**Day-17**

**11-02-2025**

**==============**

**Switch Statement:**

**============**

-> switch is keyword which we can use to define the selection statement.

**When we need switch?**

**===============**

Let us assume:

there is a variable like "x"

int x = 3;

=> I want to compare 'x' with four values:

1, 2, 3 and 4

if(x == 1)

System.out.println("one");

else if(x == 2)

System.out.println("two");

else if(x == 3)

System.out.println("three");

else if(x == 4)

System.out.println("four");

else

System.out.println("Invalid choice");

=> When you want write multiple conditions on same variable and the condition is always with "equality" then we can always use "switch statement".

if(x > 20)

System.out.println("Satisfied");

else

System.out.println("Not Satisfied");

-> When we want write multiple conditions on a single variable and the condition is with other than "equality" then we can use "if-else".

**Switch Syntax:**

**==========**

switch(variable/expression){

case value1: statement1;

break;

case value2: statement2;break;

case value3: statement3;break;

.......

case valuen: statement;break;

default: statement;

}

ex:

Let us assume:

there is a variable like "x"

int x = 3;

=> I want to compare 'x' with four values:

1, 2, 3 and 4

int x = 13;

switch(x){

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;

default:System.out.println("Invalid Choice");

}

**break:**

**====**

-> keyword

-> break can be used in switch and loops

-> to get immediate termination/stopped from the program, we can use "break"

**How switch can work?**

**==============**

-> switch can validate the value of the variable/expression with each and case value. If the value/expression can match with any of case value, then that particular case block can execute. And because of the break the program can be terminated from executed block.

-> When the variable/expression is not matching with any of the given case values, then the control can make execute "default" block automatically.

Note:

====

break is mandatory for case value blocks.

break is not required for default block.

default is optional block.

**default block in switch:**

**===============**

-> default is optional in switch block.

-> writing of break is optional in default block.

-> We can able to define default at the top or at the end of switch block, but the flow of execution for the default is after all the case value blocks only.

class SwitchStatement{

public static void main(String[] args)

{

int x = 1;

switch(x)

{

default:System.out.println("Invalid choice");

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 label value can always allow to define with any integral type (byte/short/int/long) or character type only.

If we have to define with other than above options, we can get "compile time error".

class SwitchStatement{

public static void main(String[] args)

{

double price = 1.75;

switch(price){

case 1.5:System.out.println("Hi");

break;

case 1.75:{

System.out.println("Hello");

break;

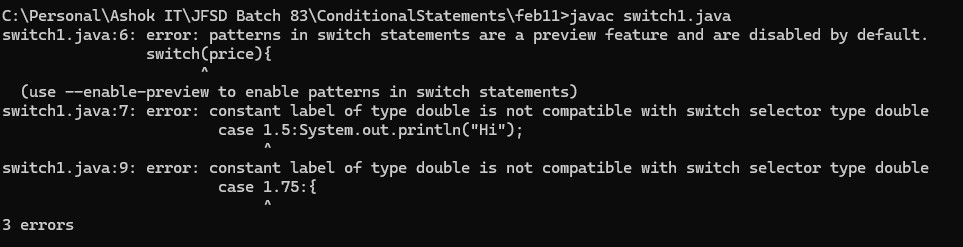
}

}

}

}

Error: compile time error



-> switch case labels allowed to define with strings also.

class SwitchStatement{

public static void main(String[] args)

{

String day = "Monday";

switch(day){

case "Sunday":System.out.println("one");break;

case "Monday":System.out.println("two");break;

case "Tueday":System.out.println("three");break;

case "Wednesday":System.out.println("four");break;

case "Thirsday":System.out.println("five");break;

case "Fridday":System.out.println("six");break;

case "Saturday":System.out.println("seven");break;

default:System.out.println("Invalid");

}

}

}

**Day-18**

**17-02-2025**

**================**

Note:

=====

1) switch, case and default are the keywords which we can use to define the switch statement.

2) break is mandatory for each case block. If you are not defining break in the case block, then the statements from satisfied case block to till the last can able to execute.

3) default block is optional.

4) break in default is also optional.

**switch with expression:**

**===============**

Expression:

=======

-> it is the combination of variables and symbols/operators (represents an operations)

Ex: + ==> sum

- ==> minus etc.

Ex: a+b-c

-> Identifiers can allow to define with different text formats:

1) Lower case

2) Upper case

3) camel case

Ex: javaFullStack

4) Capitalize ==> Javafullstack

5) Title ==> JavaFullStack

class SwitchPractice{

public static void main(String[] args)

{

int x = 5; // compile time variable definition

switch(x\*2){

case 10:System.out.println("Hi");break;

case 20:System.out.println("Bye Bye");break;

default: System.out.println("No match found!");

}

}

}

**Scenarios of Switch case:**

**=================**

1) How can you group multiple cases in a switch statement to execute the same block of code?

(or)

WAP using Switch in java to check whether the given character is consonant or vowel.

class SwitchPractice{

public static void main(String[] args)

{

char ch = 'p';

switch(ch){

case 'a':

case 'A':

case 'e':

case 'E':

case 'i':

case 'I':

case 'o':

case 'O':

case 'u':

case 'U':System.out.println("It is a vowel");break;

default: System.out.println("It is a consonant");

}

}

}

Note:

====

1. case block in the switch always accept value only. (Not accept an expression).
2. Switch can be with both variable and expression.

2) Can you use a switch with string value?

Ans: Yes

class SwitchPractice{

public static void main(String[] args)

{

String day = "monday";

int dayNum = 0;

switch(day){

case "monday":

case "Monday":

case "MONDAY":

dayNum = 1;

break;

case "tuesday":

case "Tuesday":

case "TUESDAY":

dayNum = 2;

break;

default:

System.out.println("No Match Found!");

}

System.out.println("Day number = "+dayNum);

}

}

3) What happened when you forget to add a break in switch case-block?

(or)

What is fall-through condition?

4) How does switch statement behave when using char value?

5) Cn you define switch on a range of values?

Ans: No

Why:

range : 1 to 10

x ==> 1 to 10

x >= 1 and x <= 10

6) Can you use switch on null values?

class SwitchPractice{

public static void main(String[] args)

{

String a = null;

switch(a){

case "A":

System.out.println("Hi");

break;

default:

System.out.println("Variable has defined with null value");

}

}

}

Assignment:

=========

1) WAP using in java to check whether the given character is consonant or vowel (without switch).

2) Can we define nesting of switch?

**Day-19**

**18-02-2025**

**=============**

**null keyword:**

**=========**

-> null is a literal or pre-defined value

-> null can be used to assign for variables of type reference based like strings.

Note:

====

we cannot use the null for variables of primitive types.

Syntax:

String s = null;

Note:

=====

-> Keywords of Java can be classified into two types:

1) Pre-defined values/literals

===================

-> are three:

true

false

null

2) Pre-defined words

**Nesting of switch:**

**============**

-> A switch block inside the another switch block is called as "nesting of switch".

Logic-1:

=====

class NestingOfSwitch{

public static void main(String[] args){

int day = 1;

int month = 2;

switch(day){

case 1:

switch(month){

case 1:System.out.println("Jan 1st");break;

case 2:System.out.println("Feb 1st");break;

default:System.out.println("Invalid Month number");

}

break;

case 2:

switch(month){

case 1:System.out.println("Jan 1st");break;

case 2:System.out.println("Feb 1st");break;

default:System.out.println("Invalid Entry");

}

break;

default:System.out.println("Invalid day number");

}

}

}

Logic-2:

=======

class NestingOfSwitch{

public static void main(String[] args){

int day = 1;

int month = 2;

switch(day){

case 1:

case 2:

switch(month){

case 1:System.out.println("Jan 1st");break;

case 2:System.out.println("Feb 1st");break;

default:System.out.println("Invalid Entry");

}

break;

default:System.out.println("Invalid day number");

}

}

}

**Why break in switch?**

**==============**

-> to avoid the fall-through condition, we can use "break" in switch.

-> fall-through condition means:

executing the code from satisfied case block till the break or till end of switch.

**Scenario based questions on Conditional statements:**

**==================================**

1) In terms of performance, when we can use switch and when we can use if-else?

========================================

-> switch can be mostly used compared with if-else

because, the switch have a tendency to jump from one block of code to another block of code.

It can make escape from the sequential execution. But if-else cannot.

// username = "ADMIN"

// admin

//"admin".equals("ADMIN") ==> false

// "admin".equalsIgnoreCase("ADMIN") ==> true

import java.util.Scanner;

class LoginCheck{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

String username, password;

System.out.println("Enter Username:");

username = scan.nextLine();

System.out.println("Enter Password:");

password = scan.nextLine();

if("admin".equalsIgnoreCase(username) && "pass123".equalsIgnoreCase(password)){

System.out.println("Login Successful");

}

else{

System.out.println("Login Fail");

}

}

}

2) WAP IN JAVA THAT CHECKS IF THE USER NAME IS "admin" AND THE PASWORD IS "pass123". IF BOTH MATCH, PRINT "LOGIN SUCCESSFUL" OTHERWISE PRINT "INVALID CREDENTIALS".

3) WAP USING IF-ELSE TO CHECK IF A NUMBER IS EVEN OR ODD.

4) IMPLEMENT A JAVA PROGRAM USING IF-ELSE TO CHECK VOTER ELIGIBILITY.

5) WRITE A JAVA PROGRAM USING IF-ELSE-IF-ELSE LADDER TO ASSIGN GRADES BASED ON THE STUDENT'S PERCENTAGE.

PERCENTAGE >= 85 ==> 'A'

PERCENTAGE >= 70 BUT BELOW TO 85 ==> 'B'

PERCENTAGE >= 60 BUT BELOW TO 75 ==> 'C'

PERCENTAGE >= 50 BUT BELOW TO 60 ==> 'D'

PERCENTAGE >= 40 BUT BELOW TO 50 ==> 'E'

PERCENTAGE < 40 ==> 'F'

6) WRITE A JAVA PROGRAM USING SWITCH THAT PRINTS:

"stop" for red color

"get ready" for yellow

"go" for green

"invalid color" for other options.

7) WAP USING IF-ELSE TO CHECK IF A YEAR IS LEAP YEAR OR NOT.

import java.util.Scanner;

class TrafficSystem{

public static void main(String[] x)

{

Scanner s = new Scanner(System.in);

String color;

System.out.println("Enter the color:");

color = s.next();

switch(color){

case "red":

case "Red":

case "RED":

System.out.println("Stop!");

break;

case "yellow":

case "Yellow":

case "YELLOW":

System.out.println("Get Ready!");

break;

case "green":

case "Green":

case "GREEN":

System.out.println("Go!");

break;

default:

System.out.println("Invalid Color");

}

}

}

======================================

Solution-2:

===========

import java.util.Scanner;

class TrafficSystem{

public static void main(String[] x)

{

Scanner s = new Scanner(System.in);

String color;

System.out.println("Enter the color:");

color = s.next().toLowerCase();

switch(color){

case "red":

System.out.println("Stop!");

break;

case "yellow":

System.out.println("Get Ready!");

break;

case "green":

System.out.println("Go!");

break;

default:

System.out.println("Invalid Color");

}

}

}

8) WAP THAT TAKES THE ACCOUNT BALANCE AND WITHDRAWL AMOUNT AS INPUT. AND ENSURES:

I) WITHDRAWL AMOUNT IS MUST BE MULTIPLES OF 100 ONLY.

II) BALANCE IS SUFFICIENT

III) IF VALID, DEDUCT AMOUNT FROM BALANACE AND MAKE IT DISPLAY THE REMAINING BALANCE.

IV) IF INVALID, DISPLAY THE MESSAGE "INSUFFICIENT FUNDS SO TRANSACTION FAILED."

import java.util.Scanner;

class BankingSystem{

public static void main(String[] x)

{

Scanner s = new Scanner(System.in);

double accountBalance, amount, balance;

System.out.println("Enter The Account Balance:");

accountBalance = s.nextDouble();

System.out.println("Enter The Withdraw Amount:");

amount = s.nextDouble();

if(amount % 100 == 0){

System.out.println("Withdraw is in processing...");

if(amount <= accountBalance){

System.out.println("Balance is Sufficient..");

balance = accountBalance - amount;

System.out.println("Remaining Balance = "+balance);

}

else{

System.out.println("Insufficient funds!");

}

}

else{

System.out.println("Withdraw amount should be multiple of 100.");

}

}

}

=====================================

/\*

A SHOPPING WEBSITE OFFERS A DISCOUNT BASED ON YOUR PURCHASE AMOUNT.

PURCHASE >= 5000 ==> DISCOUNT IS 20%

PURCHSE >= 2000 ==> DISCOUNT IS 10%

PURCHASE >= 1000 ==> DISCOUNT IS 5%

PURCHSE < 1000 ==> 0% DISCOUNT

CALCULATE THE TOTAL BILL AFTER DISCOUNT ON YOUR PURCHASE.

\*/

import java.util.Scanner;

class CartSystem{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);

double purchase, bill;

System.out.println("Enter the cost of purchase:");

purchase = s.nextDouble();

if(purchase >= 5000){

bill = purchase - (purchase \* 20/100);

}

else if(purchase >= 2000){

bill = purchase - (purchase \* 10/100);

}

else if(purchase >= 1000){

bill = purchase - (purchase \* 5/100);

}

else{

bill = purchase;

}

System.out.println("The Final bill = "+bill);

}

}