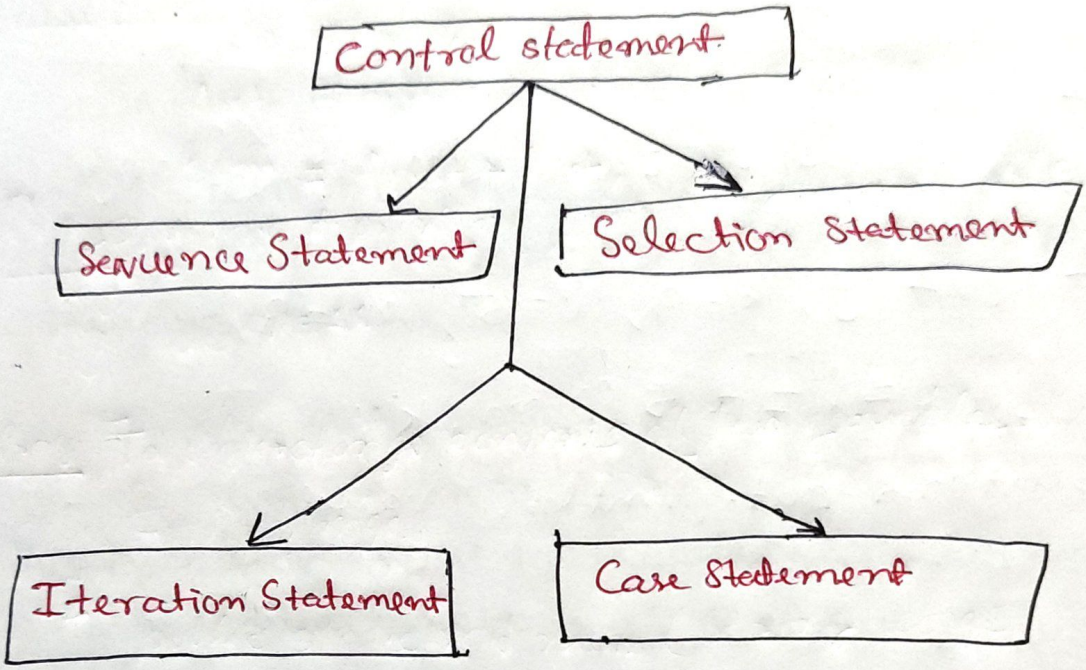
Control Statements

"Statements that control the flow of execution in a program are called control statement"



Control Statement

Sequence Control Statement-

It means various steps in a program are executed sequentially i.e in the same order in which they appear in the program. By default statements in a program are executed sequentially. You need not do anything to achieve this concept.

Simple Interest

Prog
```
class Simple
{
    public static void main (String args[])
{
```

```
int p, t;
double r, si;
    p = 2000;
    r = 2.5;
    t = 5;
    si = (p*r*t)/100;
    System.out.println("Simple interest = Rs" + si);
}
}.
```

Prog to calculate gross salary.

```
Class Gross
{
  public static void main(String args[])
  {
    double bs, da, hra, gs;
      bs = 2000;
      da = (40/100) * bs;
      hra = (20/100) * bs;
      gs = bs + da + hra;
    System.out.println("Gross Salary = Rs" + gs);
  }
}.
```

Prog to calculate cost price of an item

```
Class Cost
{
  public static void main(String args[])
  {
    double cp, sp, pr;
      sp = 4000;
      pr = 500;
      cp = sp - pr;
    System.out.println("CP =" + cp);
  }
}.
```

Prog to calculate sum of first and last digit of four digit no.

```
Class Sum
{
  public static void main (String args[])
  {
    int num, first, last, sum;
      num = 4257;
      first = num/1000;
      last = num % 10;
      sum = first + last;
    System.out.println("sum of first and last digit=" +sum);
  }
}
```

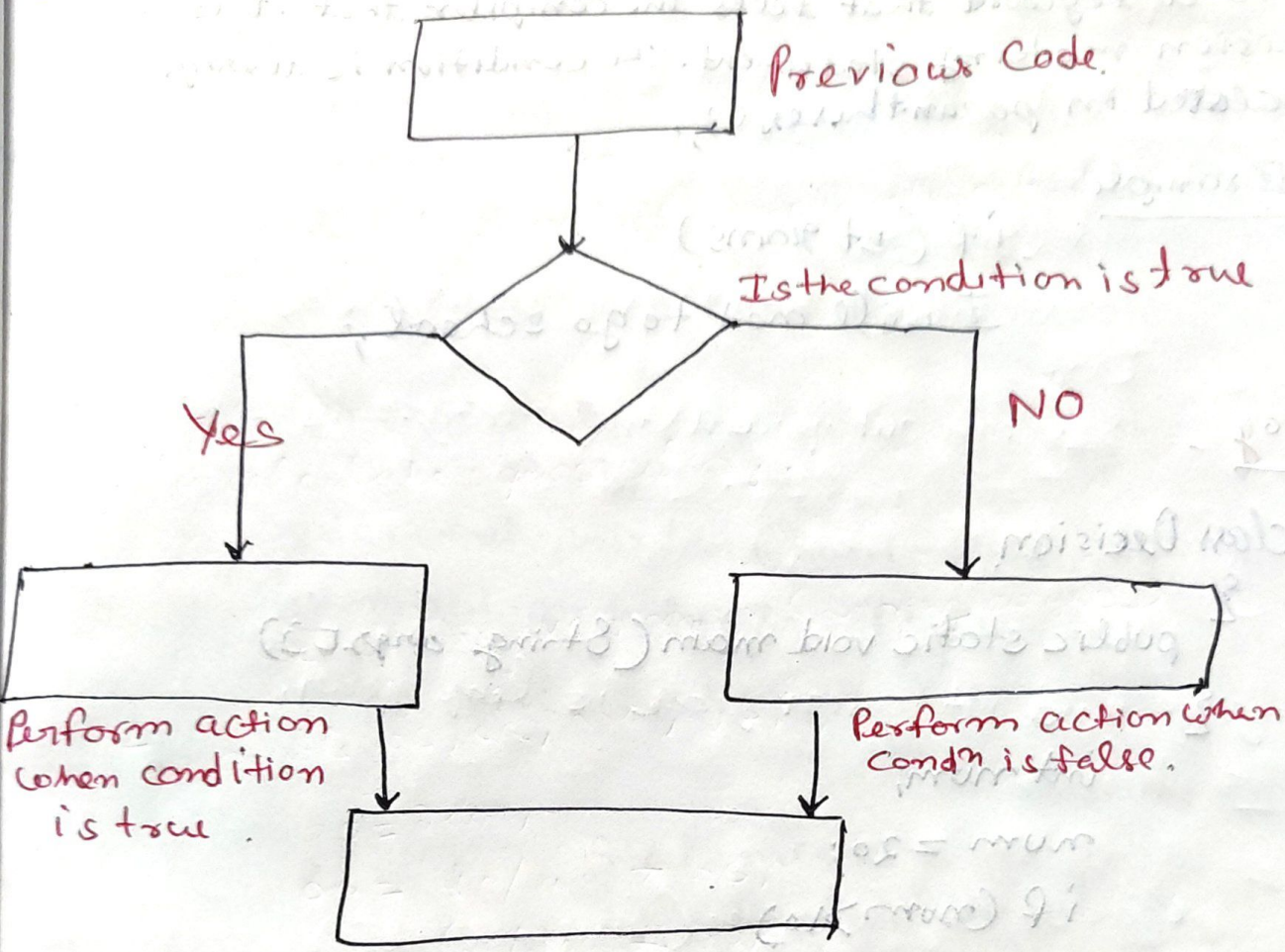
op = 11

Prog To interchange contents of two variables.

```
Class Interchange.
{
  public static void main (String args[])
  {  int c, d, temp;
       c = 57;
       d = 45;
    System.out.println(" Before Interchang");
    System.out.println(" c=" +c + " d=" +d);
      temp = c;
       c = d;
       d = temp;
    System.out.println(" After interchange");
    System.out.println(" c=" +c + " d=" +d);
  }
}.
```

Decision Control Statement

Statement that control execution of program based upon some condition are decision control statement.



There are two types of decision control statement.

> if statement

> Switch Case.

if Statement

if statement is used to implement decision control statement. When this statement is encountered. Compiler automatically gets the instruction that follows is a decision control statement.

In if statement we test a condition and if the condn is true then the statement followed by if is evaluated. If the condition is false. then the compiler skip that part of the code

General form

```
if (condition)
        statement;
```

if is a keyword that tells the compiler that it is a decision making statement. The condition is always enclosed in parantheses ( ).

Example

```
if (it rains)
        I will not to go school;
```

Prog.

```
class Decision
{
  public static void main (String args[])
  {
    int num;
    num = 20;
    if (num>10)
System.out.println("Number is greater than 10");
  }
}
```

If with multiple statements

General form

```
if (condition)
{
    statement 1;
    Statement 2;
}
```

```
class Decision
{
  public static void main(String args[])
  {
    int a, b;
    int sum, mul;
    a = 8;
    b = 2;
    if (a>b)
      {
        sum = a+b;
        mul = a*b;
      System.out.println("Sum = " + sum);
      System.out.println("multiplication = " + mul);
    }
  }
```

If ___ else ___ Statement

The if statement can execute a single statement or multiple statements but that too only when the condition is true. You know it does nothing when condn is false. Sometimes you want to execute a set of statements even when condition is false. That task can be accomplished by using if ___ else statements

General form

```
if (condition)
    statement 1;
else
    statement 2;
```

Prog

```
class Decision
{
  public static void main (String args[])
{
    int yos;
    yos = 10;
    if (yos >3)
    System.out.println("salary = Rs 35000");
    else
    System.out.println("salary = Rs 10000");
}
}
```

Prog to find out given year is leap year,

```
class Decision
{
  public static void main (String args[])
{
    int year, rem;
    year = 2002;
    rem = year%4;
    if (rem==0)
    System.out.println("Given year is leap year
    else
    System.out.println("Given year is not leap
                                        year");
}
}.
```

Prog that will find out given no is even or odd.

```
class Decision
{
  public static void main(String args[])
  {
    int num, rem;
    num = 6;
    rem = num%2;
    if(rem == 0)
    System.out.println("Given no is even");
    else
    System.out.println("Given no is odd");
  }
}.
```

Nested if ---- else --- statement.

When we write entire if --- else block either inside the body of if statement or inside the body of else ~~statem~~ statement then it is called nested if --- else.

General form

```
if (condition)
   statement 1;
else
   {
     if(condition)
        statement 1;
     else
        statement 2;
   }.
```

Or

```
if (condition)
  {
    if (condition)
        Statement 1;
    else
        Statement 2;
  }
  else
  {
    Statement;
    Statement;
  }
```

Prog

```
class Nest
  {
    public static void main (String args[])
  {
      int a=5, b=6, c=2;
          d=
      if ((a>b)&&(a>c))
        {
          System.out.println("a is greate");
        else
          {if ((b>a)&&(b>c))
            System.out.println("b is greater")
          else
            System.out.println("c is greater")
          }
  } }
```

else if Ladder

We can also write program in one more way. This is based upon the sequence of nested ifs.

General form

```
if(condition)
    statement1;
else if (condition)
    statement;
else if (condition)
    statement;
else if (condition)
    statement;
else
    statement;
```

Prog

```
Class Marks
{ public static void main (String args[])
{ int m1, m2, m3, m4, m5;
  double total, per;
m1 = 79, m2 = 90, m3 = 89, m4 = 98, m5 = 88;
total = m1+m2+m3+m4+m5;
per = (total/500) *100;
if (per >= 60)
    System.out.println("First Division");
else if (per >= 50)
    System.out.println("Second Division");
else if (per >= 40)
    System.out.println("Third Division");
else
    System.out.println("Fail");
}
}
```

Use of logical operators

They are more efficient to implement as they can check more than one condition in one pair of parantheses.

Program

```
class Greater
{
  public static void main(String args[])
  {
      int a=40;
    if ((a>40) && (a<45))
    System.out.println("a is greater the 40
                        but less than 45");
    else
    System.out.println("a=" + a);
  }
}
```

O/P = 40

Prog

```
class Age
{
  public static void main (String args[])
  {
      int ramesh, shyam, suresh;
        ramesh = 23;
        shyam = 25;
        suresh = 34;
    if ((ramesh < shyam) && (ramesh < suresh
    system.out.println ("Ramesh is youngest")
```

```
if ((Shyam < ramesh) && (Shyam < suresh))
    System.out.println("Shyam is youngest");
if ((Suresh < Ramesh) && (Suresh < shyam))
    System.out.println("Shyam is youngest");
}
}
```

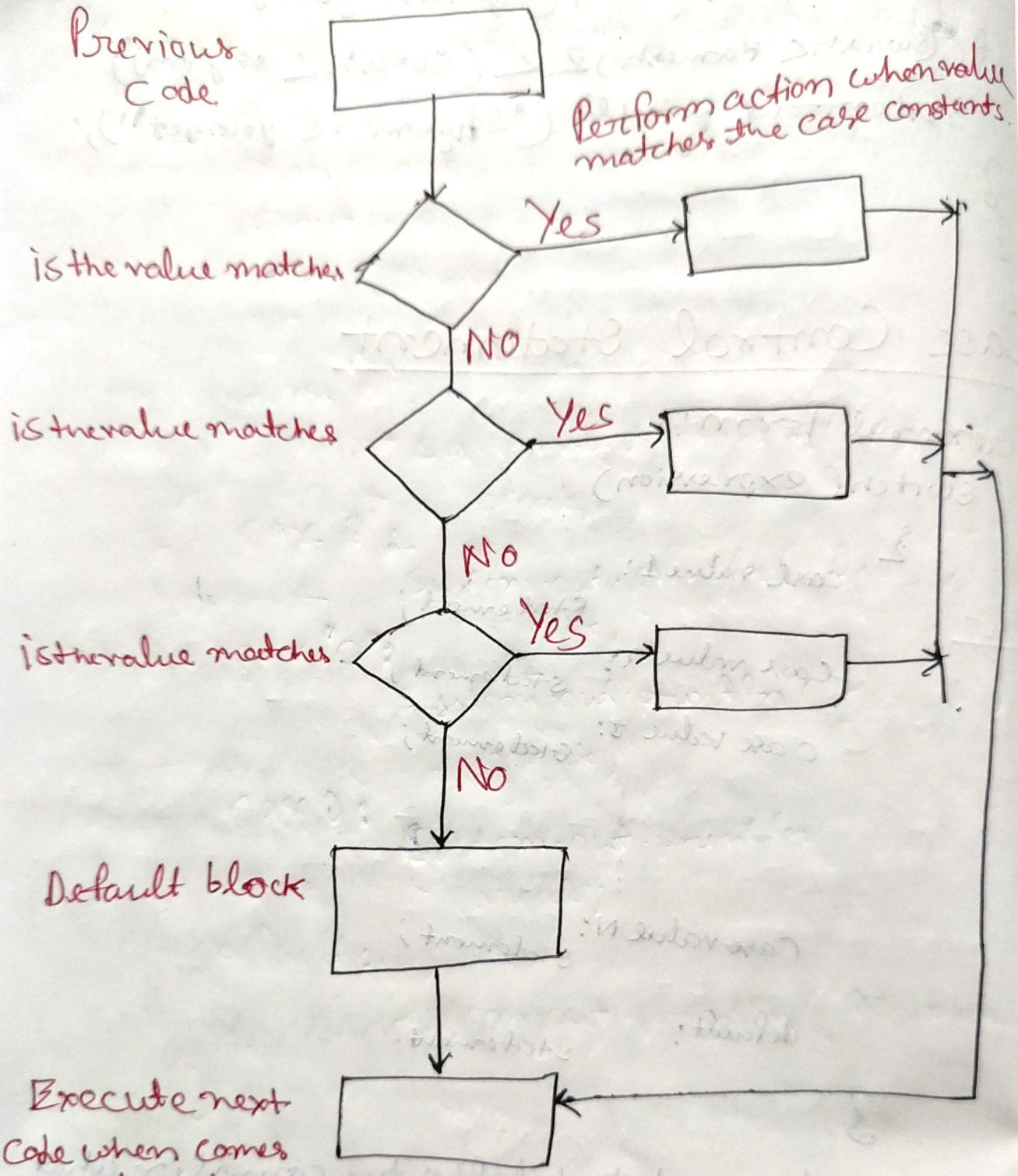
Case Control Statement

General form:

```
switch(expression)
{
    case value 1:
                    Statement;
    case value 2:   statement;
    case value 3:   Statement;
        |
        |
    Case value N:   statement;
    default:        statement.
}.
```

Switch is keyword that tells the compiler what switch case proceeds. expression is any value of type byte, short, int or char. Case is again keyword that specifies that following choices are provided. Value of each case expression must match with the type of value specified in expression. It means that if variable is of int type then all case values

Should be integer.



WORKING OF SWITCH CASE

Prog

```
class show
{
public static void main(String args[])
{ int choice;
  choice=3;
  switch(choice)
  {
  case 1:
        System.out.println("Hello");
  case 2:
        System.out.println("How are you");
  case 3:
        System.out.println("Get well soon");
  case 4:
        System.out.println("Take Care");
  default:
        System.out.println("Good bye");
  }
}
}
```

O/P - Get well soon
Take Care
Good bye

Switch case using break

```
class Show
{ public static void main (String args[])
  {
      char choice;
      choice = 'B';
     switch (choice)
      {
      case 'A':
              System.out.println ("Apple");
                break;
      case 'B':
              System.out.println ("Ball");
                break;
      case 'C':
              System.out.println ("carpet");
                break;
      case 'D':
              System.out.println ("Dust");
                break;
      default:
              System.out.println ("No match")
      }
  }
}.
```

O/P . Ball

~~Class~~

Prog to execute same set of statements for multiple cases.

```
class Show
{
    public static void main (String args[])
    {
        int a;
        a = 1;
        Switch(a)
        {
            Case 1:
            Case 2:
            Case 3:
                System.out.println ("Good Day");
                break;
            Case 4:
            case 5:
                System.out.println("Good Bye");
                break;
            default:
                System.out.println ("default");
        }
    }
}
```

O/P "Good Bye"

Some points to remember while using switch.

(a) It is not necessary to write case literals in sequence as 1,2,3 and so on. You may write it in any order in which you like as 34, 2, 7.

(b) You need not to include braces inside cases if you want to point multiple statements as you need to include in case of if - else.

(c) You can't use floating-point numbers as case constant.

(d) Each statement written inside switch must belong to some case. Though it will not give any error but on the same way that statement will never be executed.

(e) You can use constant expressions like 4*9 in the switch expression but can't use integer or relational or any other type of expression.

following are wrong

```
switch (a+b)
switch (a>b)
```

but you can use it like this

```
switch (6+1)
```

Nested Switch

When we define a switch inside a switch.

case then it is termed as nested switch. Lets see an example.

- Program to show concept of nested switch

```
class Show
{
    public static void main (String args[])
    {
        int a, b;
        a=1, b=2;
        switch(a)
        {
            case 1:
                switch(b)
                {
                    case 2:
                        System.out.println("Good");
                        break;
                    case 3:
                        System.out.println("Bad");
                        break
                } break;
            case 4:
                system.out.println("Good Bye");
                break;
            default:
                system.out.println("default");
        }
    }
}
```

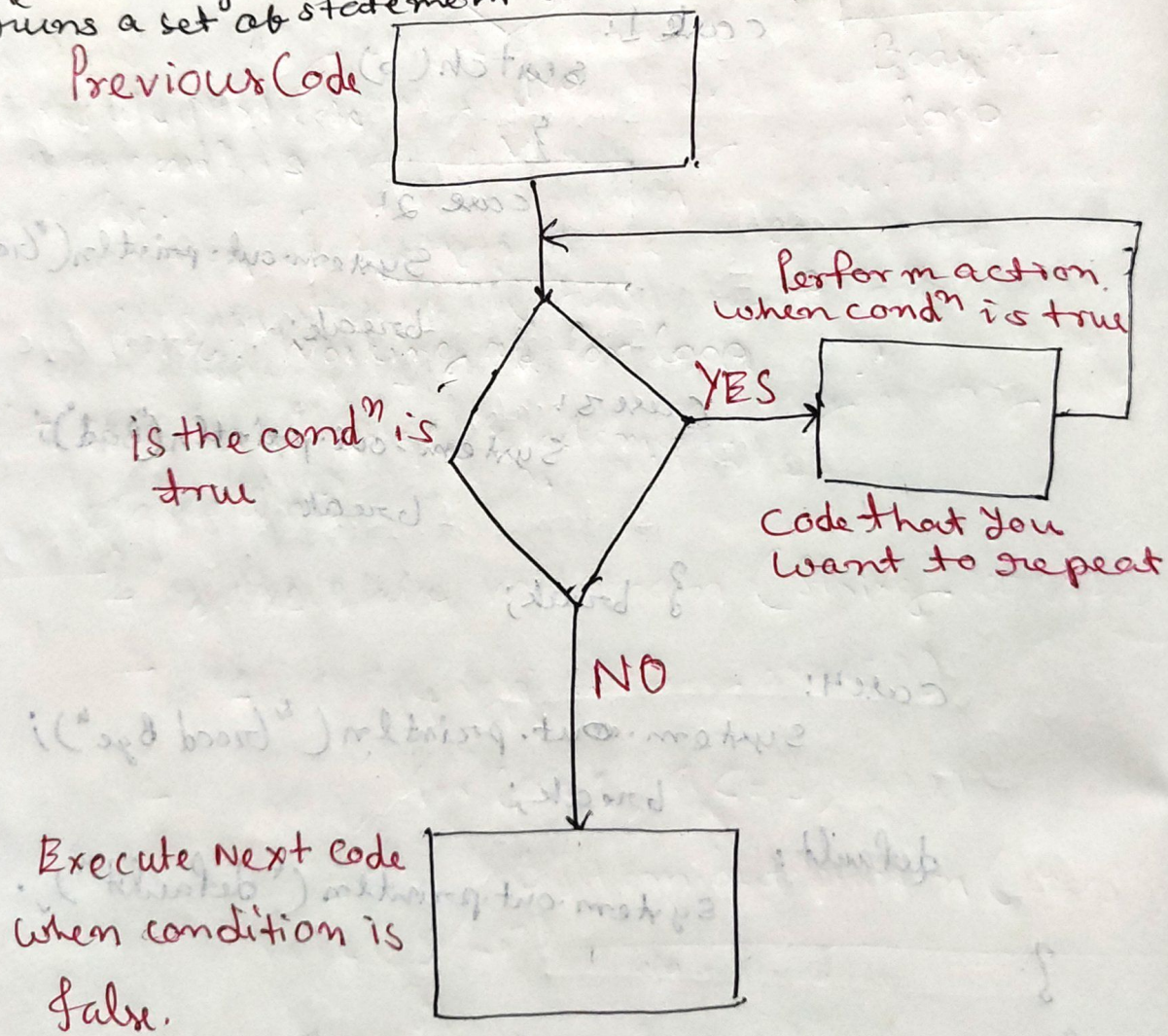
O/p – Good

ITERATION CONTROL STATEMENT

Iteration in general means that act or an instance of iterating, repetition. If we talk about mathmetics it refers to a computational procedure in which a cycle of operations is repeated, often to approximate the desired result more closely.

In computer science it means the process of repeating a set of instructions a specified number of times or until a specific result is achieved.

The following structure shows a loop structure that runs a set of statement until a condⁿ becomes true.



Running a set of statement until a condⁿ becomes false.

There are three types of iteration control statement in java.

- For Loop
- While Loop
- Do While Loop.

for Loop :

for loop is the most commonly used loop. It is used to display values repeatedly as per condition given. Task is repeated until the condition becomes false. for statement is started with the keyword for. There are three expressions which appear with in a for statement

- Initializing Expression.
- Test Expression.
- Increament/Decrement Expression.

Initializing Expression: variable is initialized i.e it is assigned a value.

Test Expression: here condition is specified.

Increament/Decrement Expression: value of variable is incremented/Decremented. here

The general form

```
for (initialization; condition; increment/Decrement)
{
        statement block;
}.
```

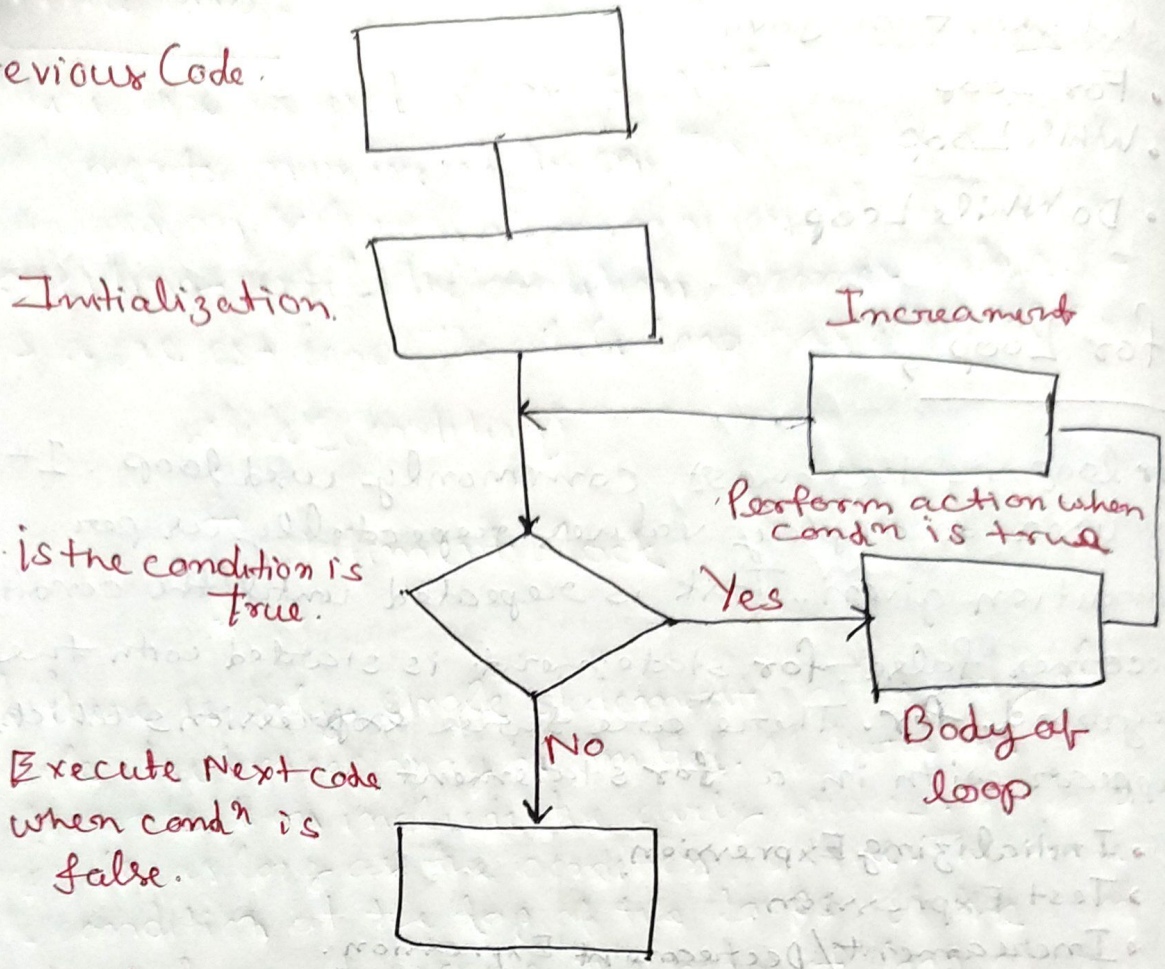


figure: Working of for loop

for loop with single statement.

```
class Show
{ public static void main (String args[])
  {
      int a;
      for (a=1; a<=5; a++)
          System.out.println ("Good bye"
  }
}.
```

```
o/p - Good bye
      Good bye
      Good bye
      Good bye
      Good bye.
```

For loop with multiple statement

```
class Show
{
    public static void main (String args[])
    {  int i;
       for (i = 1, i <= 5, i ++)
       {
           System.out.print ("Good");
           System.out.println ("value of i = "+i);
       }
    }
}
```

O/P

Good value of i = 1
Good value of i = 2
Good value of i = 3
Good value of i = 4
Good value of i = 5

Prog that will declare a loop ~~inside~~ variable inside the for statement.

```
Class Show
{   public static void main (String args[])
    {  for (int i=0; i<=10; i+=2)
       {
           System.out.print ("Good day");
           System.out.println(+i);
       }
    }
}
```

O/P Good day 0 " 4 " 6 " 10
Good day 2 " 6 " 8

while loop

while loop is used when we want to execute set of statements number of times. In this loop code execution is based on truthness of a condition. It will go on executing until the condition becomes false. It is a pre-test loop.The while loop has the general form:

```
while(condition)
{
   statement block
}
```

If there is only a single statement in the body of **while**, the braces may be omitted. If there are multiple statements then include the curly braces. The loop will repeat as long as the condition is true. The **while** tests its condition at the top of the loops. If the condition is true then it execute the body of loop otherwise if the condition is false to begin with, the loop will not execute at all. The condition may be any expression which evaluates boolean constants.

Program that demonstrate use of while

```
class Show          //Class declaration
 {
            // call to main function
   public static void main( String args [] )
    {
          int i ;
```

```
        i = 1;
        while(i< 5)
            {
                System.out.println("Nandini");
                    i = i + 1;
            }
    }
}
```

OUTPUT

Nandini
Nandini
Nandini
Nandini

Program to print factorial of a number.

```
        // A Java Program to print factorial of a number
class Show        //Class declaration
{
        // call to main function
    public static void main (String args [])
        {
            int i, num, fact;
            i = 1;
            num = 5;
            fact = 1;
            while(i<=num)
                {
                    fact=fact*i;
                    i = i + 1;
                }
            System.out.println ("Factorial of 5 =" +fact);
        }
    }
```

OUTPUT

Factorial of 5 =120

Program to check whether given number is palindrome or not.

Do while loop

Do while loop is used to execute a statement or set of statements number of times based on some condition. It is post-test loop, as in this loop condition is tested after execution of statement.

The general form of the do loop is

```
do
{
    statement block
} while (condition);
```

If only one statement is included, the braces are not necessary, but they add clarity to the statement. The do loop repeats as long as the condition is true. The do loop is the only loop that will always have at least one iteration because the condition is tested at the bottom of the loop. The diagram below shows how do-while works.

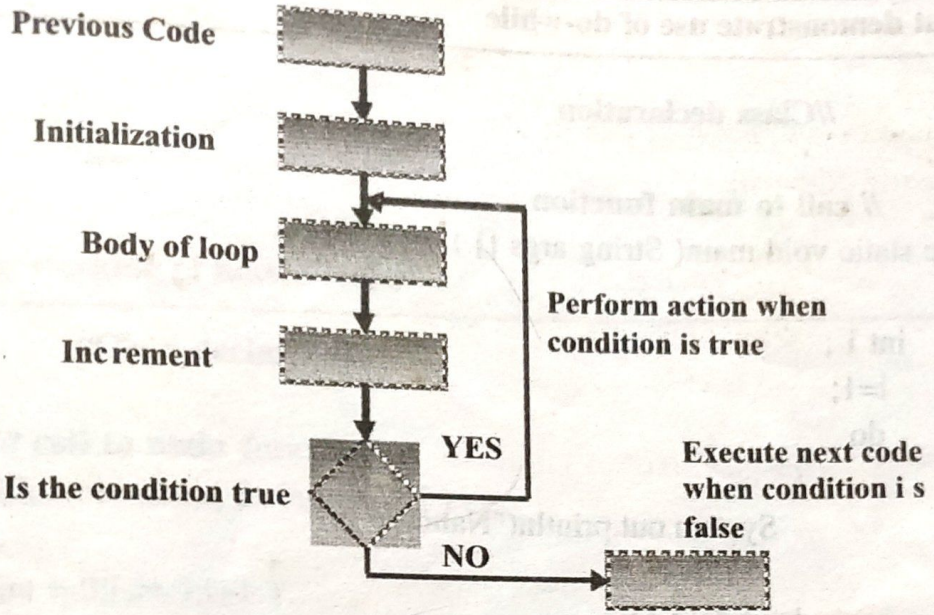


Figure : Working of do-while

Program that demonstrate use of do-while

```
class Show          //Class declaration
 {
            // call to main function
   public static void main( String args [] )
     {
          int i ;
           i=1;
           do
                {
                  System.out.println("Nandini");
                     i=i+1;
                } while(i<5);
        }
   }
```

OUTPUT

Nandini
Nandini
Nandini
Nandini

Program that demonstrate use of do-while

```
class Show          //Class declaration
  {
                // call to main function
     public static void main( String args [] )
       {
            int i ;
             i=1;
             do
                 {
                     System.out.println("Nandini");
                        i=i+1;
                 } while(i>5);
        }
     }
```

OUTPUT

Nandini