Conditional Statements in Java

Learning Scope

Introduction, Normal flow of control, Conditional flow of control: if, if and only if, nested if, if else, if else if ladder, System.exit (0) - to terminate the program; Multiple branching of control: switch case, fall through in switch case, Menu driven programs.

INTRODUCTION

Decision making is an important aspect of our life. It guides us to do our work in the right direction based on available situations. It also helps us in choosing a specific action to be taken depending on a given condition.

Let us take a real life example of decision making:

If Age > 19, then "You are selected to the Senior Team" else "You are selected to the Junior Team".

In the statement shown above either of the two options "You are selected to the Senior Team" or "You are selected to the Junior Team" will be considered, based on the given condition "Age > 19". If the condition is true, then you are selected to the senior team, otherwise you are selected to the junior team.

Similarly in computer programming also, we need to carry one action out of two or more actions based on a given condition.

In this chapter, we are going to discuss decision making statements in which the flow of control plays a vital role in the condition based operations.

As soon as the command is given to execute a program, the control reaches the first statement of the program. It keeps executing the statements sequentially until the end of the program is reached. Such movement of control is termed as *'flow of control'*.

The flow of control during the execution of a program takes place in the following ways:

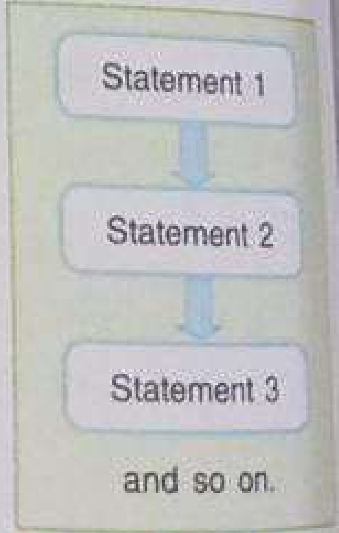
(*a*) Normal flow of control

(*b*) Conditional flow of control

(*c*) Multiple branching of control

Normal Flow of Control

It is the normal procedure where the control keeps on executing each and every statement of the program with the top down approach. As soon as one statement has been executed, the control moves to the next line for further

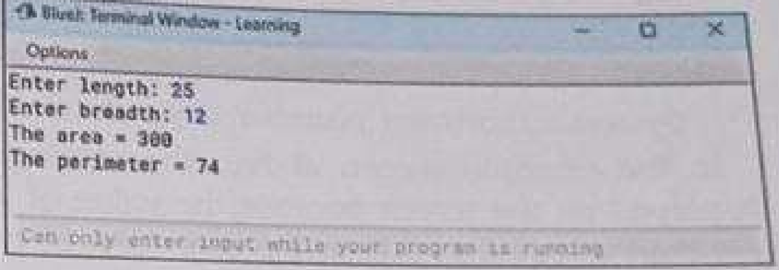


Normal flow of control

processing. At the end, it returns to the system after displaying the result of the program.

A program is illustrated to show the normal flow of control:

```
// To find the area and perimeter of a rectangle
import java.util.*;
public class Rectangle
{
public static void main(String args[ ])
  {
  Scanner in = new Scanner(System.in);
  int l,b,p,ar;
  System.out.print("Enter length:");
  l=in.nextInt( );
  System.out.print("Enter breadth:");
  b=in.nextInt( );
  ar=l*b;
  p=2*(l+b);
  System.out.println("The area = "+ar);
  System.out.println("The perimeter = "+p);
  }
}
```



```
Enter length: 25
Enter breadth: 12
The area = 300
The perimeter = 74
```

```
Can only enter input while your program is running
```

Here, you can see that the control keeps on executing each and every statement of the program in a top down approach. After compilation, it takes the values of the length and breadth from the user and then displays the result.

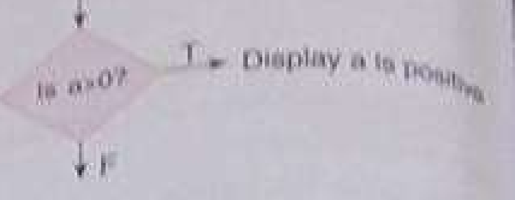
But, this normal flow of control may not be helpful if the complexity increases. In some cases, you may need to transfer the control to a defined statement by leaving some statements unexecuted. Under such circumstances, conditional flow of control is created to resolve the problems.

Conditional flow of Control

Sometimes, it may happen that you want to operate a block of statements when the given condition holds true. In case the given condition is false, the execution of the current block should be ignored and the control should move to execute another block. In this situation, the control executes a specific block of statements based on a given condition which is known as conditional flow of control.

The conditional flow of control can be achieved by using 'if' statement in the following ways:

- if statement
- if and only if statement
- if-else statement
- if-else-if statement
- nested if statement



if statement

This statement is used when a statement or a block of statements to be executed based on a given condition, holds true. In case, the condition is false, it ignores the block and moves forward with the normal flow of control.

Construct of if statement

```
if(condition)
  Statement
    or,
if(condition)
 {
  Statement 1
  Statement 2
  :
  :
  Statement n
 }
```

With reference to this syntax, 'if' statement is used to check a condition. If the condition is true then the immediate next statement is executed. In case the condition is false, the control ignores execution of the next statement and moves to execute other statements of the program.

A situation may arise when you need to execute more than one statement/s based on a condition. In such a case, all the statements must be enclosed within curly brackets { } inside the if block.

For example,

Single statement using if

```
int a=10;
if(a>0)
   System.out.println("Number is positive");
```

In the example shown above, the message "Number is positive" will be displayed on the screen because the value of 'a' is more than 0. If the value is less than or equal to 0 then the execution of print statement will be ignored.

Multiple statements using if

```
int x=9,p=0; double t=0.0D;
if(x % 2 != 0)
 {
 p=x*x;
 t=Math.sqrt(x);
 }
System.out.println("Square of number"+p+" and the Square root"+t);
```

Compound Statement

A statement that includes one or more statements within it marked under curly brackets { }, is said to be a **compound statement**.

With reference to the above example, the if statement checks whether the value of x=9 is odd or not. Here, the condition is true. Hence, it enters the block of statements enclosed within curly brackets { }, finds square and square root of the number and displays the result as:

```
Square of the number 81 and the Square root 3.0
```

If you give value of x as an even number, the condition will become false and the associated block of statements will be ignored. The control will directly use the print statement and display as:

Square of the number 0 and the Square root 0.0

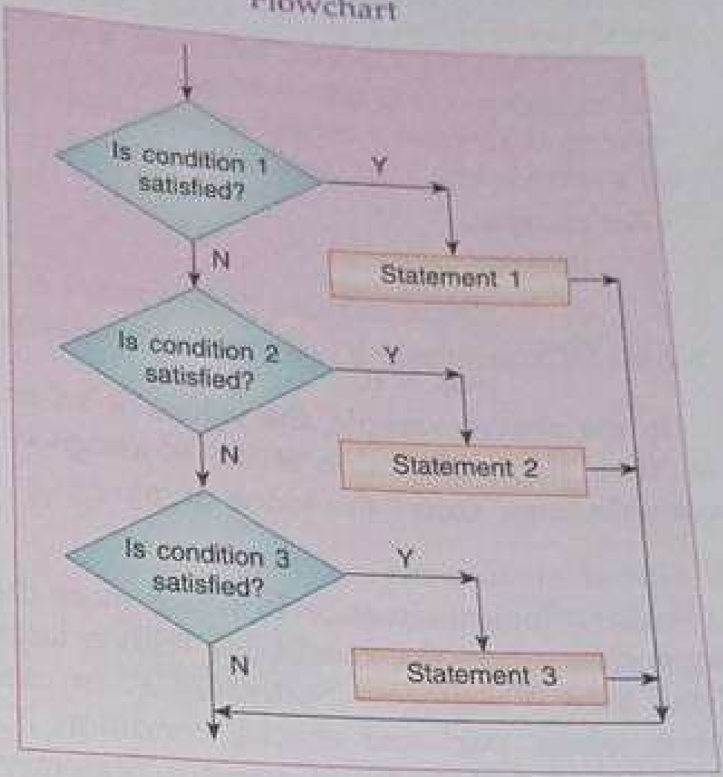
If and only if statements (Use of multiple if)

It is a logical situation in which multiple statements can be operated selectively. If condition 1 is true then control will execute statement 1. In case it is false, the control moves to check condition 2 rather than executing any statement. Further statement 2 or statement 3 are operated based on the true / false status of condition 2 or condition 3 respectively.

Construct of multiple if

```
if (condition 1)
        Statement 1
if (condition 2)
        Statement 2
if (condition 3)
        Statement 3
```

Flowchart



Is condition 1 satisfied?
Y
N
Statement 1
Is condition 2 satisfied?
Y
N
Statement 2
Is condition 3 satisfied?
Y
N
Statement 3

For example:

```
if(a<0)
   System.out.println("A negative number");
if(a>0)
   System.out.println("A positive number");
if(a= =0)
   System.out.println("The number is zero");
```

In the above example, first of all it is checked whether the value of the variable (say, a) is less than zero or not. If it is true, then it displays the message "A negative number". In case the first condition is false, the control enters the next 'if' part and checks whether the value is greater than zero or not. If it is true, the system displays the message "A positive Number". In the same way, the control keeps on verifying other conditions.

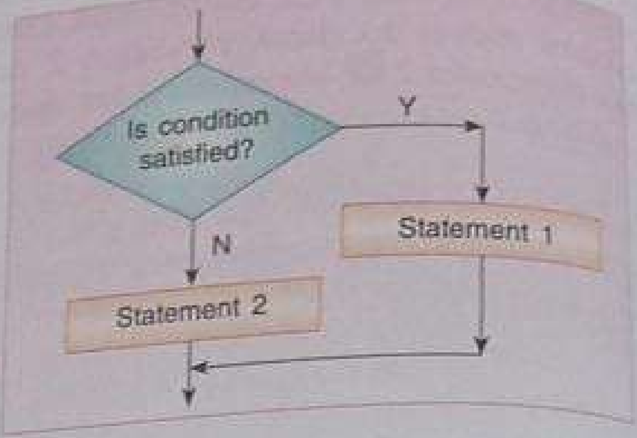
if-else statement

It is a logical situation in which either of the two actions is to be performed depending upon a specified condition. When the condition is 'True', it performs one set of statements, otherwise it performs another set of statements.

Construct of if-else

```
if(condition)
        Statement 1
else
        Statement 2
```

Flowchart



For example,

```
    if(m>n)
            max = m;
    else
            max = n;
```

In the above example, the values of 'm' and 'n' are compared. If the condition is 'True' then, the value of 'm' is assigned to a variable 'max', otherwise the variable 'max' stores the value of 'n'.

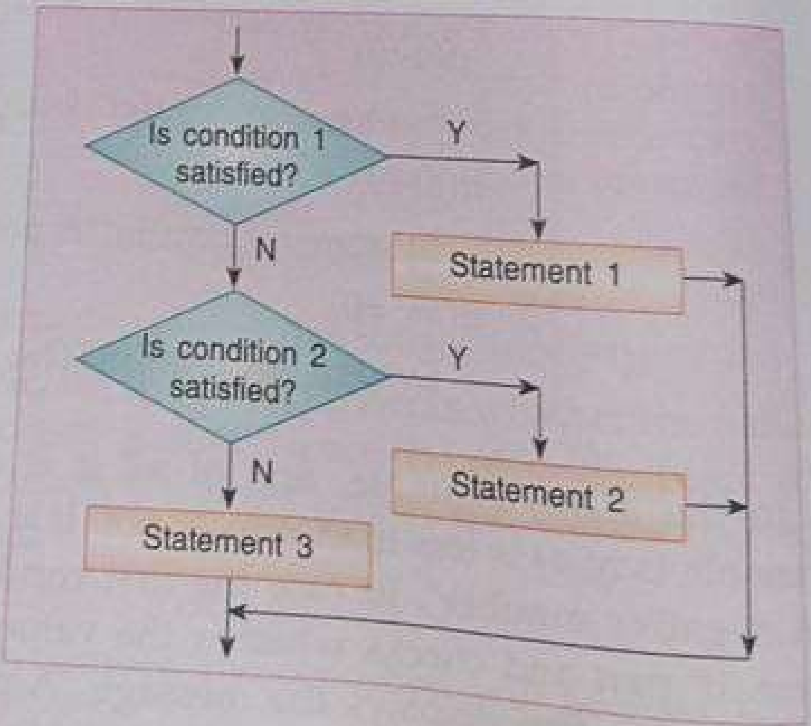
if-else-if statement

It is also a logical situation in which more than two actions are to be performed, depending upon the conditions with a keyword 'elseif'. When 'condition 1' is 'True' then it performs 'Statement 1'. In case 'condition 1' is false, the control enters 'elseif' part and checks 'condition 2'. If it is true, the control executes 'Statement 2', otherwise it executes 'Statement 3'.

Construct of if-else-if:

```
if(condition 1)
        Statement 1
else if(condition 2)
        Statement 2
else
        Statement 3
```

Flowchart



For example,

```
if(a>b)
    System.out.println("First number is greater");
else if(b>a)
    System.out.println("Second number is greater");
else
    System.out.println("Both the numbers are equal");
```

In the above example, first of all it checks whether the first number is greater than the second number or not. If it is true, then it displays the message "First number is greater". In case the condition is false, the control enters 'else if' part and checks whether the second number is greater than the first number or not. If it is true, then the system displays the message "Second number is greater" otherwise, the control automatically enters the next 'else' part and displays the message "Both the numbers are equal".

nested if statement

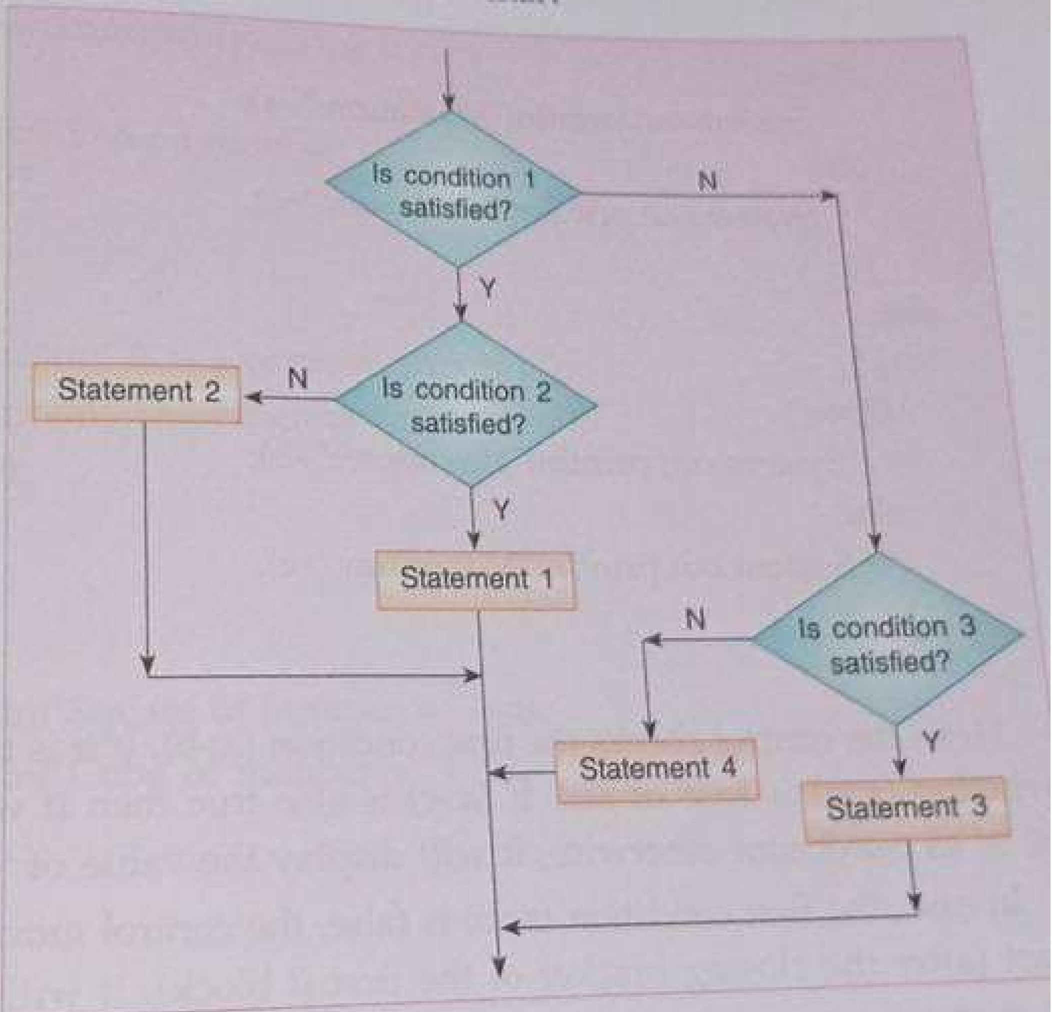
The term 'nested' means one action is taken within another action.

Thus, nested if statement is a statement where an 'if' statement is placed within another 'if' statement.

Construct of nested if

```
if(condition 1)
 {
  if (condition 2)
   Statement 1
  ---------------
  else
   Statement 2
  ---------------
 }
else
 {
  if(condition 3)
   Statement 3
  ---------------
  else
   Statement 4
  ---------------
 }
```

Flowchart



Is condition 1 satisfied?
N
Y
Statement 2
N
Is condition 2 satisfied?
Y
Statement 1
N
Is condition 3 satisfied?
Y
Statement 4
Statement 3

For example,

```
if(x>10)
   {
   if (x<100)
      System.out.println("2-Digit number");
   else
      System.out.println("Not a 2-Digit number");
   }
else
      System.out.println("1-Digit number");
```

With reference to the above snippet, if the condition (x>10) is true then it will enter the block and check another condition (x <100). If this condition is true, then it will display "2-Digit number" otherwise, "Not a 2-Digit number". In case, the first condition (x>10) is false then the control will ignore checking the internal condition and will move to the else part (after the closing brace). As a result "1-Digit number" will be displayed on the screen.

Let us take another example;

```
if(a>b)
   {
   if(a>c)
      System.out.println("Maximum:"+a);
   else
      System.out.println("Maximum:"+c);
   }
else
   {
   if(b>c)
      System.out.println("Maximum:"+b);
   else
      System.out.println("Maximum:"+c);
   }
```

Here, the control checks the first condition (a>b). If it is true then it will check whether (a>c) is true or not. If (a>c) is also true then it will display the value of 'a' as maximum otherwise, it will display the value of 'c' as maximum.

In case, the first condition (a>b) is false, the control moves to operate the else part (after the closing bracket of the first if block). It will display the value of 'b' as maximum or the value of 'c' as maximum depending upon the condition whether it is 'true' or 'false' respectively.

SYSTEM.EXIT(0)

Sometimes, it may happen that the execution of the whole program is not over but your requirement is fulfilled. Under such situation, you would like to terminate the execution in the middle of the program. 'System.exit(0)' function will help you to carry out this task.

During the course of running the program, as soon as 'System.exit(0)' function is invoked, it will terminate the execution at that moment. Thus, the execution of remaining statements of the program will be ignored.

Syntax:

```
System.exit(0);
```

Suppose, you want to find the square and cube of a positive number. If the number is positive, it should execute and display the result. But for a negative number, the program must terminate its execution. It can be illustrated as:

```
// A program to display the square and cube of a positive number
import java.util.*;
public class Positive
{
public static void main(String args[ ])
 {
  Scanner in = new Scanner(System.in);
  int n,sq,cb;
  System.out.println("Enter a number:");
  n = in.nextInt( );
  if(n < 0)
    {
    System.out.println("It is a negative number.");
    System.out.println("The program terminates.");
    System.exit(0);
    }
  sq = n * n;
  cb = n * n * n;
  System.out.println("Square of number = "+sq);
  System.out.println("Cube of number = "+cb);
 }
}
```

```
        System.out.print("Enter a number: ");
        n=in.nextInt( );
        d1=n/10;
        d2=n%10;
        sum=d1+d2;
        p=d1*d2;
        if((sum+p)==n)
            System.out.println(n+" is a Special 2-digit number");
        else
            System.out.println("Not a Special 2-digit number");
    }
}
```

Prog. 3 Write a program to enter three sides of a triangle and check whether the triangle is possible or not. If possible then display whether it is an Equilateral, an Isosceles or a Scalene Triangle, otherwise display the message "Triangle is not possible".

```
// A program to display the type of triangle
import java.util.*;
public class Triangle
{
    public static void main (String args[ ])
    {
        Scanner in = new Scanner(System.in);
        int a,b,c;
        System.out.print("Enter second side of a triangle: ");
        a=in.nextInt( );
        System.out.print("Enter third side of a triangle: ");
        b=in.nextInt( );
        System.out.print("Enter first side of a triangle: ");
        c=in.nextInt( );
        if((a+b>c) &&(b+c>a) && (c+a>b))
        {
            System.out.println("Triangle is possible");
            if((a==b)&&(b==c)&&(c==a))
                System.out.println("Equilateral triangle");
            else if((a==b)||(b==c)||(c==a))
                System.out.println("Isosceles triangle");
            else if((a!=b)&&(b!=c)&&(c!=a))
                System.out.println("Scalene triangle");
        }
        else
            System.out.println("Triangle not possible");
    }
```

Prog. 4 Write a program to input three unequal numbers and display the second smallest number.
Sample Input: 65, 41, 98
Sample Output: 65

```
// A program to display the second smallest number
import java.util.*;
public class Smallest
{
public static void main(String args[ ])
{
 Scanner in = new Scanner(System.in);
 int a,b,c;
 System.out.print("Enter first number: ");
 a=in.nextInt( );
 System.out.print("Enter second number: ");
 b=in.nextInt( );
 System.out.print("Enter third number: ");
 c=in.nextInt( );
 if((a<b)&&(a<c))
 {
   if(b<c)
      System.out.println("The second smallest number: "+b);
   else
      System.out.println("The second smallest number: "+c);
 }
 if((b<c)&&(b<a))
 {
   if(c<a)
      System.out.println("The second smallest number: "+c);
   else
      System.out.println("The second smallest number: "+a);
 }
 if((c<a)&&(c<b))
 {
   if(a<b)
      System.out.println("The second smallest number: "+a);
   else
      System.out.println("The second smallest number: "+b);
 }
}
}
```

Prog. 5 Write a program to enter three numbers and a character. Find and display the sum of the numbers, if the given character is 's' and product of the numbers if the given character is 'p'. The program displays a message "Invalid Character", if the user enters a letter other than 's' or 'p'.

```
// A program to display the sum and product of the numbers
import java.util.*;
public class Numbers
{
   public static void main (String args[ ])
   {
      Scanner in = new Scanner(System.in);
      int a, b, c, sum=0, pr=1;
      char ch;
      System.out.print("Enter first number: ");
      a=in.nextInt( );
      System.out.print("Enter second number: ");
      b=in.nextInt( );
      System.out.print("Enter third number: ");
      c=in.nextInt( );
      System.out.print("Enter 's' for sum and 'p' for product of three numbers: ");
      ch=in.next( ).charAt(0);
      if(ch=='s')
      {
         sum=a+b+c;
         System.out.println("The sum of three numbers: "+sum);
      }
      else if (ch=='p')
      {
         pr=a*b*c;
         System.out.println("The product of three numbers:"+pr);
      }
      else
         System.out.println("Entered an invalid character!!!");
   }
}
```

Prog. 6 In an examination, the grades are awarded to the students in 'Science' according to the average marks obtained in the examination.

Marks	Grades
80% and above	Distinction
60% or more but less than 80%	First Division
45% or more but less than 60%	Second Division
40% or more but less than 45%	Pass
Less than 40%	Promotion not granted

Write a program to input name and marks in Physics, Chemistry and Biology. Calculate the average marks. Display the name, average marks and the grade obtained.

```
// A program to display the grades awarded
import java.util.*;
public class Grades
{
 public static void main(String args[ ])
 {
  Scanner in = new Scanner(System.in);
  int p, c, b;
  float avg = 0;
  String name, gr ="";
  System.out.print("Enter name: ");
  name= in.nextLine( );
  System.out.print("Enter marks in Physics: ");
  p= in.nextInt( );
  System.out.print("Enter marks in Chemistry: ");
  c= in.nextInt( );
  System.out.print("Enter marks in Biology: ");
  b= in.nextInt( );
  avg=(p+c+b)/3;
  if(avg>=80)
     gr="Distinction";
  if(avg>=60 && avg<80)
     gr="First Division";
  if(avg>=45 && avg<60)
     gr="Second Division";
  if(avg>=40 && avg<45)
     gr = "Pass";
  if(avg<40)
     gr = "Promotion not granted";
  System.out.println("Name : "+name);
  System.out.println("Average Marks :" +avg);
  System.out.println("Grade : "+gr);
 }
}
```

Prog. 7 The State Electricity Board calculates the electricity bill for their consumers according to the units consumed (per month) as per the given tariff.

Units Consumed	Charges
Up to 100 units	₹ 4.80/unit
More than 100 units and up to 300 units	₹ 5.30/unit
More than 300 units and up to 500 units	₹ 6.80/unit
More than 500 units	₹ 7.50/unit

Write a program to input name of the consumer, consumer number, month and units consumed. Calculate and display the electricity bill with all the details.

```
// A program to display the electricity bill
import java.util.*;
public class Electricity
{
    public static void main(String args[ ])
    {
        Scanner in = new Scanner(System.in);
        String name,mt;
        int cn,u;
        double p=0.0;
        System.out.print("Enter consumer name: ");
        name=in.nextLine( );
        System.out.print("Enter consumer number: ");
        cn=in.nextInt( );
        System.out.print("Enter month: ");
        mt=in.next( );
        System.out.print("Enter units consumed: ");
        u=in.nextInt( );
        if(u<=100)
            p=u*4.80;
        if(u>100 && u<=300)
            p=u*5.30;
        if(u>300 && u<=500)
            p=u*6.80;
        if(u>500)
            p=u*7.50;
        System.out.println("Consumer name: "+name);
        System.out.println("Consumer number: "+cn);
        System.out.println("Electricity bill for the month: "+mt);
        System.out.println("Units consumed: "+u);
        System.out.println("Amount to be paid: "+p);
    }
}
```

Prog. 8 Ashish Kunal randomly enters a number on his computer. Write a program to check whether the number entered by him is a one digit number, a two digit number or a three digit number. Now, perform these tasks:

- If it is a one digit number then display its square.
- If it is a two digit number then display its square root.
- If it is a three digit number then display its cube root.

Otherwise, display the message "The number entered is more than three digits".

```
// A program to check the number of digits of a number import java.util.*;
public class Digits
{
public static void main(String args[ ])
{
Scanner in = new Scanner(System.in);
int n;
System.out.print("Enter a number: ");
n= in.nextInt( );
if(n>=0 && n<10)
  {
  System.out.println("A one digit number");
  System.out.println("Square of the number: "+(n*n));
  }
if(n>=10 && n<100)
  {
  System.out.println("A two digit number");
  System.out.println("Square root of the number: "+Math.sqrt(n));
  }
if(n>=100 && n<1000)
  {
  System.out.println("A three digit number");
  System.out.println("Cube root of the number: "+Math.cbrt(n));
  }
if(n>=1000)
  System.out.println("The number entered contains four digits or more")
}
}
```

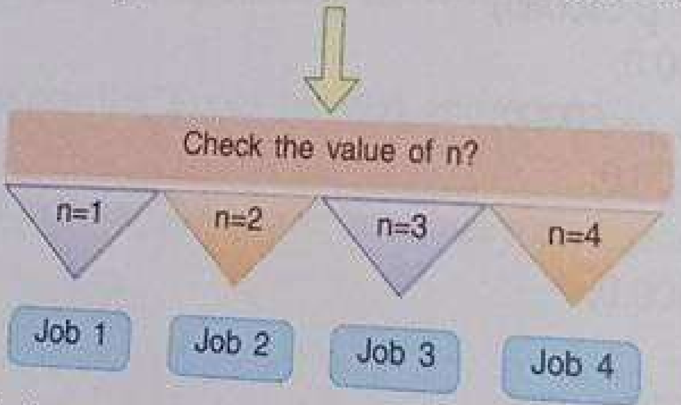
```
      System.out.println("The discount: ₹"+d);
      System.out.println("The tax: ₹"+tax);
      System.out.println("The amount to be paid: ₹"+np);
      }
      if(k==2)
      {
      System.out.print("Enter the amount of purchase: ");
      p=in.nextInt( );
      if(p<=20000)
          d=p*2.5/100.0;
      if(p>20000 && p<40000)
          d=p*5.0/100.0;
      if(p>40000 && p<60000)
          d=p*7.0/100.0;
      if(p>60000 )
          d=p*8.5/100.0;
      rp=p-d;
      tax=rp*12.5/100.0;
      np=rp+tax;
      System.out.println("Customer's name: "+name);
      System.out.println("The price of the product: ₹"+p);
      System.out.println("The discount: ₹"+d);
      System.out.println("The tax: ₹"+tax);
      System.out.println("The amount to be paid: ₹"+np);
      }
   }
}
```

MULTIPLE BRANCHING OF CONTROL

(SWITCH CASE STATEMENT/MENU DRIVEN PROGRAM/USER'S CHOICE)

You have already learnt the application of all types of conditional statements. These statements are basically used to guide the control to execute either of two block of statements.

Now, imagine a situation where you want to carry out a specific task out of a number of choices for a given condition. The situation is shown as:



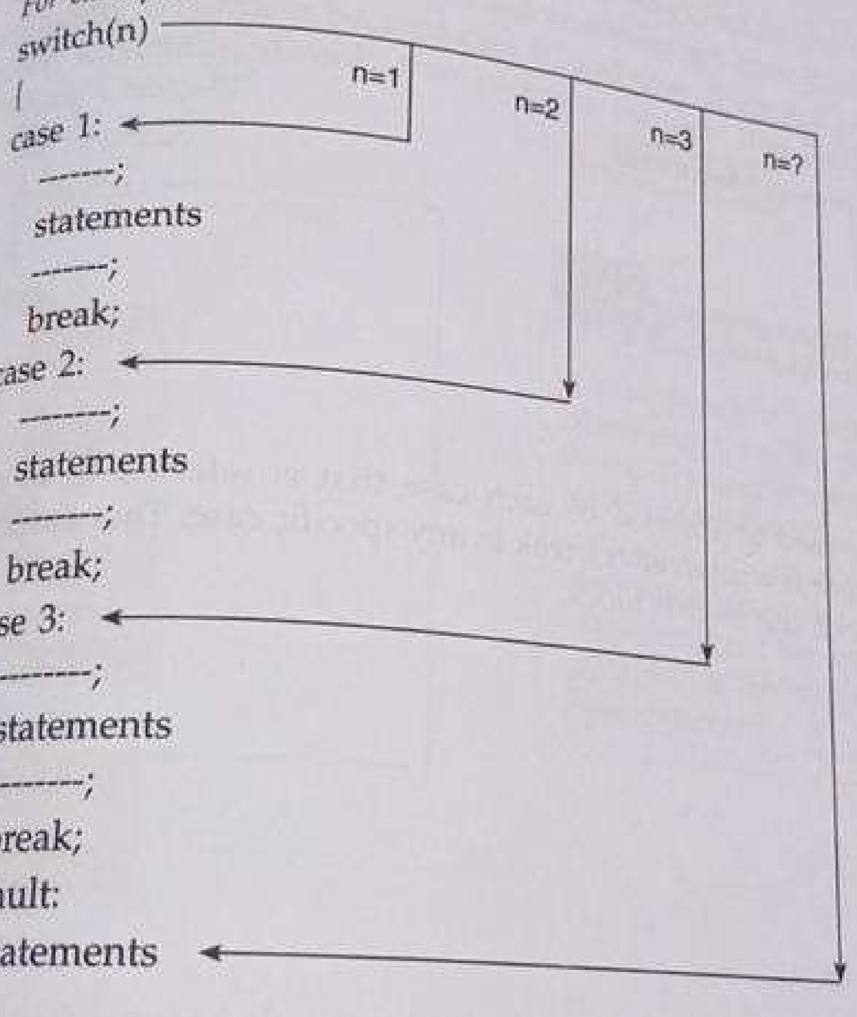
A number of 'if-else' constructs may be required to deal with the above situation. Hence, a number of 'if-else' statements can be replaced with only one statement, that is, the 'Switch Case' statement.

Thus, the Switch Case statement is a multiple branching statement where a particular block of statements is executed out of a number of blocks as per the user's choice. In this system, the control goes to a specific case and executes the statements for the given switch value. It also includes a 'default' case. The default case is executed only when the switch value doesn't match any of the given cases.

Syntax:

```
switch(control variable)
{
case 1:
   Block 1
case 2:
   Block 2
case 3:
   Block 3
default :
   Block 4
}
```

For example,



```
switch(n)
{
case 1:
   -------;
   statements
   -------;
   break;
case 2:
   -------;
   statements
   -------;
   break;
case 3:
   -------;
   statements
   -------;
   break;
default:
   statements
}
```

With reference to the above structure, a number of cases are taken care of by a single switch statement. The control decides the execution of a specific case based on the value of the control variable (say, n). If the value of the control variable (n) is 1, the control goes to execute the statements in case 1. The statements