The Switch Statements

10

10.1 The Syntax of Switch Statements

The switch statement is a mechanism for controlling the flow of the program based on exact values of one data. Often a switch statement is used in a menu, where the action to be performed is selected based on a value.

A switch statement has three components, the header, the body, and the anchors. The header takes the form of:

```
switch ( x )
```

where `x` is a data and its type must be `int`, `char`, `short`, `long`, or `String`. The body is a series of statements encompassed in a pair of curly brackets. An anchor is either in the form of `default:` or `case A:` for some literal `A` whose type is the same as the type of `x`. Anchors appear between the statements in the body. Multiple `case`-type anchors may appear in the body, but no two of them can have the same associated literals. The anchor `default:` may or may not appear, but it cannot appear more than once. Furthermore, `break` can be used as a valid statement.

For instance, with an `int` variable `x`, the following switch statements can be written:

```
1 // Example No.1
2 switch ( x )
3 {
4   case 11: System.out.print( "Eleven" );
5   default: System.out.print( "Other" );
6   case 8: System.out.print( "Eight" );
7 }
```

```
1 // Example No.2
2 switch ( x )
3 {
4   case 11: System.out.print( "Eleven" ); break;
5   default: System.out.print( "Other" ); break;
6   case 8: System.out.print( "Eight" );
7 }
```


M. Ogihara, *Fundamentals of Java Programming*,
https://doi.org/10.1007/978-3-319-89491-1_10

Table 10.1 The output generated by the three examples of `switch`

Example no.	The value of x		
	11	8	Others
1	ElevenOtherEight	Eight	OtherEight
2	Eleven	Eight	Other
3	Eleven	Eight	

```
1 // Example No.3
2 switch ( x )
3 {
4   case 11: System.out.print( "Eleven" ); break;
5   case 8: System.out.print( "Eight" );
6 }
```

The anchors specify the entry points into the body. `case A:` means

"if the value of `x` is equal to `A`, start the execution of the body from here"

and `default:` means

"if none of the associated values of the `case` anchors match the value of `x`, start the execution of the body from here".

The execution of the body is terminated either when the execution reaches the end of the body or when the execution encounters `break`. For this reason, the `break` appearing at the end of the body is redundant and thus can be removed.

The above three examples produce the following results: In the first example, `break` does not appear in the body. Therefore, after entering the body, the execution continues until the very end. In the second example, each `System.out.print` statement is followed by `break`. Therefore, after entering the body, the program executes the `System.out.print` following the anchor, and then terminates the execution of the body. In the last example, if the value of `x` is either 11 or 8, the action to be performed is the same as that of the second example. However, if the value of `x` is neither, there is no action to perform, so nothing occurs.

Table 10.1 summaries this analysis.

To see how we can utilize a switch statement for a menu, consider printing various shapes on the screen upon the user's request. This program asks the user to choose, from a menu of possible shapes, one shape to print:

```
1 ****************************************
2 This program prints a shape of your choice
3 Select by entering number
4 0. Right-angle triangle
5 1. Isosceles
6 2. Square
7 3. Parallelogram
8 Enter your choice: 3
9 Enter height: 10
```

```
Here is your figure!
##########
 ##########
  ##########
   ##########
    ##########
     ##########
      ##########
       ##########
        ##########
         ##########

```

In Lines 8 and 9, the program prompts the user to choose the shape and its size, and receives `3` and `10` from the user. The output of the shape appears after receiving the size parameter.

As shown in the example, there are four possible shapes to choose in the menu. The user can choose the size of each shape. The generation of the four shapes are accomplished by four methods, `rightAngle`, `isosceles`, `square`, and `parallelogram`. Each of the four receives a size parameter. The task of directing the flow to the generation of the shape chosen by the user can be accomplished as follows using an if-else statement:

```
1     if ( choice == 0 )
2     {
3       rightAngle( height );
4     }
5     else if ( choice == 1 )
6     {
7       isosceles( height );
8     }
9     else if ( choice == 2 )
10    {
11      square( height );
12    }
13    else if ( choice == 3 )
14    {
15      parallelogram( height );
16    }
```

The switch statement replaces these `if` and `else if` conditional tests with case names:

```
1     switch ( choice )
2     {
3       case 0: rightAngle( height ); break;
4       case 1: isosceles( height ); break;
5       case 2: square( height ); break;
6       case 3: parallelogram( height ); break;
7     }
```

This probably makes it clearer that the choices are 0 through 3.

The entire program of the selective shape generation appears next. First comes the part in the method `main` that receives the input from the user: the value for the selection variable `choice` (Lines 18 and 19) and the value for the height variable `height` (Lines 20 and 21). The switch statement we discussed appears in Lines 25–31. This version lacks the final `break`, since a `break` statement at the end of the body of a switch statement is redundant.

```
1  import java.util.*;
2
3  public class ShapeSelection
4  {
5    public static void main( String[] args )
6    {
7      Scanner keyboard = new Scanner( System.in );
8      int choice, height;
9
10     System.out.println( "****************************************" );
11     System.out.println( "This program prints a shape of your choice" );
12     System.out.println( "Select by entering number                " );
13     System.out.println( "0. Right-angle triangle                  " );
14     System.out.println( "1. Isosceles                             " );
15     System.out.println( "2. Square                                " );
16     System.out.println( "3. Parallelogram                         " );
17     System.out.println( "****************************************" );
18     System.out.print( "Enter your choice: " );
19     choice = keyboard.nextInt();
20     System.out.print( "Enter height: " );
21     height = keyboard.nextInt();
22
23     System.out.println( "Here is your figure!" );
24
25     switch ( choice )
26     {
27       case 0: rightAngle( height ); break;
28       case 1: isosceles( height ); break;
29       case 2: square( height ); break;
30       case 3: parallelogram( height );
31     }
32   }
33
```

Listing 10.1 A program that uses a switch statement to generate a shape (part 1). The part for receiving an input from the user

The four shape-generation methods receive an `int` parameter `height` and print `height` lines using a for-loop that iterates over the sequence `1, ..., height` with a variable named `i`. The action of the loop-body is to call a method named `line` with two parameters. Depending of the relations of the values of the two parameters to `i`, different shapes are printed.

The `line` receives two `int` parameters, `whiteWidth` and `blackWidth` (Line 34), and prints `blackWidth` hash marks after an indentation of length `whiteWidth`. The indentation is generated using a for-loop in Lines 36–39 and the hash marks are printed using a for-loop in Lines 40–43.

```
34    public static void line( int whiteWidth, int blackWidth )
35    {
36      for ( int i = 1; i <= whiteWidth; i ++ )
37      {
38        System.out.print( " " );
39      }
40      for ( int i = 1; i <= blackWidth; i ++ )
41      {
42        System.out.print( "#" );
43      }
44      System.out.println();
45    }
```

Listing 10.2 A program that uses a switch statement to generate a shape (part 2). A method `line` that produces a line of `"#"` after some indentation

For `rightAngle` (Line 47), the actual call is `line( 0, i )` (Line 51). This produces `i` hash marks with no indentation. Since the value of `i` is increasing, a right-angled triangle is produced. For `isosceles` (Line 55), the actual call is `line( height - i, i * 2 - 1 )` (Line 59). As the value of `i` increases from 1 to `height`, the length of indentation decreases from `height - 1` to 0 while the number of hash marks increases from 1 to `2 * height - 1` by 2 at a time. Therefore, an isosceles triangle is produced. For `square` (Line 63), the actual call is `line( 0, height )` (Line 67). This produces `height` hash marks with no indentation. Therefore, a square is produced. For `parallelogram` (Line 71), the actual call is `line( i - 1, height )` (Line 75). The length of indentation increases from 0 to `height - 1` while the number of hash marks is `height`. Therefore, a parallelogram is produced.

```
46
47    public static void rightAngle( int height )
48    {
49      for ( int i = 1; i <= height; i ++ )
50      {
51        line( 0, i );
52      }
53    }
54
55    public static void isosceles( int height )
56    {
57      for ( int i = 1; i <= height; i ++ )
58      {
59        line( height - i, i * 2 - 1 );
60      }
61    }
62
```

Listing 10.3 A program that uses a switch statement to generate a shape (part 3). Two shape-producing methods

```
63    public static void square( int height )
64    {
65      for ( int i = 1; i <= height; i ++ )
66      {
67        line( 0, height );
68      }
69    }
70
71    public static void parallelogram( int height )
72    {
73      for ( int i = 1; i <= height; i ++ )
74      {
75        line( i - 1, height );
76      }
77    }
78  }
```

Listing 10.4 A program that uses a switch statement to generate a shape (part 4). The two remaining shape-producing methods

Here are execution examples of the program.

```
1   ****************************************
2   This program prints a shape of your choice
3   Select by entering number
4   0. Right-angle triangle
5   1. Isosceles
6   2. Square
7   3. Parallelogram
8   Enter your choice: 0
9   Enter height: 10
10  #
11  ##
12  ###
13  ####
14  #####
15  ######
16  #######
17  ########
18  #########
19  ##########
20  ****************************************
21  This program prints a shape of your choice
22  Select by entering number
23  0. Right-angle triangle
24  1. Isosceles
25  2. Square
26  3. Parallelogram
27  Enter your choice: 1
28  Enter height: 5
29      #
30     ###
31    #####
32   #######
33  #########
```

```
****************************************
This program prints a shape of your choice
Select by entering number
0. Right-angle triangle
1. Isosceles
2. Square
3. Parallelogram
Enter your choice: 2
Enter height: 7
#######
#######
#######
#######
#######
#######
#######
****************************************
This program prints a shape of your choice
Select by entering number
0. Right-angle triangle
1. Isosceles
2. Square
3. Parallelogram
Enter your choice: 3
Enter height: 20
####################
 ####################
  ####################
   ####################
    ####################
     ####################
      ####################
       ####################
        ####################
         ####################
          ####################
           ####################
            ####################
             ####################
              ####################
               ####################
                ####################
                 ####################
                  ####################
                   ####################
```

Multiple anchors can be assigned to the same entry points. Recall the program from Chap. 6 that shows four colors and produces statements based upon the choice made (List 6.10). In that code, 1 and 2 were the official University of Miami colors and 3 and 4 were the official University of Michigan colors. Using a switch statement with `default` as an anchor, we can write the following code that behaves exactly the same way:

```
1  import java.util.Scanner;
2  // ask about a color and respond
3  public class ColorSelectionSwitch
4  {
5    public static void main( String[] args )
6    {
7      Scanner keyboard = new Scanner( System.in );
8      System.out.println( "What is your favorite color?" );
9      System.out.println( "1. Orange, 2. Green, 3. Maize, 4. Blue" );
10     System.out.print( "Select from 1 to 4 : " );
11     int answer = keyboard.nextInt();
12
13     switch ( answer )
14     {
15       case 1:
16       case 2:
17         System.out.printf( "Your choice %d is excellent.%n", answer );
18         System.out.println( "It is a U. Miami Color!" );
19         break;
20       case 3:
21       case 4:
22         System.out.printf( "Your choice %d is excellent.%n", answer );
23         System.out.println( "It is a U. Michigan Color!" );
24         break;
25       default:
26         System.out.printf( "Your choice %d is invalid.%n", answer );
27     }
28   }
29 }
```

Listing 10.5 A program that responds to a color choice

The actions to be performed are identical between the choices 1 and 2, the actions to be performed are identical between the choices 3 and 4.

10.2 Using a `char` Data in a Switch-Statement

Consider computing the number of occurrences of six punctuation marks (the period, the comma, the question mark, the exclamation mark, the colon, and the semicolon) and everything else in an input character sequence. We can use seven `int` variables corresponding to the seven categories. After initializing the variables with the value of 0, we examine each character of the input sequence and increase the variable corresponding to the character. The following program executes this, using a switch-statement for handling the selection of the variable.

The program consists of three methods:

1. The method `receiveInput` receives a multiple-line input from the user.
2. The method `printInfo` prints the name of a punctuation mark and the number of its occurrences.
3. The method `main` handles the task of counting.

The program receives multiple lines of input from the user. The number of lines that the user enters is defined as a constant `LINENUMBER`. The value of the constant is 10 (Line 4).

The method `receiveInput` receives `LINENUMBER` lines from the user and returns a `String` that connects them with the newline at the end of each line. The method uses a `StringBuilder`

object named builder to build the String to be returned (Line 11). The method announces the actions to be performed and instantiates a Scanner object for reading the keyboard (Lines 8–10). The method uses a for-loop to count the number of lines entered (Line 12). At each round of iteration, the method reads one line of input (Line 14), and then appends the line and the newline to builder (Line 15). At the end, the method converts the contents of builder to a String value and returns it.

```
1  import java.util.*;
2  public class CountPunctuations
3  {
4    public static final int LINENUMBER = 10;
5
6    public static String receiveInput()
7    {
8      System.out.printf( "Enter text of %d lines.%n", LINENUMBER );
9      System.out.println( "The program will count each punctuation." );
10     Scanner keyboard = new Scanner( System.in );
11     StringBuilder builder = new StringBuilder();
12     for ( int count = 1; count <= LINENUMBER; count ++ )
13     {
14       String line = keyboard.nextLine();
15       builder.append( line + "\n" );
16     }
17     return builder.toString();
18   }
19
```

Listing 10.6 A program that counts punctuations (part 1). The program header and a program for receiving a multiple-line input

The method printInfo receives a String representing the name of a punctuation marks and an int representing the count, and prints the two values using printf (Line 22). The String appears in 20 character spaces and the count in three character spaces. The two are separated by a colon and one white space. The method main first calls the method receiveInput to obtain a

```
20   public static void printInfo( String name, int count )
21   {
22     System.out.printf( "%20s: %3d times%n", name, count );
23   }
24
```

Listing 10.7 A program that counts punctuations (part 2). The method printInfo

multiple-line input from the user. The method stores the input in a variable input (Line 27). The method then declares the variables for counting the occurrences of the six punctuation marks. The six variables are nPeriod, nComma, nQuestion, nExclamation, nColon, and nSemicolon. They correspond to the period, the comma, the question mark, the exclamation mark, the colon, and the semicolon. In addition, the method declares a variable, nOthers, for counting the occurrences of everything else. These seven variables are initialized with the value of 0 (Lines 28 and 29). The method then executes a for-loop that iterates over the sequence 0, ..., input.length() - 1 with a variable named i (Line 30). The body of the loop is a switch statement. The switch statement

examines `input.charAt( i )` (Line 32), and then depending on the value of the `char` data, it increases the value of one counter by 1. There are six `case` anchors corresponding to the six punctuation marks (Lines 34–39), and `default` that handles the remainder (Line 40).

```
25    public static void main( String[] args )
26    {
27      String input = receiveInput();
28      int nPeriod = 0, nComma = 0, nQuestion = 0, nExclamation = 0;
29      int nColon = 0, nSemicolon = 0, nOthers = 0;
30      for ( int i = 0; i <= input.length() - 1; i ++ )
31      {
32        switch ( input.charAt( i ) )
33        {
34          case '.': nPeriod ++; break;
35          case ',': nComma ++; break;
36          case '?': nQuestion ++; break;
37          case '!': nExclamation ++; break;
38          case ':': nColon ++; break;
39          case ';': nSemicolon ++; break;
40          default: nOthers ++;
41        }
42      }
43
```

Listing 10.8 A program that counts punctuations (part 3). The part for counting the characters

The last part of the code handles the reporting of the result. First, the program prints a header (Lines 44 and 45). The header includes the input lines that the user enters. The `%s` appearing in the format `String`

```
Your input:%n%s%nThe counts:%n
```

serves as the placeholder for the input. Since `receiveInput` adds the newline after each input line received, each input line is printed in an individual line. After printing the input lines, the program prints the seven counts using the `print` method (Lines 46–52).

```
44      System.out.println( "\n======================" );
45      System.out.printf( "Your input:%n%s%nThe counts:%n", input );
46      printInfo( "Period", nPeriod );
47      printInfo( "Comma", nComma );
48      printInfo( "Question Mark", nQuestion );
49      printInfo( "Exclamation Mark", nExclamation );
50      printInfo( "Colon", nColon );
51      printInfo( "Semicolon", nSemicolon );
52      printInfo( "Others", nOthers );
53    }
54  }
```

Listing 10.9 A program that counts punctuations (part 4). The part for generating a report

Here is an execution example of the program. The user enters ten lines from *Adventures of Huckleberry Finn* by Mark Twain.[1]

```
Enter text of 10 lines.
The program will count each punctuation.
"Well, who said it was?"
"Why, you did."
"I DIDN'T nuther."
"You did!"
"I didn't."
"You did."
"I never said nothing of the kind."
"Well, what DID you say, then?"
"Said he come to take the sea BATHS--that's what I said."
"Well, then, how's he going to take the sea baths if it ain't on the sea?"
========================
Your input:
"Well, who said it was?"
"Why, you did."
"I DIDN'T nuther."
"You did!"
"I didn't."
"You did."
"I never said nothing of the kind."
"Well, what DID you say, then?"
"Said he come to take the sea BATHS--that's what I said."
"Well, then, how's he going to take the sea baths if it ain't on the sea?"

The counts:
               Period:   6 times
                Comma:   6 times
       Question Mark:   3 times
    Exclamation Mark:   1 times
                Colon:   0 times
            Semicolon:   0 times
               Others: 279 times
```

Here is another program that uses a switch statement. This time, the program uses a `char` data for directing the flow. This program receives four pieces of information about a person (name, age, gender, and phone number) from the user, and stores the information in four variables. The program then receives a text input from the user. The program examines the first character of the text input, obtained using `charAt( 0 )`. If the character matches one of the four letters (`'N'`, `'A'`, `'G'`, and `'P'`) representing the four pieces of information, the program prints the information; otherwise, the program prints an error message.

The program uses `String` variables, `name`, `gender`, and `phone`, to store information about the name, gender, and phone number (Line 6). The program uses a `String` variable, `choice`, to record the input the user enters as the choice of information to recall (Line 6). The program uses an `int` variable, `age`, to record the age (Line 7). The program uses interactions with the user to receive values for the four variables (Lines 11–18). To recall information, the program prints a prompt, and then receives input in the variable `choice` (Lines 20–22). Then, the program examines `choice.charAt( 0 )` in a switch statement to decide which variable to recall (Lines 26–33). Any invalid choice is directed to the `default:` in Line 34.

[1]Samuel Langhorne Clemens (November 30, 1835 to April 21, 1910), known by his pen name Mark Twain, was an American writer.

```
import java.util.*;
public class SelectionByChar
{
  public static void main( String[] args )
  {
    String name, gender, phone, choice;
    int age;

    Scanner keyboard = new Scanner( System.in );

    System.out.print( "Type name: " );
    name = keyboard.next();
    System.out.print( "Type age: " );
    age = keyboard.nextInt();
    System.out.print( "Type gender: " );
    gender = keyboard.next();
    System.out.print( "Type phone: " );
    phone = keyboard.next();

    System.out.println( "Enter information to recall." );
    System.out.print( "  A(ge), G(ender), N(ame), P(hone): " );
    choice = keyboard.next();

    switch ( choice.charAt( 0 ) )
    {
      case 'A': System.out.println( "The age is " + age );
        break;
      case 'G': System.out.println( "The gender is " + gender );
        break;
      case 'N': System.out.println( "The name is " + name );
        break;
      case 'P': System.out.println( "The phone is " + phone );
        break;
      default:  System.out.println( "Unsupported selection!" );
    }
  }
}
```

Listing 10.10 A program that uses a switch statement for recalling memory

Here are some execution examples of the code. In the first example, the user asks to recall the age by entering `"A"`:

```
Type name: Emily
Type age: 30
Type gender: Feale
Type phone (use dash): 333-333-3333
  A(ge), G(ender), N(ame), P(hone): A
The age is 30
```

In the second example, the user asks to recall the name by entering the prefix `"Nam"` of `"Name"`:

```
Type name: Dwight
Type age: 32
Type gender: Male
Type phone (use dash): 123-123-1122
  A(ge), G(ender), N(ame), P(hone): Nam
The name is Dwight
```

In the third example, the user asks to recall the phone number by entering "Pine", whose first letter is 'P':

```
Type name: George
Type age: 40
Type gender: Male
Type phone (use dash): 343-343-3344
  A(ge), G(ender), N(ame), P(hone): Pine
The phone is 343-343-3344
```

In the last example, the user enters "Q", but there is no matching choice, so the program prints the error message:

```
Type name: Caroline
Type age: 23
Type gender: Female
Type phone (use dash): 333-222-1111
  A(ge), G(ender), N(ame), P(hone): Q
Unsupported selection!
```

10.3 Using a String Data in a Switch Statement

An alternate version of the memory recall program show next uses a switch statement that examines the value of a String data. In this version, to recall information, the user must type the exact name of the information to recall. The program handles the input that the user enters in a non-case-sensitive manner by converting it to all lowercase using the method toLowerCase() of String (Line 22).

```
1  import java.util.*;
2  public class SelectionByString
3  {
4    public static void main( String[] args )
5    {
6      String name, gender, phone, choice;
7      int age;
8
9      Scanner keyboard = new Scanner( System.in );
10
11     System.out.print( "Type name: " );
12     name = keyboard.next();
13     System.out.print( "Type age: " );
14     age = keyboard.nextInt();
15     System.out.print( "Type gender: " );
16     gender = keyboard.next();
17     System.out.print( "Type phone: " );
18     phone = keyboard.next();
19
```

Listing 10.11 A program for recalling memory (part 1). The part responsible for receiving data from the user

```
20      System.out.println( "Enter information to recall." );
21      System.out.print( "  age, gender, name, phone: " );
22      choice = keyboard.next().toLowerCase();
23
24      switch ( choice )
25      {
26        case "age": System.out.println( "The age is " + age );
27          break;
28        case "gender": System.out.println( "The gender is " + gender );
29          break;
30        case "name": System.out.println( "The name is " + name );
31          break;
32        case "phone": System.out.println( "The phone is " + phone );
33          break;
34        default:  System.out.println( "Unsupported selection!" );
35      }
36    }
37  }
```

Listing 10.12 A program for recalling memory (part 2). The part responsible for recalling information

Here are a couple execution examples of the code:

```
1  Type name: Darrell
2  Type age: 18
3  Type gender: Male
4  Type phone (use dash): 555-555-5555
5    Age, Gender, Name, Phone: age
6  The age is 18
```

```
1  Type name: Eden
2  Type age: 19
3  Type gender: Female
4  Type phone (use dash): 777-888-9999
5    Age, Gender, Name, Phone: name
6  The name is Eden
```

Summary

- A switch statement allows selecting an action to perform based on the value of a data. The possible data types that can be used in a switch statement are: `char`, `int`, `long`, `short`, and `String`.
- Each case value in a switch statement is specified using `case VALUE:` where `VALUE` is a literal matching the type of the variable examined in the switch statement.
- No two cases appearing in a switch statement have the same values.
- `default:` is a keyword that corresponds to "otherwise".
- If `break` is encountered during the execution of the body of a switch statement, the execution is immediately terminated.

Exercises

1. **Converting to and from a switch statement, 1** Convert the following switch statement to an equivalent if-else statement:

```
switch ( a )
{
  case 1: System.out.println( "A" ); break;
  case 2: System.out.println( "B" ); break;
  case 3: System.out.println( "C" ); break;
  default:
}
```

2. **Converting to and from a switch statement, 2** Convert the following switch statement to an equivalent if-else statement:

```
switch (a)
{
  case 1: System.out.println( "X" );
  case 2: System.out.println( "Y" ); break;
  case 3: System.out.println( "Z" ); break;
  default:
}
```

3. **Converting to and from a switch statement, 3** Convert the following switch statement to an equivalent if-else statement:

```
switch ( a )
{
  case 1: System.out.println( "i" );
  case 2: System.out.println( "ii" );
  case 3: System.out.println( "iii" ); break;
  default:
  case 4: System.out.println( "iv" ); break;
}
```

4. **Converting to and from a switch statement, 4** Convert the following if-else statement to an equivalent switch statement:

```
if ( a == 0 )
{
  System.out.println( "A" );
}
else if ( a == 1 )
{
  System.out.println( "B" );
}
else
{
  System.out.println( "C" );
}
```

5. **Converting to and from a switch statement, 5** Convert the following if-else statement to an equivalent switch statement:

```
if ( a == 0 )
{
  System.out.println( "F" );
}
else if ( a == 1 )
{
  System.out.println( "T" );
}
```

6. **Converting to and from a switch statement, 6** Convert the following switch statement to an equivalent if-else statement, where `a` is an `int` data:

```
switch ( a )
{
  case 0: System.out.println( "X" ); break;
  case 1: System.out.println( "Y" ); break;
  case 3: System.out.println( "W" );
  case 2: System.out.println( "Z" ); break;
  default: System.out.println( "?" );
}
```

Programming Projects

7. **Processing a sequence of 2d movements** Write a program named `Movements2D` that receives a sequence of characters composed solely of `'U'`, `'D'`, `'L'`, and `'R'` from the user, and processes it as a sequence of moves in the two-dimensional grid, where the initial position is $(0, 0)$. The letters `'U'`, `'D'`, `'L'`, and `'R'` correspond to increases the y-coordinate by 1, decreases the y-coordinate by 1, increases the x-coordinate by 1, and decreases the x-coordinate by 1. Use a switch statement to process an individual letter. Use a for-loop to go through the input, and for each move made, print the new location. For instance, if the input sequence is `"ULDDRU"`, the output of the program must be:

```
(0,1)
(-1,1)
(-1,0)
(-1,-1)
(0,-1)
(0,0)
```

The characters other than `'U'`, `'D'`, `'L'`, and `'R'` will be treated as "not moving".

8. **Processing a sequence of 3d movements** Write a program named `Movements3D` that receives a sequence of characters composed solely of `'F'`, `'B'`, `'U'`, `'D'`, `'L'`, and `'R'` from the user, and processes it as a sequence of moves in the three-dimensional grid, where the initial position is $(0, 0, 0)$. The letters `'F'`, `'B'`, `'U'`, `'D'`, `'L'`, and `'R'` increases the x-coordinate by 1, decreases the x-coordinate by 1, increases the z-coordinate by 1, decreases the z-coordinate by 1, increases the y-coordinate by 1, and decreases the y-coordinate by 1. Use a switch-statement to process an individual letter. Use a for-loop to go through the input, and for each move made, print the new location. For example, if the input sequence is `"UFLDDRUB"`, the output of the program must be:

```
(0,0,1)
(1,0,1)
(1,-1,1)
(1,-1,0)
(1,-1,-1)
(1,0,-1)
(1,0,0)
(0,0,0)
```

The characters other than the six letters will be treated as "not moving".

9. **Rotating vowels** Write a program named `RotateVowels` that receives a `String` data from the user, and replaces each letter in the input corresponding to a vowel to another vowel. The replacing rule is as follows: `'a'` become `'e'`, `'e'` become `'i'`, `'i'` become `'o'`, `'o'` become `'u'`, `'u'` become `'a'`, `'A'` become `'E'`, `'E'` become `'I'`, `'I'` become `'O'`, `'O'` become `'U'`, `'U'` become `'A'`, and any other character stays the same. For example, the method returns `"Luaos"` when given `"Louis"` as input, and returns `"Iest"` when given `"East"` as input. Write the code so that the method builds its output using a `StringBuilder` object. The program reads the examines the input character by character, and then appends the character after conversion to the `StringBuilder` object. Use a switch statement in determining which character must be appended.
10. **Treasure hunting in 2D** Write a program named `TreasureHunting2D` that plays the game of finding a treasure hidden at a location on the 2D grid, where the coordinates are integers between 1 and 10, where the player can make at most ten guesses and the program provides up to two pieces of advice depending on the players's guesses.
    - The first piece of advice is based upon the Manhattan distance between the true location and the guess, where the distance value is the sum of the absolute difference in the x-coordinates and the absolute difference in the y-coordinates.
      (a) If the distance is 0, the program announces:
      `"You have found the treasure!"`.
      (b) If the distance is between 1 and 3, the program announces:
      `"The treasure is very close."`.
      (c) If the distance is between 4 and 6, the program announces:
      `"The treasure is somewhat close."`.
      (d) If the distance is greater than 6, the program announces:
      `"The treasure is not close."`.
    - The second piece of advice is given in the second round and onwards, and appears only when the guess is still incorrect. The advice informs if the present guess is closer than the previous guess with the statement, `"You are closer."`, `"You are farther."`, or `"The same distance."`.

    Make sure that after the user has correctly found the location, the execution of the loop-body will skip even if there are more rounds remaining.