System...

Switch Statements + Nested Case in java

Topic

(•) Switch statement

In switch cases you can jump to various cases equal based on your expression.

Syntax :

```
switch (expression){

    // case
    Case one :
        // do something
    break;

    Case two :
        // do something
    break;

    default :
        // do something
```

Note :

- cases have to be the same type as expression, must be const. or literal
- duplicate case values are not allowed
- break is use to terminate the sequence
- if break is not used, it will continue to next case
- default will execute when none of the above does
- if default is not at the end, put break after it.

```
}
System.out.println(ans);
```

Topic Switch Statements + Nested Case in java

(●) Switch Statement

In Switch cases you can jump to various cases based on your expression.

Syntax :

```
switch (expression){

    // case
    Case one:
        // do something
        break;

    Case two:
        // do something
        break;

    default:
        // do something
```

Note :

- cases have to be the same type as expression, must be const. or literal
- duplicate case values are not allowed
- break is use to terminate the sequence
- if break is not used, it will continue to next case
- default will execute when none of the above does
- if default is not at the end, put break after it.

Date : / /

Problem 1 Describe Fruit

code:

```
String fruit = it.next();   or input.next();

Switch (fruit) {
    case "Mango":
        System.out.println("King of fruits");
    case "Apple":          > break;
        Sout ("A sweet red fruit");
    case "Orange":         > break;
        Sout ("Round fruit")
        break;
    Case "Grapes":
        Sout ("small fruits");
        break;
    default:
        Sout ("please enter a valid fruit");
}
```

click & Alt + Enter ↓ directly change the whole code to ENHANCED SWITCH which is much clean

or

```
Switch (fruit) {
    case "Mango" -> Sout ("King of fruit");
    case "Apple" -> Sout ("A sweet red fruit");
    case "Orange" -> Sout ("Round fruits");
    case "Grapes" -> Sout ("small fruits");
    default -> Sout ("please enter a valid fruit")
```

New Syntax:

```
switch (expression) {
    case one → // do this;
    case two → // do this;
    default → // do this;
}
```

No.

Date : / /

(●) Nested Switch Case :

```
switch (expression) {
        case one :
            // code block
        break ;
    case two :
        // code : // code block
        break ;
    Case three :
        switch (expression) {
            case one :
                // code block
            break ;
            case two :
                // code block
            break ;
            default ;
                // code block
        }
        break ;
    default :
        // code block
}
```

Problem: Display day name b/w 1 & 7.

```
code = int day = in.nextInt();
       switch (day) {
           case 1 -> Sout ("Monday");
           case 2 -> Sout ("Tuesday");
           case 3 -> Sout ("Wednesday");
           case 4 -> Sout ("Thursday");
           case 5 -> Sout ("friday");
           case 6 -> Sout ("saturday");
           case 7 -> Sout ("Sunday");
       }
```

Problem: Weekdays & Weekends

```
int day = in.nextInt();
switch (day) {
    case 1, 2, 3, 4, 5 -> Sout ("Weekdays");
    Case 6, 7 -> Sout ("weekends");
}
```

(•) ~~Nested switch Statement Case :~~

~~code~~ ~~Syntax.~~