Switch Statement

"check a variable's value against a list of options "cases" & run different pieces of code based on which value matches. It's simpler way to write multiple if-else statements when you're comparing the same variable to different values."

Working :-

1. provide a variable (like name) to the switch statement.
2. The program compares name with the values in each "case".
3. When it finds a match, it runs the code for that case.
4. If none of the cases match, you can have a "default" case that runs.

Allowed Data types	Not allowed datatypes
1) int.	1) float.
2) byte.	2) double.
3) Short.	3) long.
4) char.	4) boolean
5) String.	5) other complex objects (like
6) enum types.	custom classes, arrays collections)

Problem statement

Write a Java program to handle user subscription statuses using a switch statement.

Scanner : take input from user.

Switch statement : Determine the o/p based on the subscription status entered by the user.

1. "subscribed" : "You are subscribed. You will receive notifications."

2. "not subscribed" : "No Notifications. Please subscribe to receive notifications."

3. "trial" : "You are on a trial. Subscribe soon to continue receiving notifications."

4. "expired" : If user "Your subscription has expired. Renew to receive notifications".

5. "default" : "Invalid input. Please enter a valid subscription status."

The program should handle case-insensitive input (e.g. subscribed, Subscribed or SUBSCRIBED.)

Code.

```
Scanner scan = new Scanner(System.in);

// Taking i/p from user for subscription status.

System.out.println("Enter your subscription status
    (subscribed, not subscribed, trial, expired): ");

String status = scan.nextLine().toLowerCase();

// Convert i/p to lowercase to avoid case sensitivity.

// Switch statement to handle different cases.

switch(status) {
    case "subscribed":
        Sout("You are subscribed. You will receive
            notifications.");
        break;

    case "not subscribed":
        Sout("No notifications. Please subscribe to
        receive notifications.");
        break;

    case "trial":
        Sout("You are on a trial. Subscribe soon
        to continue receiving notifications.");
        break;
```

```
case "expired":
        Sout("Your subscription has expired.
        Renew to receive notifications.");
        break;

default:
        Sout(" Invalid input. Please enter a
        valid subscription status.");
        main(args);
        [redacted]
}

scan.close();
}
}
```

toLowerCase() method :-

This method converts all characters of the String into lower case, & returns the lower-case String.

for ex

```
String s1 = "UPPERCASE";

Sout (s1.toLowerCase());
```

↓

o/p uppercase

Date
Page

break statement

→ break is used inside a loop to come out of it.

→ break is used inside the switch block to come out of the switch block.

→ break can be used in nested blocks to go to the end of a block. Nested blocks represents a block written within another blocks.

Java Calculator using switch

★ Write a java program that performs basic arithmetic operations.

1. Prompt the user for 2 integers.
2. Display a menu with options:
   - 1. Add
   - 2. Substract
   - 3. Divide
   - 4. Multiply.
3. Execute the corresponding operation based on user choice.
4. Handle division by zero with an error message.
5. Re-prompt the user for valid input if an invalid option is selected.
6. End the program after a valid operation.

Code

```
private static void displayMenu() {
    Sout(" Select a choice: ");
    Sout(" 1. add ");
    Sout(" 2. Substract");
    Sout(" 3. divide");
    Sout(" 4. multiply");
}

private static void add(int a, int b) {
    int result = a + b;
    Sout(" Addition is: " + result);
}
```

```
private static void sub(int a, int b){
    int result = a - b;
    Sout("Substraction is: " + result);
}

private static void div(int a, int b){
    if(b==0){
        Sout("Error: Division by zero is undefined.");
    }
    else{
        int result = a/b;
        Sout("Division is: " + result);
    }
}

private static void mul(int a, int b){
    int result = a * b;
    Sout("Multiplication is: " + result);
}

public static void main(String[] args){
    Scanner scan = new Scanner(System.in);
    while(true){
        Sout("Enter 1st number: ");
        int num1 = scan.nextInt();
        Sout("Enter 2nd number: ");
        int num2 = scan.nextInt();
        displayMenu();
        int choice = scan.nextInt();
```

```
switch (choice) {
    case 1:
        add(num1, num2);
        break;

    case 2:
        sub(num1, num2);
        break;

    case 3:
        div(num1, num2);
        break;

    case 4:
        mul(num1, num2);
        break;

    default:
        Sout("Error, Please select a valid
            choice!");
        continue;  // to loop back to menu.

    }
    break;  // exit the loop after a valid
            operation
}
Scan.close();
```

continue

In Java, the continue statement is used to skip the current iteration of a loop & continue with the next iteration. When encountered, it causes the loop to bypass the remaining code in its current iteration & move directly to the loop's next iteration.

```
for(int i=0; i<10; i++){
    if (i%2 == 0){
        continue; // skip the current iteration
                  if i is even.
    }
    Sout(i);      // prints only odd number
}
```

When i is even number, continue statement causes the loop to skip printing i & go straight to the next iteration.

continue statement is useful when you want to skip certain conditions or steps inside a loop.