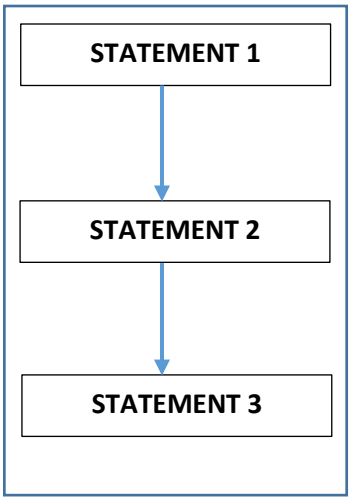
CHAPTER 5 FLOW OF CONTROL

PROGRAMMING CONSTRUCTS

- In a program, statements may be executed sequentially, selectively or iteratively.
- Every programming language provides constructs to support sequence, selection or iteration.

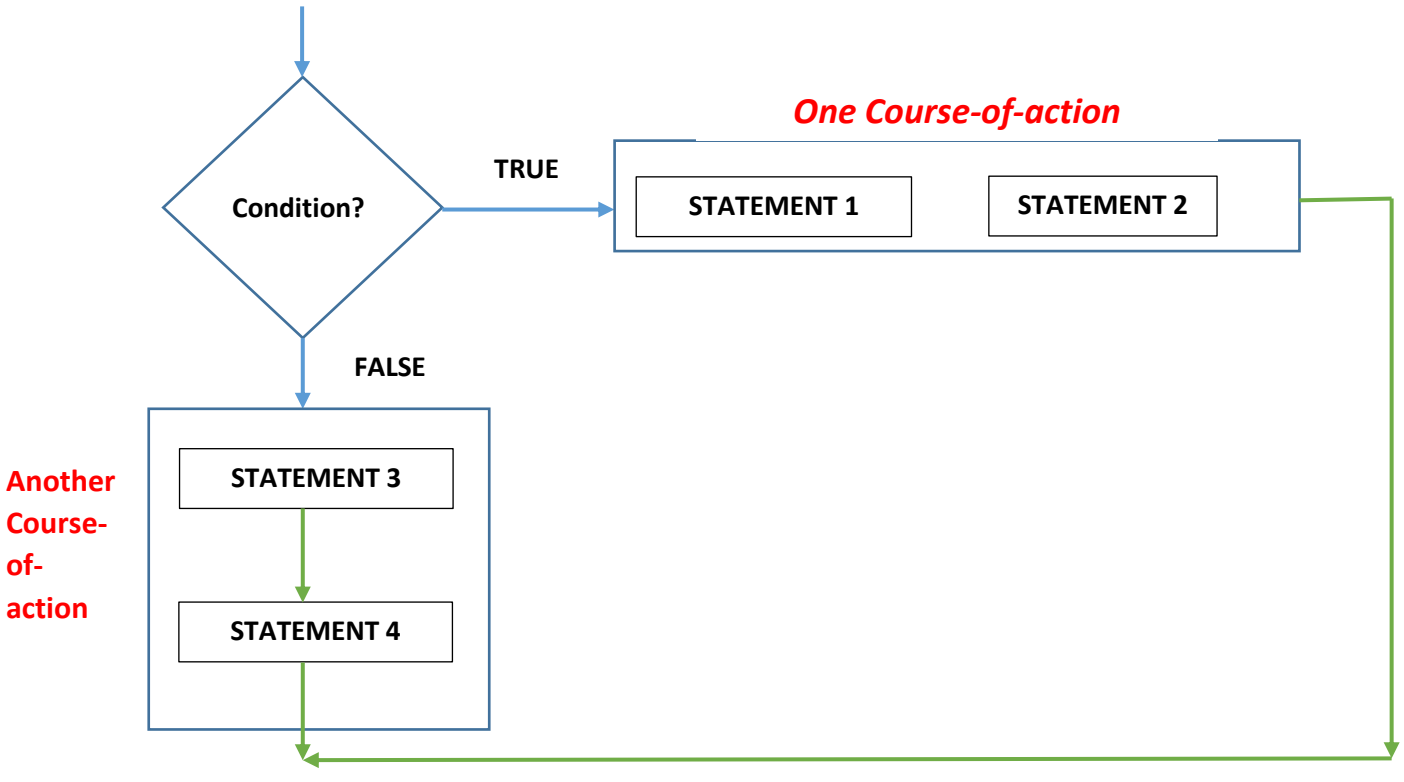
SEQUENCE

- The sequence construct means the statements are being executed sequentially.
- Java execution starts with first statement, and then each statement is executed in turn.



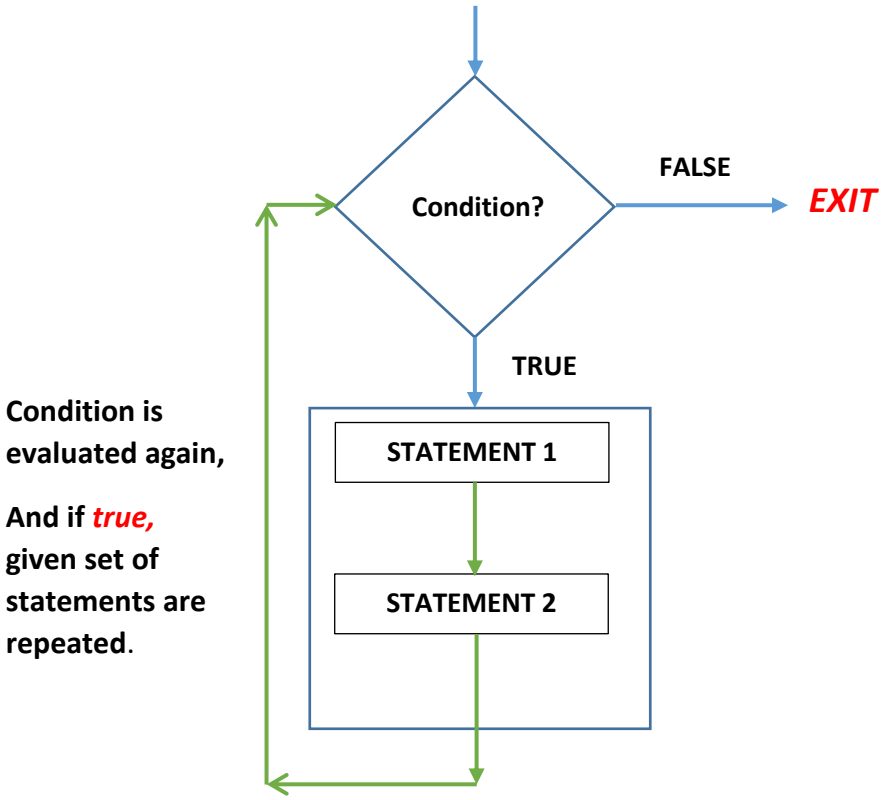
SELECTION

- The selection construct means the execution of statement(s) depending upon a condition-test.
- If a condition evaluates to true, a course of action (a set of statements) is followed otherwise another course of action (a different set of statements) is followed.



ITERATION

- The iteration construct means **repetition** of a set of statements depending upon a condition-test. Till the time a condition is true, a set of statements are repeated again and again. As soon as the condition becomes false, the repetition stops.
- The iteration construct is also called **looping construct**.



- The set of statements that are repeated again and again is called the **body of loop**.

SELECTION STATEMENTS

- The selection statements allow to choose the set of instructions for execution depending upon an expression's truth value.
- Java provide two types of selection statements : **if** and **switch**.

The if statement of Java

- An if statement tests a particular condition; if the condition evaluated to true, a statement or set of statements is executed.
- The Syntax (general form) of the if statement is as shown below :

if (**boolean expression/condition**)
statement ;

where a statement may consist of a single statement or a compound statement. The condition must be enclosed in parentheses. If the condition evaluates to true, the statement is executed, otherwise ignored.

e.g.
```
if (ch == 2)
    R.setText("It is number 2");
```

e.g.2 Write code to test whether a number stored in variable **Num** is positive or not. Make use of if statement only.

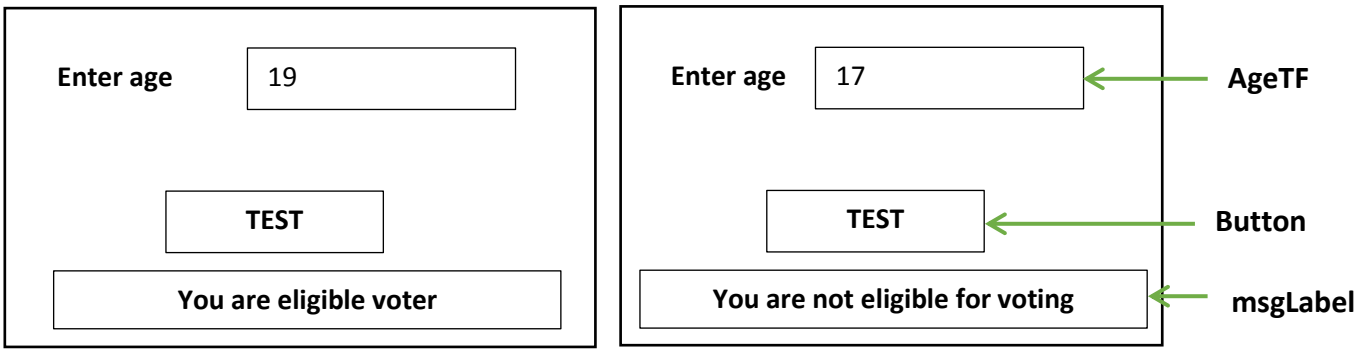
```
if (Num > 0)
  RTF.setText ("Number is positive");
```

Explanation

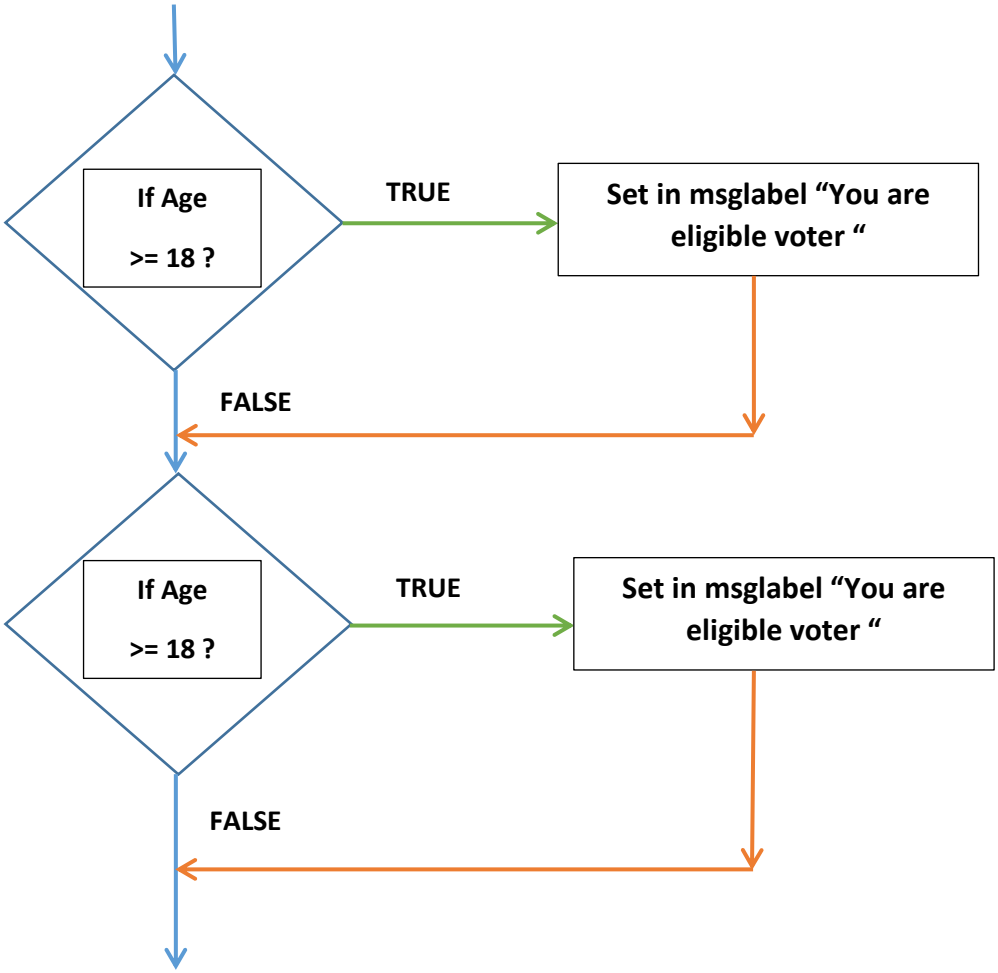
The if statement makes a decision , if the value of the logical expression Num > 0 is ***true***, the statement (***RTF.setText ("Number is positive");)*** gets executed ; if the value of the logical expression is ***false,*** it does not execute the statement. That means no message is printed when the condition (***Num > 0***) is false.

Question:

Obtain age of a person and then display whether he/she is eligible for voting.

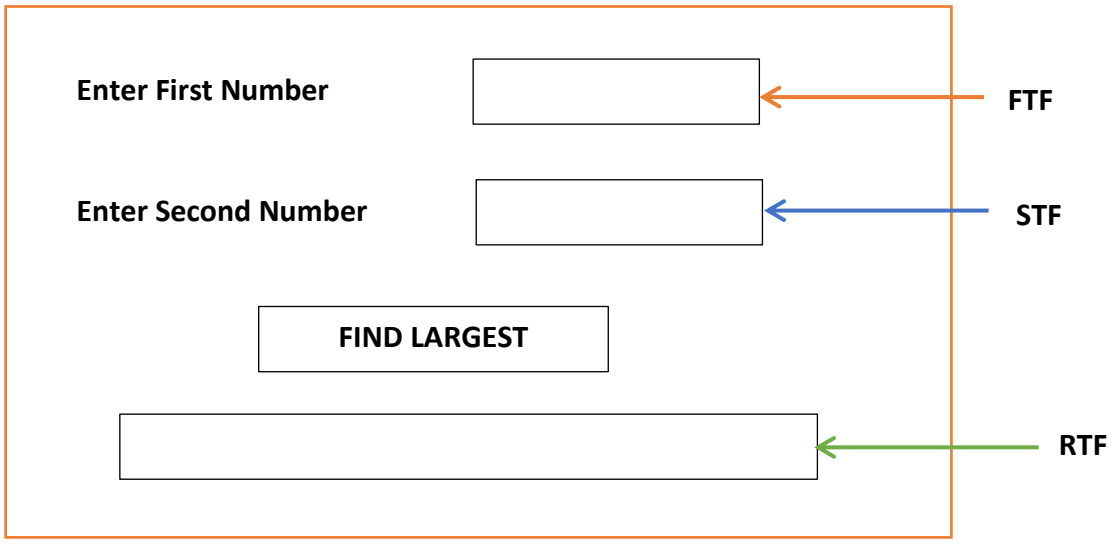



```
int age ;
  String txt = ageTF.getText ( );
  age = Integer.parseInt(txt);
    if (age >= 18)
        msgLabel.setText ("You are eligible voter") ;
    if (age < 18)
        msgLabel.setText("You are not eligible for voting");
```



Question:

Find the largest of two numbers.

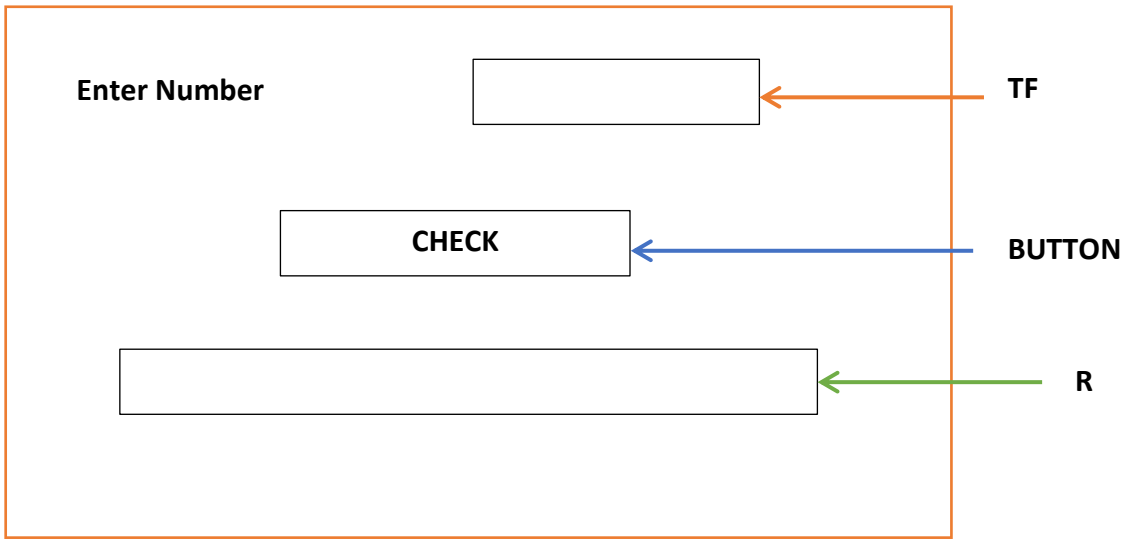


Solution:

```
String a = FTF.getText ( );
 int n1 = Integer.parseInt (a);
String b = STF.getText( );
 int n2 = Integer.parseInt (b) ;
    if (n1 > n2)
        RTF.setText(" "+n1) ;
    if (n2 > n1)
        RTF.setText(" "+n2) ;
```

Question:

Find whether number entered in textfield is positive,negative or zero.



Solution:

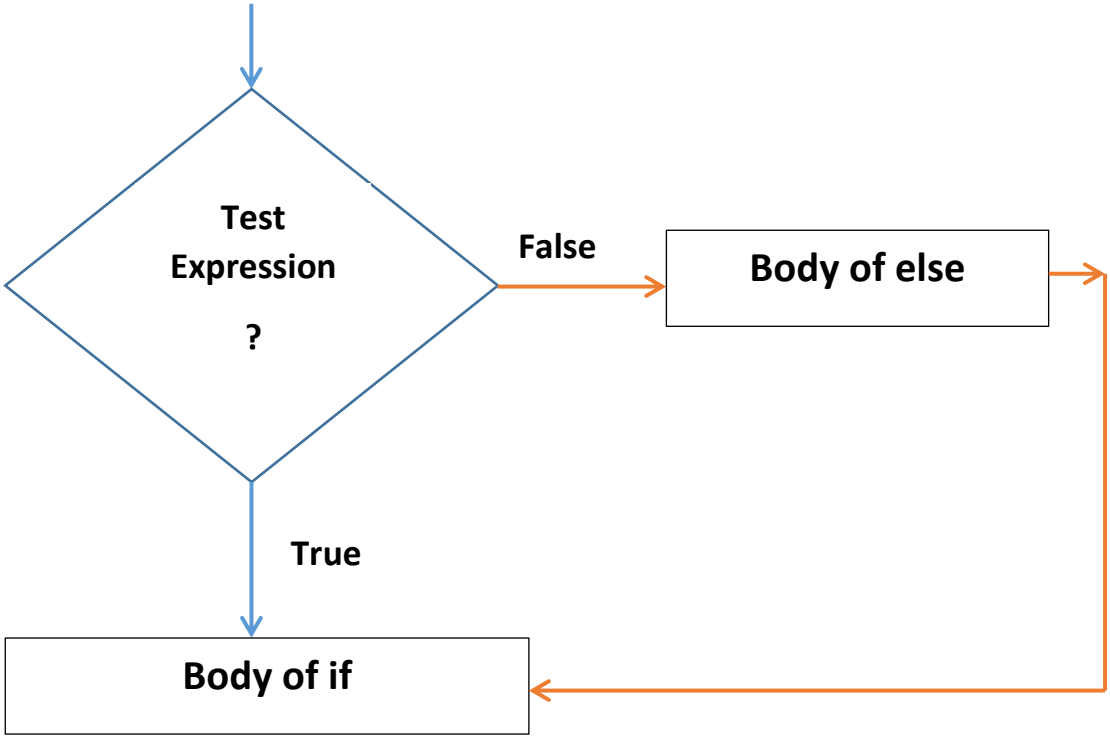
```
String q = TF.getText ( );
 int n = Integer.parseInt (q);
      if (n > 0)
          R.setText ("Number is positive");
      if (n < 0)
           R.setText ("Number is negative");
     if (n == 0)

           R.setText ("Number is zero");
```

if else statement

- The syntax (general form) of if else statement is the following :

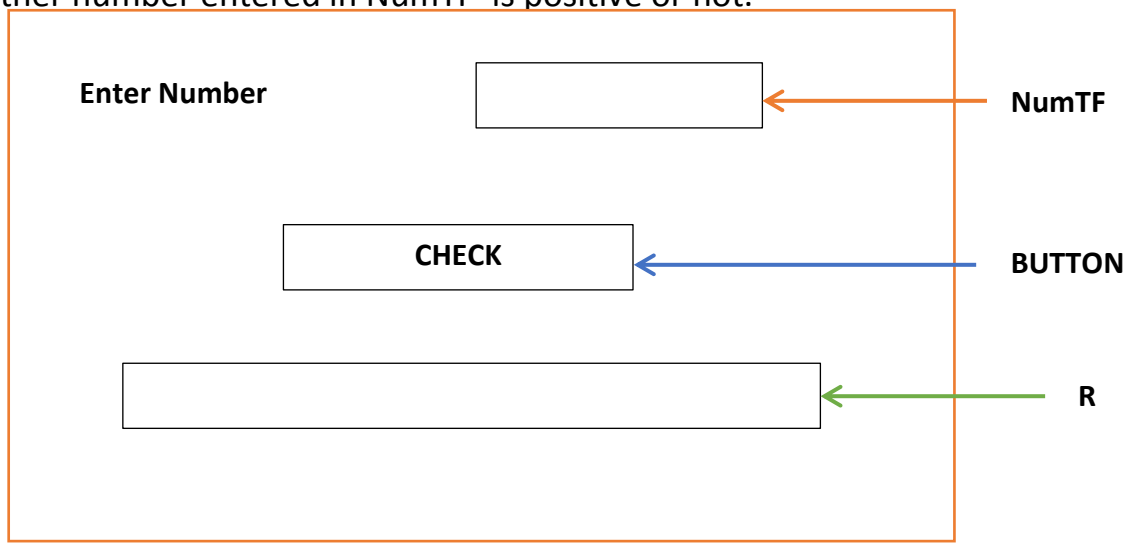
```
if (expression)
    statement 1 ;
else
    statement 2 ;
```

If the **expression** evaluates to ***true***, the statement 1 is executed, otherwise statement 2 is executed. The statement 1 and statement 2 can be single statement or compound statement.



Question:

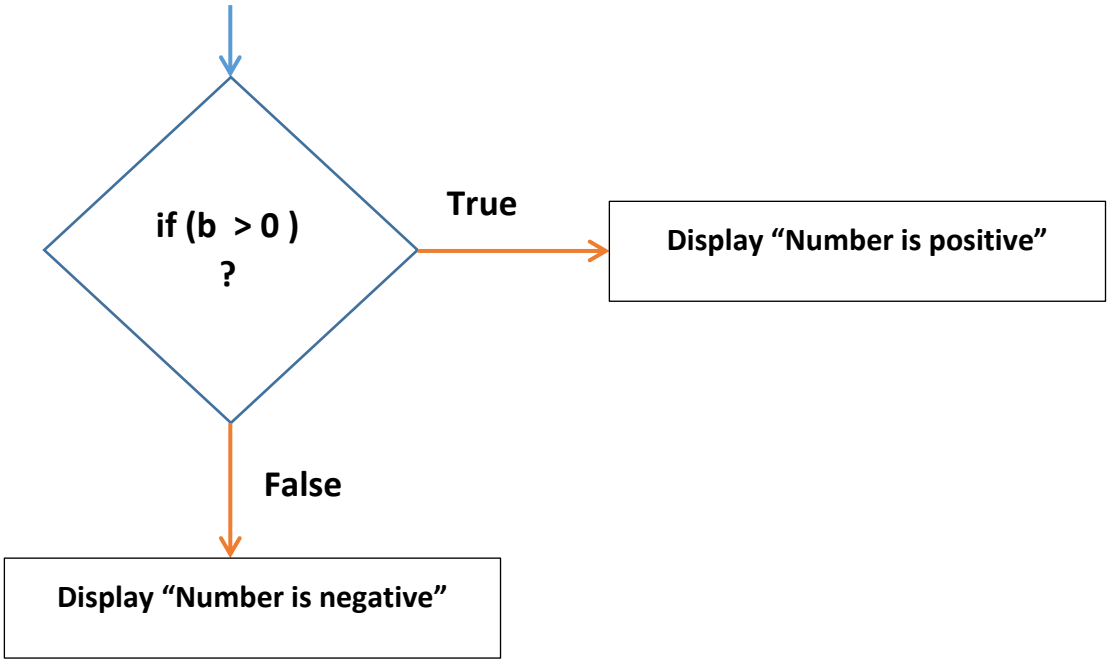
Find whether number entered in NumTF is positive or not.



Solution:

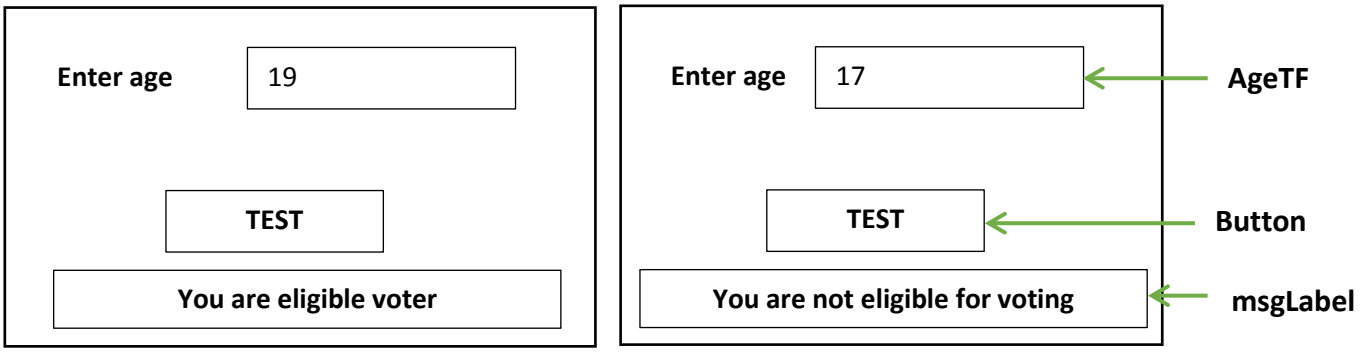
```
String a = NumTF.getText ( );
int b = Integer.parseInt(a);
if (b > 0 )
        R.setText( " Number is positive ");
else
        R.setText( " Number is negative ");
```

Explanation



Question:

Obtain age of a person and then display whether he/she is eligible for voting.**(using if else)**

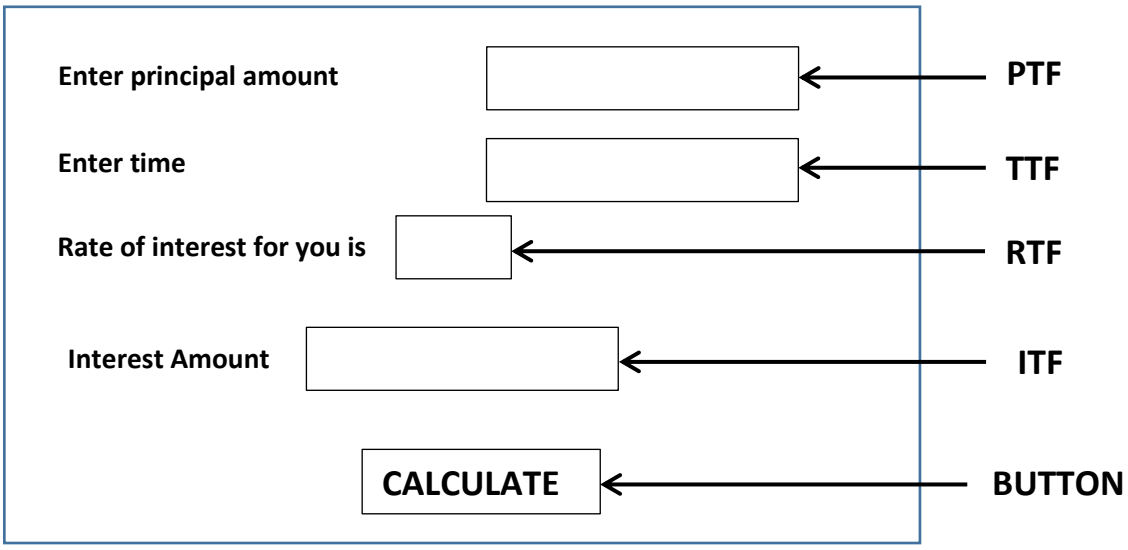


Solution:

```
String x = AgeTF.getText ( );
int age = Integer.parseInt(x);
  if (age >= 18)
        msgLabel.setText("You are eligible voter");
  else
        msgLabel.setText("You are not eligible for voting");
```

Question:

Obtain principal amount and time and calculate simple interest as per following specifications: **If principal is greater than or more than Rs.10000**, then rate of interest is 6% otherwise it is 5%.



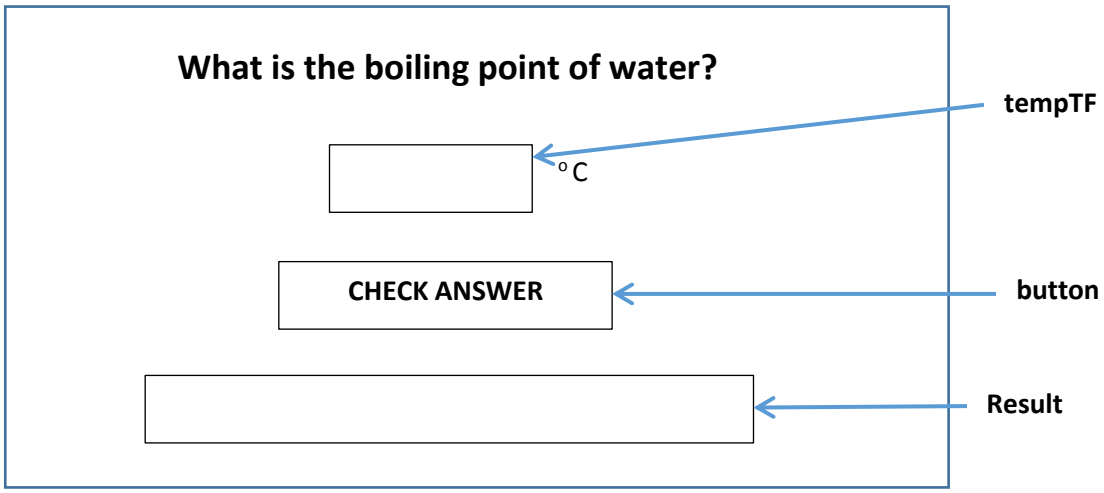
Solution:

```
double rate;
String t =  PTF.getText( );
int prin = Integer.parseInt(t);
String g = TTF.getText( );
int time = Integer.parseInt(g);
      if (prin >= 12000)
      {
           RTF.setText("6%");
           rate =0.06 ;
       }
      else
       {
            RTF.setText("5%");
            rate =0.05 ;
       }
         double intr = prin * time * rate ;
         ITF.setText(" " + intr);
```

***** the curly brackets are not required if only ONE statement follows if or else, but in case of multiple statements, curly brackets are required.***

Question:

Program to obtain boiling point of water from user and report whether the user has entered the correct answer or not. Make use of if else statement.
(Hint : Boiling point of water is 100^{o} C).



Solution:

```
String t = tempTF.getText( );
int temp = Integer.parseInt(t);

  if(temp == 100)
        Result.setText("You are right. It is 100° C") ;
  else
        Result.setText("Your answer is wrong") ;
```

Nested if

- A nested if is an if that has another if in its if's body or in its else's body.
- The nested if can have one of the following three forms:

1. **If statement nested within if body**

```
    if(expression1) {

            if(expression2)
                  statement 1 ;
            [ else
                  statement 2 ; ]
       }
        else
             body of else ;
```

2. **If statement nested within else body**

```
      if(expression 1)
            body of if ;

            else {
                  if(expression2)
                     statement 1;
                 [ else
                      Statement 2; ]
               }
```

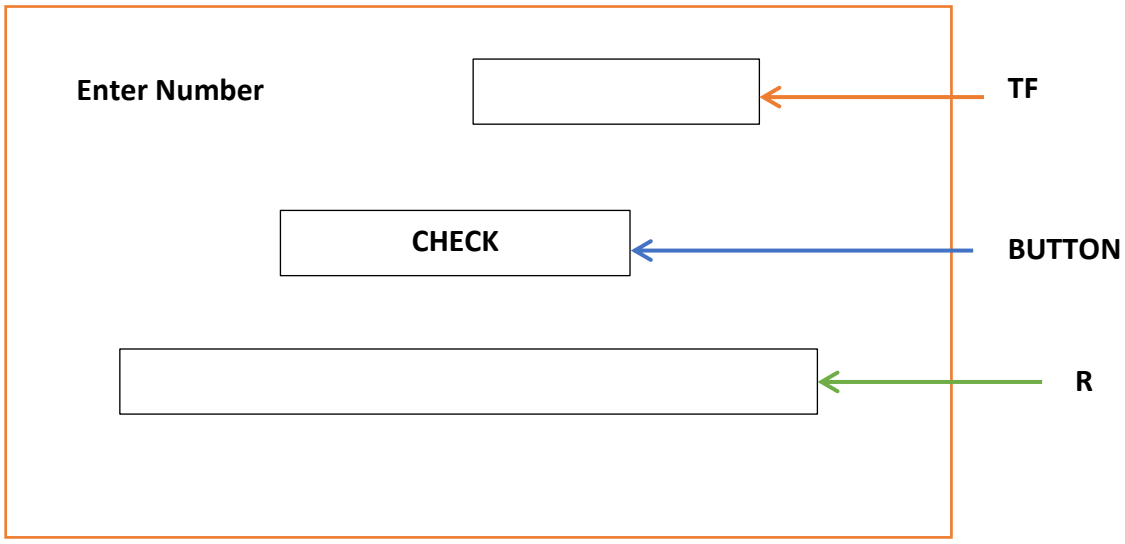
3. If statement nested within if and else body

```
if (expression1) {

        if(expression2)
          statement 1;
        [ else
            statement 2 ; ]
    }
      else {

        if(expression2)
          statement 1;
        [ else
            statement 2 ; ]
  }
```

** **The part in [ ] means, it is optional.**

Question:

Find whether number entered in textfield is positive,negative or zero. (using nested if)

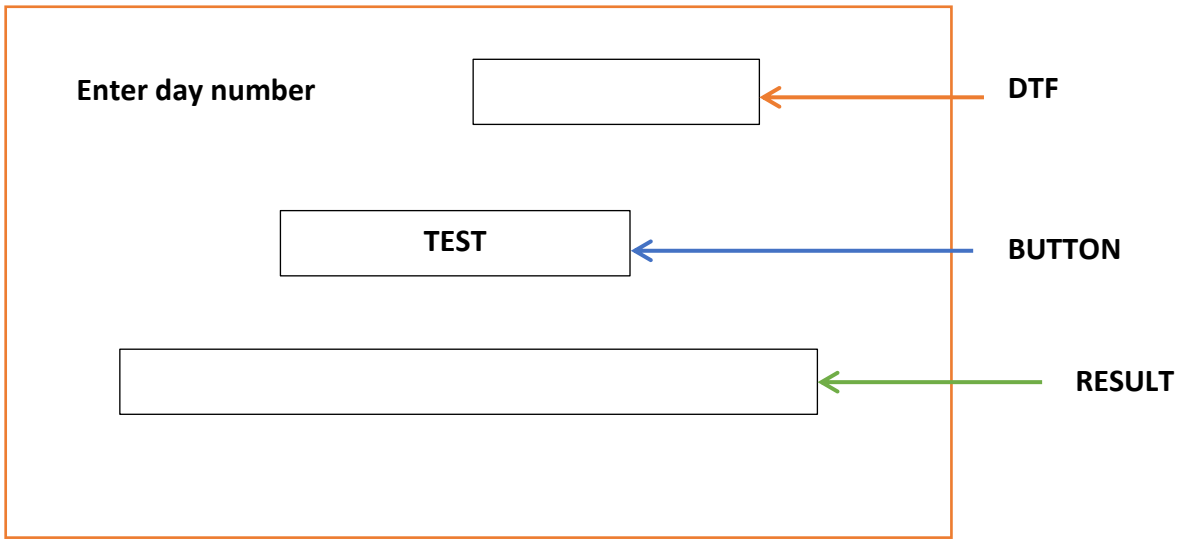


Solution:

```
String q = TF.getText ( );
int n = Integer.parseInt (q);
    if (n > 0)
        R.setText ("Number is positive");
    else {
            if(n < 0)
                R.setText ("Number is negative");
            else
                R.setText ("Number is zero");
             }
```

Question:

Write code to read the Day number and display the weekday. E.g. for 1 -> Sunday, 2-> Monday etc.



Solution:

Using if statement only

```
String a = DTF.getText( );
int  w = Integer.parseInt(a);
if(w == 1)
      RESULT.setText(“Day is Sunday”);
 if(w == 2)
      RESULT.setText(“Day is Monday”);
 if(w == 3)
      RESULT.setText(“Day is Tuesday”);
 if(w == 4)
      RESULT.setText(“Day is Wednesday”);
 if(w == 5)
      RESULT.setText(“Day is Thursday”);
 if(w == 6)
      RESULT.setText(“Day is Friday”);
 if(w == 7)
      RESULT.setText(“Day is Saturday”);
```

Using nested if statement

```
String a = DTF.getText( );
int  w = Integer.parseInt(a);
if(w == 1)
        RESULT.setText("Day is Sunday");
else {
        if(w == 2) {
           RESULT.setText("Day is Monday");
                }
         else {
                if(w == 3) {
                   RESULT.setText("Day is Tuesday");
                        }
                else {
                        if(w == 4) {
                          RESULT.setText("Day is Wednesday");
                              }
                   else {
                        if(w == 5) {
                          RESULT.setText("Day is Thursday");
                              }
                         else {
                              if(w == 6) {
                                 RESULT.setText("Day is Friday");
                                       }
                              else {
                                  if(w == 7) {
                                     RESULT.setText("Day is Saturday");
                                        }
                                }
                            }
                        }
                }
        }
```

The Dangling-else Problem

- The nested if-else statement introduces a source of potential ambiguity referred to as dangling-else problem.
- ***This problem arises when in a nested if statement, number of ifs is more than the number of else clause.***
- The question then arises, with which **if** does the additional else clause property matchup. e.g.

```
if (ch > = 1)        <-- outer if
    if(ch >= 4)      <-- inner if
        ++ucase ;
else
        ++others;
```

In this code, it is not clear whether the else statement belongs to inner if or outer if.

- Java matches a **dangling else statement with the preceding unmatched if statement**.

e.g.2.

```
if(expr1)        <-- outer if
   if(expr2)     <-- inner if
        statement -1;
   else          <-- inner else
        statement -2;
   else          <-- outer else
        statement -3;
```

the inner else goes with immediately preceding unmatched if which is inner if. The outer else goes with immediately preceding unmatched if which is now outer if. Thus ***statement-3*** gets executed if expr1 is **false**.

- The second way to resolve dangling else problem is to **place the last occurring unmatched if in a compound statement**, as shown below :

```
if(expr1)  {      <-- outer if
   if(expr 2)     <-- inner if
       statement 1;
   }
else
       statement 2;
```

The if else if ladder

- It takes the following general form :

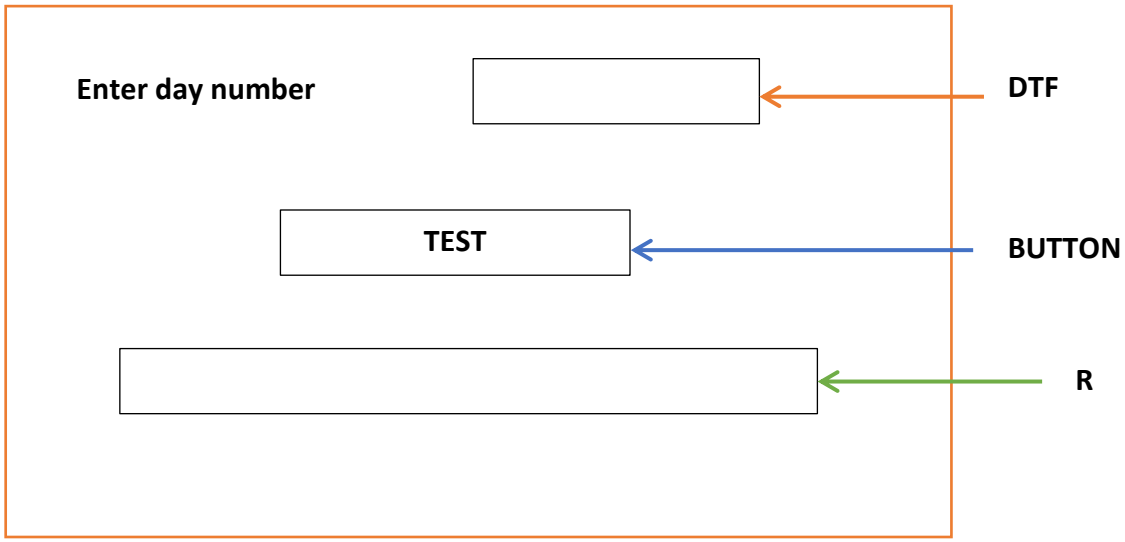
```
if (expression 1)
      statement 1;
else if (expression 2)
      statement 2;
 else if (expression 3)
      statement 3;
      .
      .
      .
 else
      statement 4 ;
```

The ***expressions*** are evaluated from the top downward. As soon as an ***expression*** evaluates to **true**, the statement associated with it is executed and the rest of the ladder is bypassed or ignored. If **none of the expressions are true**, the ***final else*** gets executed.

e.g.

Question:

Write code to read the Day number and display the weekday. E.g. for 1 -> Sunday, 2-> Monday etc. **(Using if else if ladder)**

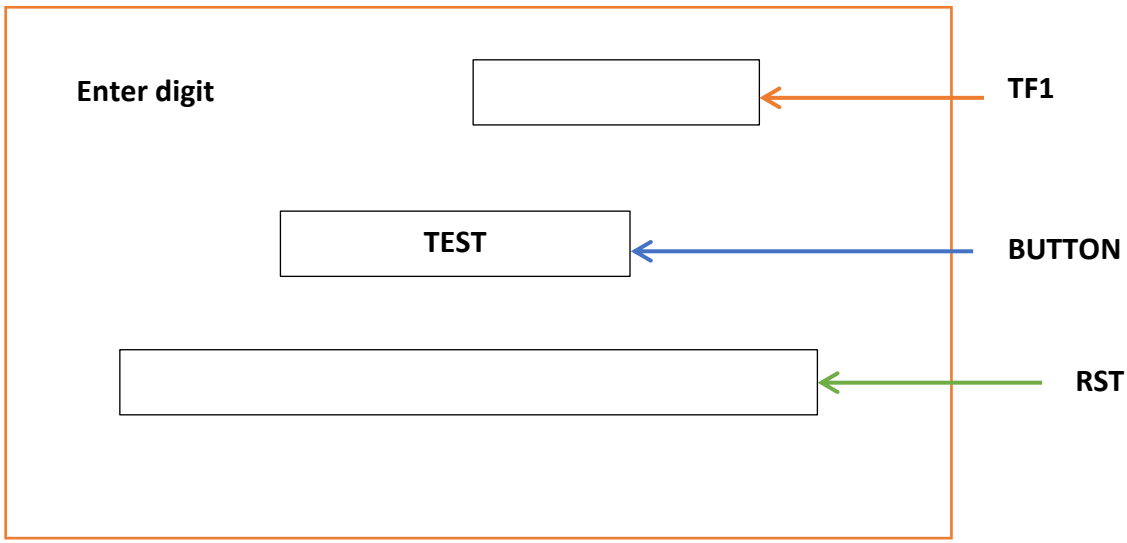


Solution:

```
String a = DTF.getText( );
int  w = Integer.parseInt(a);
if(w == 1)
      RESULT.setText("Day is Sunday");
 else if(w == 2)
      RESULT.setText("Day is Monday");
 else if(w == 3)
      RESULT.setText("Day is Tuesday");
 else if(w == 4)
      RESULT.setText("Day is Wednesday");
 else if(w == 5)
      RESULT.setText("Day is Thursday");
 else if(w == 6)
       RESULT.setText("Day is Friday");
 else if(w == 7)
       RESULT.setText("Day is Saturday");
 else
       RESULT.setText("Please Enter no. between 1 & 7");
```

Question:

Obtain a single digit (0 – 9) and display it in words. **(Using if else if ladder)**



Solution:

```
String z = DTF.getText( );
int  dig = Integer.parseInt(z);
if(dig == 0)
      RESULT.setText("Digit is zero");
 else if(dig == 1)
      RESULT.setText("Digit is one");
 else if(dig == 2)
      RESULT.setText("Digit is two");
 else if(dig == 3)
      RESULT.setText("Digit is three");
 else if(dig == 4)
      RESULT.setText("Digit is four");
 else if(dig == 5)
       RESULT.setText("Digit is five");
 else if(dig == 6)
       RESULT.setText("Digit is six");
 else if(dig == 7)
       RESULT.setText("Digit is seven");
 else if(dig == 8)
       RESULT.setText("Digit is eight");
 else if(dig == 9)
       RESULT.setText("Digit is nine");
 else
       RESULT.setText("Please enter digit between 0 and 9");
```

The ?: Alternative to if

```
if(expression1)
    statement 1;
 else
    statement 2;
```

The above form of if can be alternatively written using ?: as follows:

expression1? expression2: expression 3 ;

It works in the same way as the above given form of if.

e.g.

```
int c ;
if (a > b)
    c = a;
else
    c = b;
```

can be written as :

```
int c = a>b ? a : b ;
```

Comparing if and ?:

1. compared to if else sequence, ?: offers more concise, clean and compact code, but it is less obvious as compared to if.

2. When ?: operator is used in its nested form, it becomes complex and difficult to understand. This form of ?: is generally used to conceal the purpose of code.

The switch statement

- It is a multiple branch selection statement.
- This selection statement successively tests the value of an ***expression*** against a list of ***integer or character constants***. When a match is found, the statements associated with that constant are executed.

- **Syntax of switch statement is as follows:**

```
switch (expression)
{
   case constant1 :   statement1;
                      break ;
   case constant2 :   statement2 ;
                      break;
   case constant3  :   statement2 ;
                      break ;
    .
    .
    .
    default : statement ;
}
```

- The ***expression*** is evaluated and its values are matched against the values of the ***constants specified in the case statements***. When a match is found, the statement associated with that case is executed until the break statement or the end of switch statement is reached.

- The **default statement gets executed when no match is found**.

- If the program control flows to the next case below the matching case, in the absence of break, this is called **fall through.**

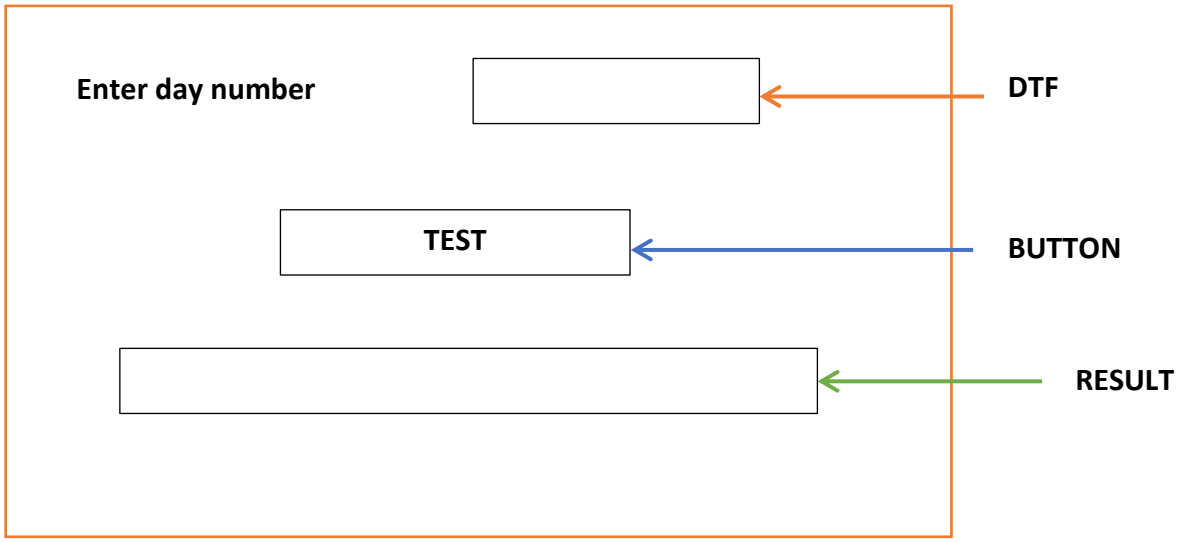
Points to remember:

1. A **case statement** cannot exist by itself, **outside a switch**.
2. The **break statement**, used under switch, is one of **Java Jump Statements**.
3. ***When a break statement is encountered in a switch statement, program execution jumps to the line of code outside the body of switch.***

e.g.1.

Question:

Write code to read the Day number and display the weekday. E.g. for 1 -> Sunday, 2-> Monday etc. **(Using switch statement)**

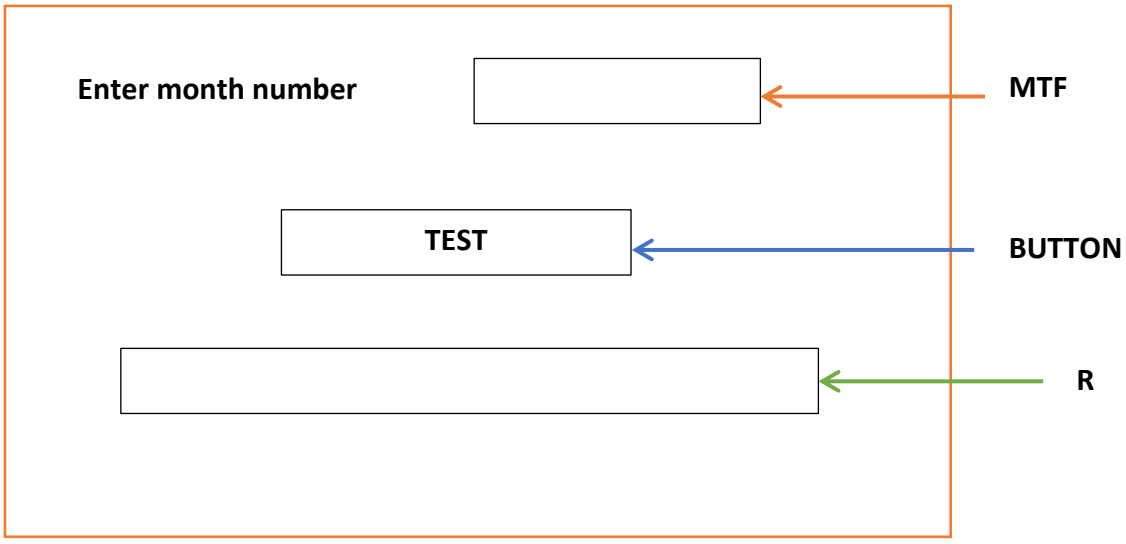



```
String a = DTF.getText( );
int  w = Integer.parseInt(a);
switch(w) {

    case 1 : RESULT.setText("Day is Sunday");
             break ;
    case 2 : RESULT.setText("Day is Monday");
             break ;
    case 3 : RESULT.setText("Day is Tuesday");
             break ;
    case 4 : RESULT.setText("Day is Wednesday");
             break ;
    case 5 : RESULT.setText("Day is Thursday");
             break ;
    case 6 : RESULT.setText("Day is Friday");
             break ;
    case 7 : RESULT.setText("Day is Saturday");
             break ;
   default : RESULT.setText("Enter Number between 1&7");

}
```

e.g.1.

Question:

Write code to read the months number and display the name of month . E.g. for 1 -> January, 2-> February, 3-> March, etc.**(using switch statement)**




```
String x = MTF.getText( );
int  m = Integer.parseInt(x);
switch(w) {
    case 1 : R.setText("Month is January");
             break ;
    case 2 : R.setText("Month is February");
             break ;
    case 3 : R.setText("Month is March");
             break ;
    case 4 : R.setText("Month is April");
             break ;
    case 5 : R.setText("Month is May");
             break ;
    case 6 : R.setText("Month is June");
             break ;
    case 7 : R.setText("Month is July");
             break ;
    case 8 : R.setText("Month is August");
             break ;
    case 9 : R.setText("Month is September");
             break ;
    case 10 : R.setText("Month is October");
             break ;
```

```
        case 11 : R.setText("Month is November");
                  break ;
        case 12 : R.setText("Month is December");
                  break ;
        default : R.setText("Enter Number between 1&12");

    }
```

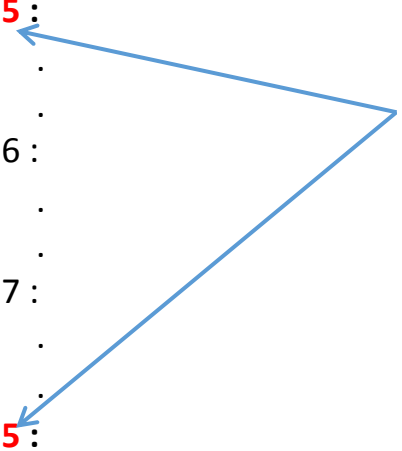
Difference between switch and if else

	switch statement	if else statement
1.	switch statement can only test for Equality.	if else statement can evaluate a relational or logical expression.
2.	switch statement cannot handle floating point tests.The case labels of switch must be byte, short, int or char.	if else statement can handle floating tests apart from integer test.
3.	switch case label value must be a constant.	if two or more variables are to be compared, use if else statement.

Some important points about switch statement

- Multiple identical case expressions are not allowed. E.g.

```
switch(val) {
        case 5 :
            .
            .
        case 6 :
            .
            .
        case 7 :
            .
            .
        case 5 :
            .
            .
    }
```



Two identical case expressions are not allowed

Example to illustrate the working of switch in the absence of break statement.

What will be the output of following code fragment if the value of ch is
(i) a (ii) c (iii) d (iv) h (v) b

```
switch(ch)  {
        case 'a' : System.out.println("It is a");
        case 'b' : System.out.println("It is b");
        case 'c' : System.out.println("It is c");
                   break ;
        case 'd' : System.out.println("It is d");
                   break ;
        default : System.out.println("Not abcd");

    }
```

Conversion from if else to switch & switch to if else

Rewrite the following code fragment using switch:

```
if(a == 0)
   System.out.println("Zero");
if(a == 1)
   System.out.println("One");
if(a == 2)
   System.out.println("Two");
if(a == 3)
   System.out.println("Three");
```

Rewrite the following code fragment using if :

```
String Remarks;
int code = Integer.parseInt(jTextField1.getText( ) );
switch (code)
 {
   case 0 : Remarks ="100% Tax Exemption";
            break;
   case 1 : Remarks = "50% Tax Exemption";
            break;
   case 2 : Remarks ="25% Tax Exemption";
           break;
   default : Remarks = "Invalid Entry!";
 }
```

ITERATION STATEMENTS

- The iteration statements allow a set of instructions to be performed repeatedly until a certain condition is fulfilled. The iteration statements are also called loops or looping statements.
- Java provides **three kinds of loops :** 1. for loop 2. while loop 3. do while loop

ELEMENTS THAT CONTROL A LOOP (PART OF A LOOP)

1. Initialization Expression

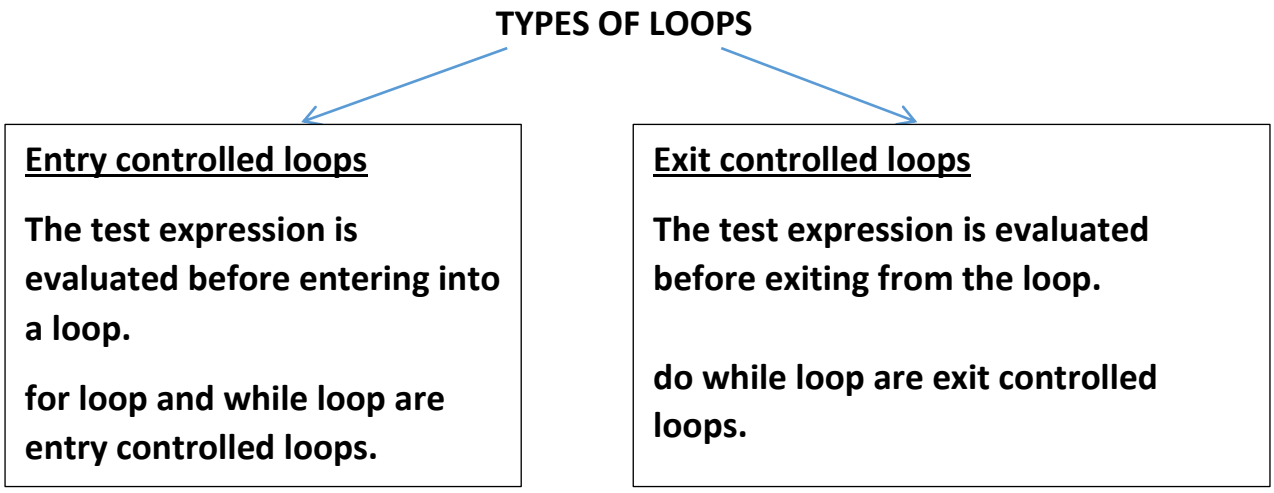
- The initialization expression give the loop variable their first value.

- The initialization expression is executed only once, in the beginning of the loop.

2. Test Expression

- The test expression is an expression whose truth value decides whether the loop body will be executed or not.

- If the test expression evaluates to true the loop body gets executed, otherwise the loop is terminated.



3. Update expression

- The update expression change the value of loop variable.

- The update expression is executed at the end of the loop after the loop body is executed.

4. The Body of the Loop

- The statements that are executed repeatedly form the body of the loop.

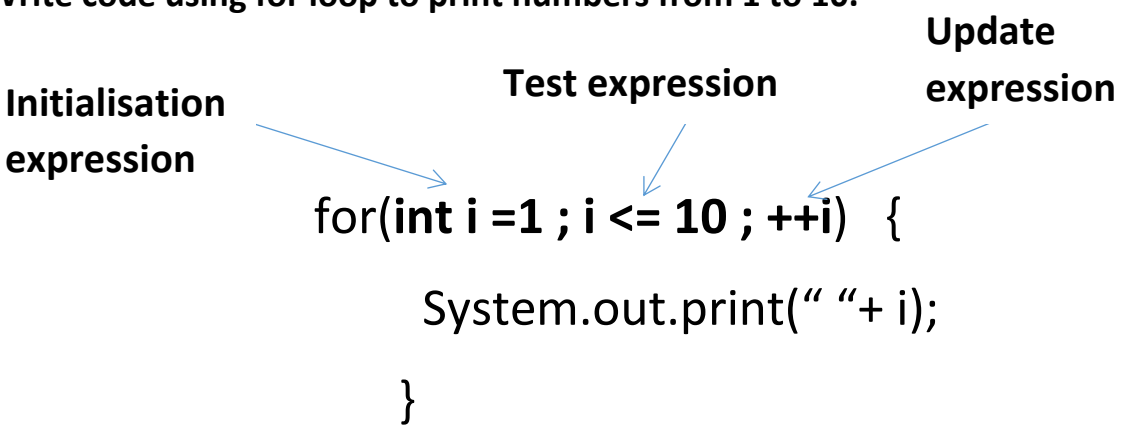
for loop

The Syntax of the for loop is :

for(**initialization expression ; test expression ; update expression**)

body of loop ;

e.g.1. Write code using for loop to print numbers from 1 to 10.



Upon execution, the above code will yield :

1 2 3 4 5 6 7 8 9 10

Explanation

1. First , initialization expression is executed i.e. i = 1 which gives the first value 1 to variable i.

2. Then, the test expression is evaluated i.e., i <= 10 which results into true.

3. Since, the test expression is true, the body of the loop i.e., System.out.println(" "+i) is executed which prints the current value of i on the same line.

4. After executing the loop body, the update expression i.e. ++i is executed which increments the value of i.

5. After the update expression is executed, the test-expression is again evaluated. If it is true, the sequence is repeated from step no 3, otherwise the loop terminates.

e.g.2. Write code using for loop to print even numbers from 1 to 10.

2, 4 , 6 , 8 , 10

```
for(int i =2 ; i <= 10 ; i=i+2)  {
    System.out.print(" "+ i);
}
```

The output will be : 2 4 6 8 10

e.g. 3 Write code using for loop to print odd numbers from 1 to 10.

1 , 3 , 5 , 7 , 10

```
for(int i =1 ; i <= 10 ; i=i+2)  {
    System.out.print(" "+ i);
}
```

The output will be :1 3 5 7 9

***When series to be displayed is ascending series, then <= operator is used in test expression.**

***When series to be displayed is descending series, then >= operator is used in test expression.**

e.g.4. Write code using for loop to print numbers from 10 to 1

```
for(int b =10 ; b >= 1 ; b--)  {
    System.out.print(" "+ b);
}
```

The output will be :10 9 8 7 6 5 4 3 2 1

e.g. 5 Write code using for loop to print the even numbers between 10 and 1.

10 8 6 4 2

```
for(int p =10 ; p >=2 ; p=p-2)  {
        System.out.print(" "+ i);
      }
```

e.g. 5 Write code using for loop to print the odd numbers between 10 and 1.

9 7 5 3 1

```
for(int p =9 ; p >=1 ; p=p-2)  {
        System.out.print(" "+ p);
      }
```

e.g. 6 Write code using for loop to print the sum of series 1+2+3+4+…….+10.

1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10

```
int sum = 0;
for(int a =1 ; a <= 10 ; a=a+1)  {
        sum = sum + a ;
      }
System.out.print(" "+ sum);
```

The output will be : 55

e.g. 7 Write code using for loop to print the sum of EVEN numbers between 1 and 10.

2 + 4 + 6 + 8 + 10

```
int sum = 0;
for(int k =2 ; k <= 10 ; k=k+2)  {
        sum = sum + i ;
    }
  System.out.print(" "+ sum);
```

The output will be : 30

e.g. 8 Write code using for loop to print the sum of ODD numbers between 1 and 10.

1 + 3 + 5 + 7 + 9

```
int sum = 0;
for(int j =1 ; j <= 9 ; j=j+2)  {
        sum = sum + j ;
    }
  System.out.print(" "+ sum);
```

The output will be : 25

e.g. 9. for loop to print factorial of a number n.

1! = 1

2! = 2 * 1 = 2

3! = 3 * 2 * 1 = 6

4!= 4 * 3 * 2 * 1 = 24

5! = 5 * 4 * 3 * 2 * 1 = 120

```
int f=1;
for(int a =n ; a >= 1 ; a=a-1)  {
        f = f * i ;
    }
System.out.print("The factorial is "+ f);
```

e.g.10. for loop to print odd number between 50 to 100.

e.g. 11. for loop to print the sum of following series:

1 + 1/4 + 1/7 + 1/10 + 1/13 + 1/16 + 1/19 + 1/22 + 1/25

Variations in for loop

1. Multiple initialization and Update expression

- A for loop may contain multiple initialization and multiple update expressions. These multiple expression must be separated by commas.

```
int i , sum ;
for(i=1, sum =0 ; i < = n ; sum+=i , i++)
```

Multiple Initialization expression

Multiple Update expression

2. Optional Expression

- In a for loop, ***initialization expression*** , ***test expression*** and ***update expression*** are optional, i.e. you can skip any or all of these expressions.

e.g.

```
int i =1 , sum = 0 ;
    for(    ;  i <=20 ; i= i + 1 )
    System.out.println(" "+i) ;
```

Initialization expression skipped

3. Infinite loop

- An infinite loop can be created by omitting the test expression as shown below :

```
for(j = 25 ;     ; -- j )
    System.out.println("An infinite loop");
```

- Similarly, the following for loop is also an infinite loop.

```
for(        ;          ;          )
        System.out.println("Endless loop");
```

while loop

- The while loop is an entry controlled loop. The syntax of a while loop is:

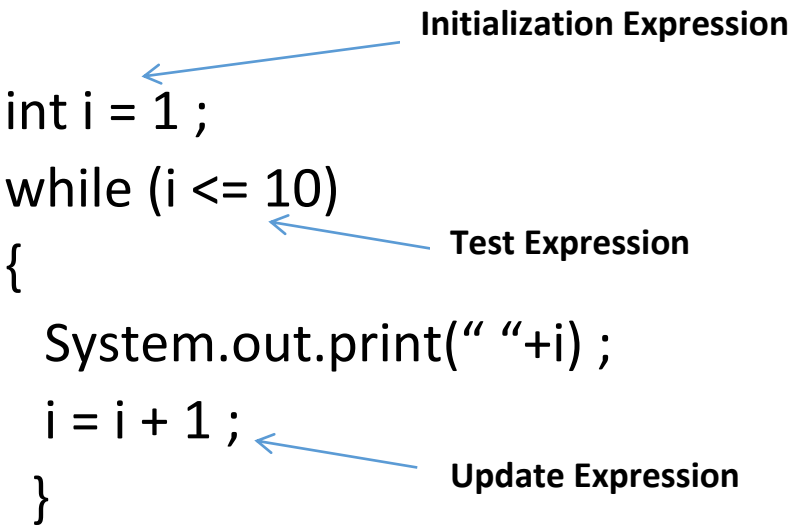
```
initialization expression ;
while(test expression)
{
   loop- body
   Update expression ;
 }
```

where loop body may contain a single or multiple statement.

The loop ***iterates*** while the **expression** evaluates to **true.** When the **expression** becomes **false**, the program control passes to the line after the loop body.

- In a while loop, a ***loop variable*** should be used initialised before the loop begins. The ***loop variable*** should be updated inside the body of while loop.

e.g.1. Code to print series from 1 to 10 using while loop.



e.g.2. Code to print even numbers between 1 to 10 using while loop.

2 4 6 8 10

```
int  i = 2 ;
while ( i <= 10)
{
     System.out.print(" "+i) ;
     i = i + 2 ;
 }
```

e.g.3. Code to print odd numbers between 10 to 1 using while loop.

9 7 5 3 1

```
int  i = 9 ;
while ( i >= 1)
{
     System.out.print(" "+i) ;
     i = i - 1 ;
 }
```

e.g.3. Code to print odd numbers between 1 to 10 using while loop.

1 3 5 7 9

```
int  i = 1 ;
while ( i <= 9)
{
      System.out.print(" "+i) ;
      i = i + 2 ;
 }
```

e.g. 4. Write code using while loop to print the sum of series 1+2+3+4+…….+10.

1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10

```
 int k = 1 , sum =0 ;
while( k <= 10)
 {
    sum = sum + k ;
    k = k + 1 ;
 }
  System.out.print(" "+sum) ;
```

e.g. 5. while loop to print factorial of a number n.

```
 int  i = n , f =1 ;
while( i >= 1)
 {
      f  = f * i  ;
      i = i – 1 ;
   }
  System.out.print("The factorial is "+ f);
```

Variations in while loop

- A while loop can be ***infinite*** if you forget to write the update expression inside its body.

  e.g.

```
int  i = 2 ;
while ( i <= 10)
{
      System.out.print(“ “+i) ;
 }
```

  This loop is not having update expression, it will print 1 infinite number of times.

do while loop

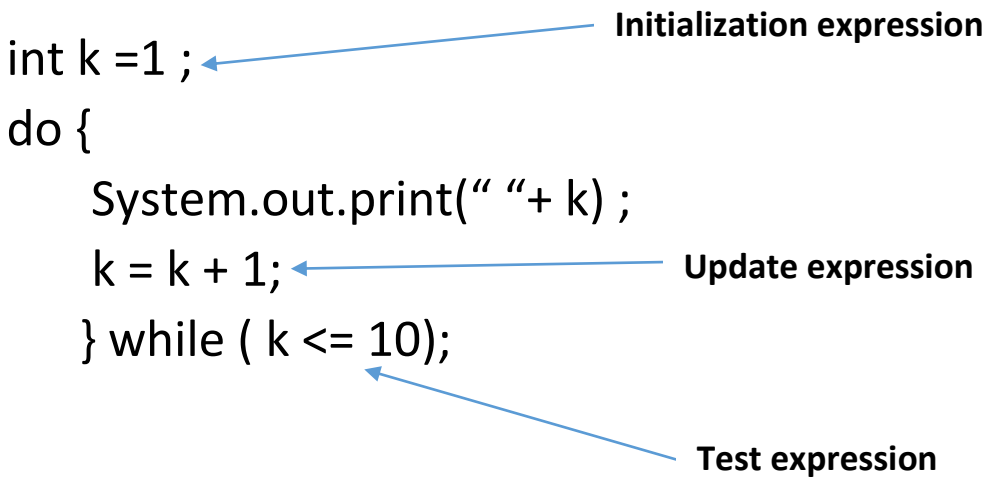
- do while is an ***exit controlled loop*** i.e. it evaluates its **test expression** at the bottom of the loop after executing its loop body statements. This means that a **do while loop always executes at least once.**

- **In do while** loop, the loop body is executed at least once, no matter what the initial state of **test-expression.**

Syntax of do while loop:

```
initialization expression ;
do {
        body of loop
        update expression ;
      } while (test expression) ;
```

e.g.1. Code to print series from 1 to 10 using do while loop.



Upon execution, the above code will yield :

1 2 3 4 5 6 7 8 9 10

e.g. 2 Write code using do while loop to print the sum of EVEN numbers between 1 and 10.

2 + 4 + 6 + 8 + 10

```
int l = 2 , sum =0 ;
do {
    sum = sum + l ;
    l = l + 2 ;
   } while ( l <= 10)
```

e.g. 3. Code to display count from 10 to 0 and then display "HAPPY LOOPING".

e.g. 4. Loop to display the following series upto 10 terms :

10 13.5 17 20.5 .....

e.g 5. Program to calculate and print the sums of even and odd integers of the first n natural numbers using a while loop.

Jump statements

- Jump statements unconditionally transfer program control.
- Java has three jump statements : **return** , **break** , **continue**

break statement

- This statement enables a program to skip over part of the code.
- It can be used with loops (***for, while , do while***), and selection statements (**if , switch**)
- Execution resumes at the statement immediately following the body of the terminated statement.

e.g.

```
for (int i = 2 ; i <= 10 ; i+=2)
 {
    if( i == 4)
          break ;
    else
          System.out.print(“ ”+i);
 }
```

The output of above code is : 2

continue statement

- continue statement also helps in skipping over part of the code. But instead of forcing termination, it forces the next ***iteration*** of the loop to take place, skipping any code in between.

e.g.

```
for (int i = 2 ; i <= 10 ; i+=2)
 {
    if( i == 4)
          continue ;
    else
          System.out.print(“ ”+i);
 }
```

The output of above code is : 2 6 8 10