multiTable() {
        for (int i = 1; i <= 9; i++) {
                printRow(i);
                System.out.println();
        }
}
```

7.5 The for statement

```
//      public static void printRow(int r, int c) {
//              int i = 1;
//              while (i <= c) {
//                      System.out.printf("%4d", r * i);
//                      i = i + 1;
//              }
//      }
//      //Having more than one method with the same name is called overloading.
//      //It is legal in Java as long as each method takes different parameters.
//      //(int n) and (int m) are not different, (int n) and (double x) are.
```

7.5 The for statement

```
public static void printRow(int r, int c) {
        for (int i = 1; i <= c; i++) {
                System.out.printf("%4d", r * i);
        }
}
```

7.5 The for statement

```
//      public static void multiTable(int n) {
//              int i = 1;
//              if (n > 30) {
//                      System.out.println("ERROR: n should not be larger than 30!");
//                      return;
//              }
//              while (i <= n) {
//                      printRow(i, n);
//                      System.out.println();
//                      i = i + 1;
//              }
//      }
```

7.5 The for statement

```
public static void multiTable(int n) {
        if (n > 30) {
                System.out.println("ERROR: n should not be larger than 30!");
                return;
        }
        for (int i = 1; i <= n; i++) {
                printRow(i, n);
                System.out.println();
        }
}
```

7.6 The do-while statement

```
System.out.println("7.6 The do-whild loop");
//The while and for statements are pretest loops.  They test the condition
//at the beginning of each pass through the loop.

//A do-while loop is the same as a while loop except that it execute the
//loop body first and then checks the loop continuation condition.
```

7.6 The do-while statement

```
do {
        System.out.println("Enter a number: ");
        if (!in.hasNextDouble()) {
                System.out.println(in.next() + " is not a number.");
        }
} while (!in.hasNextDouble());
double x = in.nextDouble();
System.out.printf("The number is %f.\n", x);
System.out.println();
```

7.7 Break and continue

```
System.out.println("7.7 Break and continue");
//When a program reaches a break statement, it exits the current loop.
//When        ...        continue ...   , it skips the current loop and
//program goes to the end of the loop body.

//Write a program which adds integers i from 1 until the sum is greater
//than or equal to 100 using the while (i >= 1) loop and break statement.
```

7.7 Break and continue

```
int sum = 0;
i = 1;

while (i >= 1) {
        sum += i;
        i++;
        if (sum >= 100)
                break;
}
System.out.println("The number is " + (i -1));
System.out.println("The sum is " + sum);
System.out.println();
```

7.7 Break and continue

//Write a program which adds integers i from 1 to 20 except 10 and 11

//using the while (i < 20) loop and continue statement.

7.7 Break and continue

```
sum = 0;
i = 0;//the assignments are shown

while (i < 20) {
        i++;
        if (i == 10 || i == 11) {
                continue;
        }
        sum += i;
}
System.out.println("The sum is " + sum);
System.out.println();
```