L16

Monday, January 24, 2022 2:40 PM

switch-case statement, for-each syntax and applications

Saurav Hathi

https://www.youtube.com/channel/UCp6MFWao5vWRnyRCxBsKnfw

switch-case - syntax and application

Q1. A switch statement allows the value of a variable or expression to change the control flow of program execution through multiple paths.
Let us consider that for a given random number between 1 and 10, we want to print the corresponding English word i.e. if 2 is given we print Two, 5 is given we print Five and so on.
One way is to write a long nested if-else-if for the 10 numbers, the other way is to use a switch statement.

Below is the basic syntax of a switch statement:

```
int value = 4;
        switch (value) { // start of switch-blockcase 1:
                                System.out.println("One");
                                break; // this ensures that we exit the switch block after printing Onecase 2:
                                System.out.println("Two");
                                break;
                        case 3:
                        ...
                        ...
                        default:
                                System.out.println("Number " + value + " is not between 1 and 10");
        } // end of switch-block
```

1. The switch (aValue) statement takes a variable/reference or an expression as the aValue and always has a switch-block.
2. The switch-block starts with an opening-brace { and concludes with a closing-brace }.
3. The case 1:, case 2: and default : are called labels.
4. The code below a case aValue: label is executed whose aValue matches the value passed into the switch (aValue) statement.
5. If no case matches the aValue passed in the the switch (aValue) statement, then the code under default: label is executed if it exists.
6. The switch block can contain any number of case : labels and one default: label.
7. The break; statement is a branching statement. It causes transfer of execution immediately to the end switch block, skipping all other cases if any below the break;.

Note: char, byte, short, int, Character, Byte, Short, Integer, String, or an enum type are the valid types whose values can be passed into a switch(aValue) statement.

In the below program the class PrintTextForm is passed a number as argument into its main(...) method. The code in the main(...) method converts the first argument it receives from String into an int and employs a switch statement to print the text form of the number.

Fill in the missing code to handle and print the missing numbers between 1 to 10.

Note: Make sure to use println(...) method and not print(...). Ensure that the spelling of the number starts with a **capital** letter.

q10870/PrintTextForm.jav

```java
package q10870;
public class PrintTextForm {
    public static void main(String[] args) {
        int value = Integer.parseInt(args[0]);
        switch (value) {
            case 1:
                System.out.println("One");
                break;
            case 2:
                System.out.println("Two");
                break;
            // Fill in the missing code below for the remaining cases
            case 3:
                System.out.println("Three");
                break;
            case 4:
                System.out.println("Four");
                break;
            case 5:
                System.out.println("Five");
                break;
            case 6:
                System.out.println("Six");
                break;
            case 7:
                System.out.println("Seven");
                break;
            case 8:
                System.out.println("Eight");
                break;
            case 9:
                System.out.println("Nine");
                break;
            case 10:
                System.out.println("Ten");
                break;
            default:
                System.out.println("Number " + value + " is not in the range 1 to 10");
        }
    }
}
```

Q2. The syntax for executing common code on similar conditions in a switch statement is given below:

```
switch (val) {
        case 1:
        case 2:
                statement_1; //do something
                break;
        case 3:
        case 4:
                statement_2; //do something else
                break;
        default:
                statement_3; //do yet another thing
}
```

In the above switch statement, when the val is 1 or 2, statement_1 will be executed.

Similarly, when the val is 3 or 4, statement_2 will be executed. And for all other values statement_3 under the default: case will be executed.

The below program with class named HappyDays is passed a String argument into its main(...) method.

The code in the main(...) method uses the switch statement to mark the type of day.

Fill in the missing code in the switch statement such that it prints saying the **given day** :

1. is a Working Day - when the arg[0] is equal to Monday or Tuesday or Wednesday or Thursday
2. is a Semi-Working Day - when the arg[0] is equal to Friday
3. is a Happy Day! - when the arg[0] is equal to either Saturday or a Sunday
4. is not a valid day of week - when the text in arg[0] does not represent a day in a week

q10871/HappyDays.java

```java
package q10871;
public class HappyDays {
    public static void main(String[] args) {
        // Fill in the missing code inside the switch statement
        switch (args[0]) {
            case "Monday":
            case "Tuesday":
            case "Wednesday":
            case "Thursday":
                System.out.println(args[0] + " is a Working Day");
                break;
            case "Friday":
                System.out.println(args[0] + " is a Semi-Working Day");
                break;
            case "Saturday":
            case "Sunday":
                System.out.println(args[0] + " is a Happy Day!");
                break;
            default:
                System.out.println(args[0] + " is not a valid day of week");
        }
    }
}
```

Q3. In a switch statement when a break; is not used under a case, a fall-through happens.

Meaning when the **value** matches such a case, the control flow will execute the **statements** under that case and will continue to execute other cases present under it until it encounters a break; statement.

For example, click on the Submit without making any changes to the below code to observe the fall-through. Observe the output when 1, 2, 3, 4 or 10 is passed.

In the below program, class FallThroughDemo is passed an integer argument into its main(...) method.

The code in the main(...) method converts the first argument available in args[0] into an integer value and uses it in the switch statement.

In the program given below, add break; statements appropriately inside the switch statement such that when :

- 1 is passed, the code should print
  One
  Two
  Three
- 2 is passed, the code should print
  Two
  Three
- 3 is passed, the code should print
  Three
- 4 is passed, the code should print
  Four
- 10 is passed, the code should print
  Ten
- any other number except 1, 2, 3, 4 and 10 is passed, the code should print
  Some other number

q10872/FallThroughDemo

```java
package q10872;
public class FallThroughDemo {
    public static void main(String[] args) {
        int value = Integer.parseInt(args[0]);
        switch (value) {
            case 1:
                System.out.println("One\nTwo\nThree");
                break;
            case 2:
                System.out.println("Two\nThree");
                break;
            case 3:
                System.out.println("Three");
                break;
            case 4:
                System.out.println("Four");
                break;
            case 10:
                System.out.println("Ten");
                break;
            default:
                System.out.println("Some other number");
        }
    }
}
```

Q4. Write a class NameOfTheDay with a **main** method. The method receives one command line argument. The method should convert the argument **String** to **int**.

The program should print the name of the weekday based on the value of the day (1-7), using the switch statement.

For example:
Cmd Args : 3
Wednesday
Similarly, if the input is given as 10 then the output should be **Number 10 is not in the range 1 to 7**.

For example:
Cmd Args : 10
Number 10 is not in the range 1 to 7
Ensure that the name of the day starts with a **capital** letter.

q10874/NameOfTheDay.j

```java
package q10874;

public class NameOfTheDay{
    public static void main(String[] args) {
        int value = Integer.parseInt(args[0]);

        switch(value){
            case 1:
                System.out.println("Monday");
                break;
            case 2:
                System.out.println("Tuesday");
                break;
            case 3:
                System.out.println("Wednesday");
                break;
            case 4:
                System.out.println("Thursday");
                break;
            case 5:
                System.out.println("Friday");
                break;
            case 6:
                System.out.println("Saturday");
                break;
            case 7:
                System.out.println("Sunday");
                break;
            default:
                System.out.println("Number "+value+" is not in the range 1 to 7");

        }
    }
}
```

for-each - syntax and application

Q1. A loop in a programming language is a statement which allows code to be repeatedly executed. A for-each loop is a kind of control statement for iterating over a collection of items.

```
for (type item: iterableCollection) {
        // code statement 1// code statement 2...
}
```

1. A for-each statement starts with for (type item: iterableCollection)

followed by its body.

1. It is a good practice to always keep the body which contains the code to be executed within an opening-brace { and a closing-brace }.
2. It is called a for-each loop because the statements contained in the body are executed once for each item in iterableCollection

The class ForEachDemo creates an array of integers named numArr. (We will learn about arrays in later sections.)
The code in the main(...) method should use a for-each loop to iterate over the array numArr.
Fill in the missing code for the for-each statement such that it initializes int i with each element in numArr and executes the code in the for-each statement's body to print i.

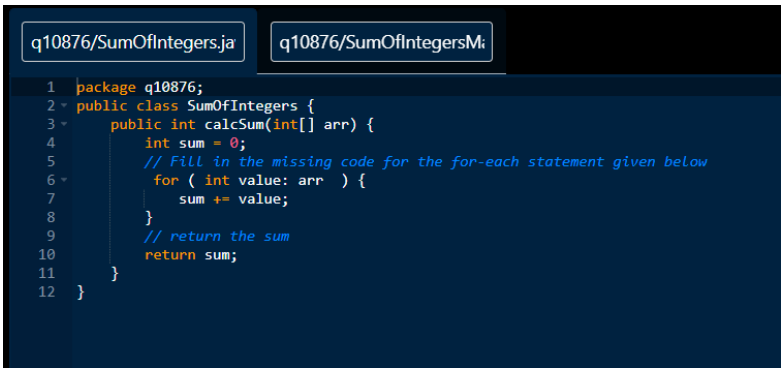
q10875/ForEachDemo.jav

```java
package q10875;
public class ForEachDemo {
    public static void main(String[] args) {
        int[] numArr = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
        for ( int i: numArr) {
            System.out.println(i);
        }
    }
}
```

Q2. Write a class SumOfIntegers with a **public** method calcSum that takes one parameter arr of type int[] and returns the sum of all integers in the arr. The return type of calcSum should be int.

Here is an example:

Cmd Args : 1 3 5 7 9
Sum = 25



Q3. Write a class PrintEvenNumbers with a **public** method printEvenNumbers that takes one parameter arr of type int[].

Write code using for-each loop to iterate over the arr if the number is even the program should print **is even** otherwise, it should print **is not even**.

Here is an example:
Cmd Args : 13 14 15
13 is not even
14 is even
15 is not even
Hint: You can use value % 2 == 0. Meaning, you can use the modulus operator (%) to divide a **number** by **2** and get the remainder, to verify if the reminder is equal to **0**.

Q4. Write a class PrintOddNumbers with a **public** method printOddNumbers that takes one parameter arr of type int[].

Use for-each loop to iterate over the arr and the program should print **is odd** if the element is odd otherwise, it should print **is not odd**.

For example:
Cmd Args : 1 2 3 4 5 6
1 is odd
2 is not odd
3 is odd
4 is not odd
5 is odd
6 is not odd