Code at Random (OPC) PVT. LTD.

Module 4: Introduction To Switch Case

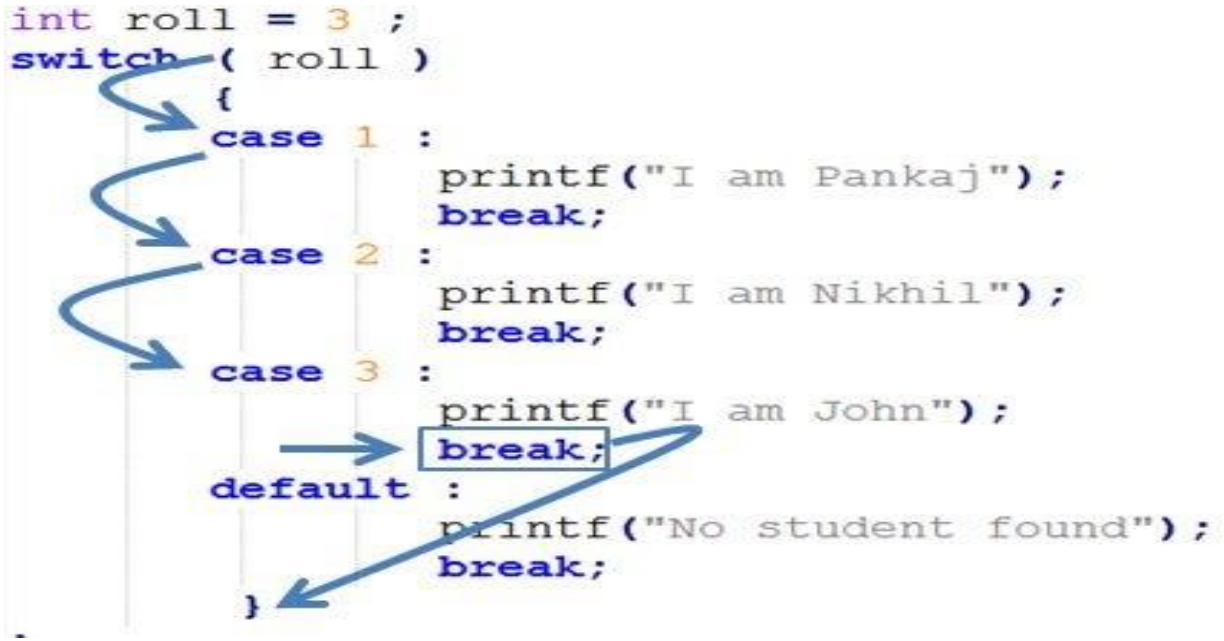
What is Switch Case?

- In Java, a switch statement is used to transfer control to a particular block of code, based on the value of the variable being tested.
- Switch statements are an efficient alternative for if-else statements.
- The switch is passed a variable, the value of which is compared with each case value. If there is a match, the corresponding block of code is executed.
- Syntax:

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

What is Break Keyword

- When Java reaches a break keyword, it breaks out of the switch block.
- This will stop the execution of more code and case testing inside the block.
- When a match is found, and the job is done, it's time for a break. There is no need for more testing.
- A break can save a lot of execution time because it "ignores" the execution of all the rest of the code in the switch block.



What is Default Keyword

- The default keyword specifies some code to run if there is no case match:
- Note that if the default statement is used as the last statement in a switch block, it does not need a break.
- Example:

```
int day = 4;
switch (day) {
  case 6:
    System.out.println("Today is Saturday");
    break;
  case 7:
    System.out.println("Today is Sunday");
    break;
  default:
    System.out.println("Looking forward to the Weekend");
}
// Output: "Looking forward to the Weekend"
```

Example Using Switch Case

```
int day = 4;
switch (day) {
  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;
}
```

Output: "Thursday" (day 4)

Multiple If-Else Vs Switch Case

Multiple If-Else	Switch Case
• It can work with all relational operators.	• It is used to test the equality.
• It can handle ranges.	• It cannot handle ranges.
• It will work with variable as well as constant.	• Case must be constant.
• It can handle floating point.	• It cannot handle floating point.
• It can work with expression	• It cannot work with expression.
• It gives low performance.	• It gives high performance.
• It takes more time to execute as compared to switch case.	• It takes less time than if-else.

Fall Through

When break statement is missing after any case, then the statements present in the next case will get executed. Such situation is known as fall-through.

Disadvantages of Multiple If-Else

- Low performance
- Lengthy
- Complexity

Programs Based On Switch Case

1. WAP to input a number (digit) & convert to words.
2. WAP to input month number & convert month name.
3. WAP to input number from 1 to 10 and print it in roman numerals.
4. WAP to input number from 1 to 10 and print if it is even or odd.

Solution of the above programs

1. WAP to input a number (digit) & convert to words.

```
import java.util.*;
class digit {
   public static void main(String Args[])    {
      int digit;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a number");
      digit = sc.nextInt();
      switch(digit)       {
         case 0:
         System.out.println("Zero");
         break;
         case 1:
         System.out.println("One");
         break;
         case 2:
         System.out.println("Two");
         break;
        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;
         default:
         System.out.println("Invalid Choice");
      }
   }
}
```

Solution of the above programs

3. WAP to input number from 1 to 10 and print it in roman numerals.

```
import java.util.*;
class roman  {
   public static void main(String Args[])    {
      int n;
      Scanner sc= new Scanner(System.in);
      n = sc.nextInt();
      switch(n)       {
         case 1:
         System.out.println("I");
         break;
         case 2:
         System.out.println("II");
         break;
         case 3:
         System.out.println("III");
         break;
         case 4:
         System.out.println("IV");
         break;
         case 5:
         System.out.println("V");
         break;
         case 6:
         System.out.println("VI");
         break;
         case 7:
         System.out.println("VII");
         break;
         case 8:
         System.out.println("VIII");
         break;
         case 9:
         System.out.println("IX");
         break;
         case 10:
         System.out.println("X");
         break;
         default:
         System.out.println("Invalid Choice");
      }
   }
}
```

Solution of the above programs

4. WAP to input number from 1 to 10 and print if it is even or odd

```
import java.util.*;
class even_odd {
   public static void main(String Args[])    {
      int n;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a number");
      n= sc.nextInt();
      switch(n)       {
         case 1:
         case 3:
         case 5:
         case 7:
         case 9:
         System.out.println("Number is odd");
         break;
         case 2:
         case 4:
         case 6:
         case 8:
         case 10:
         System.out.println("Number is even");
         break;
         default:
         System.out.println("Invalid Choice");
      }
      sc.close();
   }
}
```