Chapter 5:

Control Statements in Java

Informatics Practices

Class XII (CBSE Board)

Revised as per CBSE Curriculum 2015

"Open Teaching-Learning Material"

Visit www.ip4you.blogspot.com for more....

Authored By:- Rajesh Kumar Mishra, PGT (Comp.Sc.)
Kendriya Vidyalaya Upper Camp, Dehradun (Uttarakhand)
e-mail : rkmalld@gmail.com

Learning Objectives

In this presentation, you will learn about-

- Introduction to Control Flow
- Control Statements-
- Conditional Statements
  - If… , If…else and Else..if ladder
  - Switch Statement
- Looping Statements
  - For loop
  - While loop
  - do..while loop
- Jumping Statements (break, continue and return)
- Scope of variables
- Developing Java Applications.

Introduction

- In general a program is executed from begin to end. But sometimes it required to execute the program selectively or repeatedly as per defined condition. Such constructs are called control statements.
- The programming constructs in Java, are categorized into -
  - Sequence:

    Statements are executed in top-down sequence.
  - Selection (Conditional/Decision):

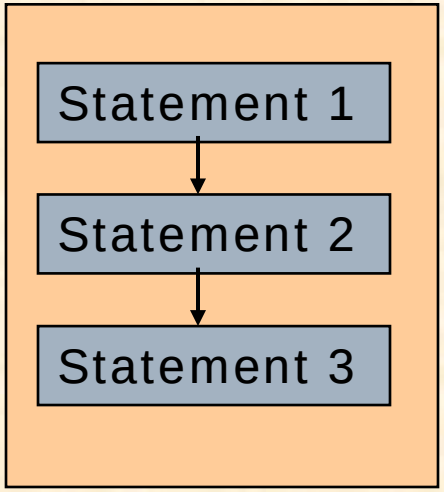
    Execution of statement depends on the condition, whether it is True or False.

    (Ex. if.., if...else, switch constructs)
  - Iteration (Looping):

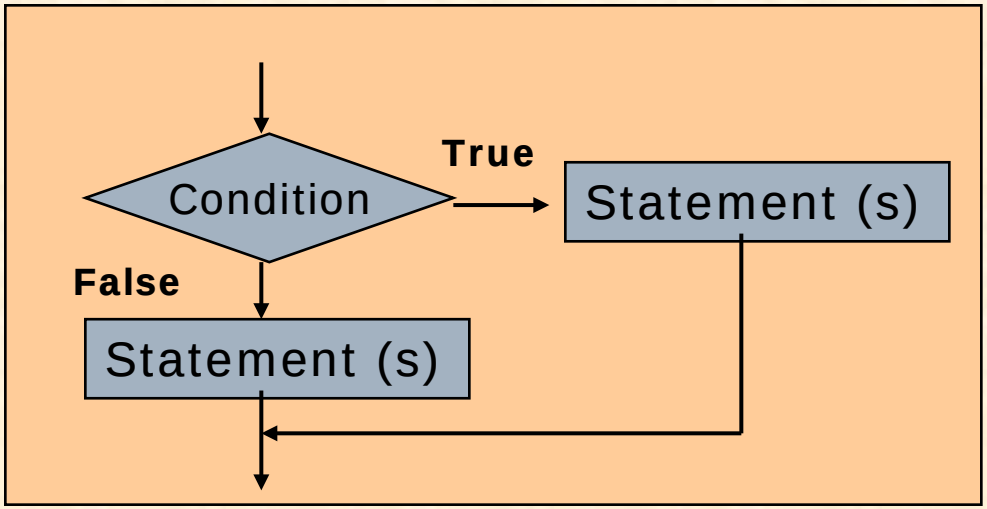
    Statement is executed multiple times (repetition) till the defined condition True or False.

    (Ex. for.. , while... , do..while loop constructs)

Control Statements - Diagrammatic Representation

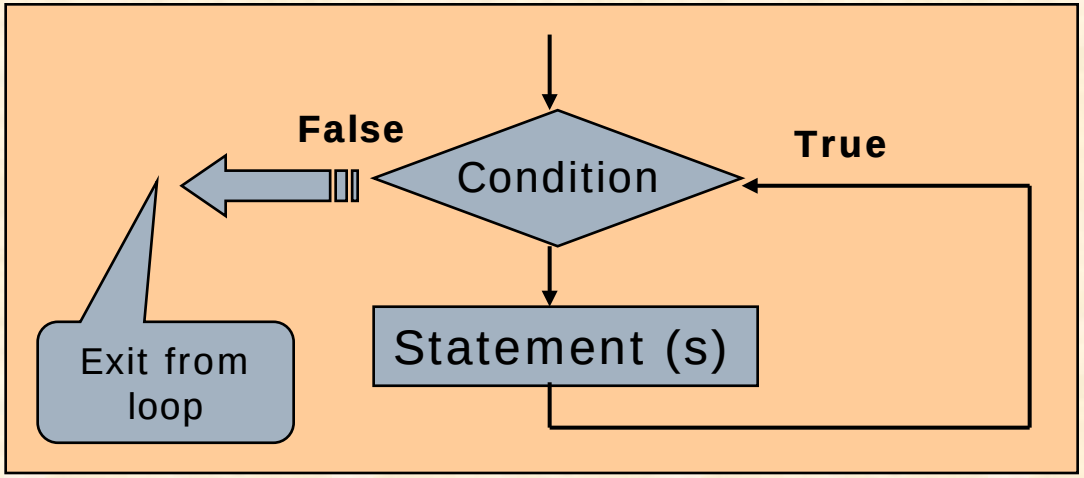


Sequence construct



Selection construct

Iteration Construct



Selection statement (if..)

- The if... statement is used to test a condition. If defined condition is true the statements are executed otherwise they are ignored.
- The condition may be simple or complex (using &&, || etc.) and <u>must be enclosed in</u> ( ).
- Expression defined as condition must be evaluated as True or False.

Syntax

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

```
if ( num>0) {
  jLable1.setText("Number is positive");
 }
```

```
if ( ch>='0' && ch<='9' ) {
  jLable1.setText("It is digit");
 }
```

In case of single statement in if... the use of { } is optional.

Selection statement (if..else..)

- The if...else.. also tests condition. If defined condition is true the statements in True block are executed otherwise else block is executed.

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

```
if ( num>0)  {
  jLable1.setText("Number is positive");
 }
else
{
jLable1.setText("Number is zero or negative");
}
```

```
if ( age>=18)
  jLable1.setText("Eligible to Vote");
else
 jLable1.setText("Not eligible to Vote");
```

In case of single statement { } is optional.

Nested if...

- An if... or if..else.. may have another if.. Statement in its true block or in false block. It is known as Nesting of if (placing an if inside another if).

```
if (condition1){
  if(condition 2)
   { ……..    }
  else
   { ……….    }
}
else{
  if(condition 3)
    { ……..   }
   else
    { ……….   }

  }
```

```
if ( num>0)
{
  jLable1.setText("Number is positive");
 }
else
{ if (num<0)
      jLable1.setText("Number is negative");
   else
      jLable1.setText("Number is zero");
}
```

Nested if.. block

If…else…If ladder

- When a else block contains another if.. and this nested else block may have another if and so on. This nesting is often known as if..else..if ladder or staircase.

```
if (condition1)
   statement 1;
 else
      if (condition 2)
           statement 2;
      else
           if (condition 3)
              statement 3;
           else
               if(condition 4)
                   statement 4;
                ........
                ........
              else
                   statement n;
```

```
if (day==1)
   jLable.setText("Sunday");
else
  if (day==2)
    jLable.setText("Monday");
  else
    if (day==3)
       jLable.setText("Tuesday");
    else
       if(day==4)
          jLable.setText("Wednesday ");
       else
         if(day==5)
           jLable.setText("Thrusday");
         else
           if(day==6)
             jLable.setText("Friday");
           else
             jLable.setText("Saturday");
```

Conditional operator and if.. statement

- The ? : (conditional operator) may be used as alternative to if..else.. statement in Java.
- In contrast to if..., ?: is more concise and compact code but it is less functional than 'if'.
- ?: produces an expression so that a single value or expression may be incorporated, whereas if.. is more flexible, whereas you may use multiple statements, expressions and assignments.
- When ?: is used as nested form, it becomes more complex and difficult to understand.

Syntax

```
Expression 1 ? Expression 2: expression 3;
```

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

```
C = a>b ? a : b ;
```

The switch statement

- Multi-branching selection can be made in Java by using switch statement.
- It tests the value of an expression (short, int, long or char type) and executes associated statements when match is found.

```
switch (<expression>)
{ case <const 1> : statement (s);
                   break;
  case <const 2> : statement (s);
                   break;
  case <const 2> : statement (s);
                   break;
  ..........
  [default : statement (s);]
}
```

```
switch (day)
{ case 1 : Dow="Sunday";
           break;
  case 2 : Dow="Monday";
           break;
  case 3 : Dow="Tuesday";
           break;
  case 4 : Dow="Wednesday";
           break;
  case 5 : Dow="Thursday";
           break;
  case 6 : Dow="Friday";
           break;
  case 7 : Dow="Saturday";
           break;
  default : Dow="Wrong Input";
}
jLable.setText("Weak day"+Dow);
```

1. No two identical constant can be used.
2. Default.. is optional and may be placed anywhere in switch block, if used.

Switch and if..else statement

The switch and if..else both are used to make a selection construct in Java, but there are some differences.

- Switch can test only equality whereas if.. can evaluate any type of relational or logical expression i.e. >,<, <=,>=,==,!=, &&, || etc. can not be used with switch..case statement.
- In switch a single value or constant can be tested but in if.. more versatile expression can be tested.
- The switch statement can handle only byte, short, int or char variable but If.. can test more data type like float, double or string etc.

Conversion of switch & if statement

```
if( grade==‘A’ )
 jTextField1.setText(“Well done”);
else if (grade= = ‘B’ )
 jTextField1.setText(“Nice effort”);
else if( grade== ‘C’ )
 jTextField1.setText(“Keep it up”);
else
 jTextField1.setText(“Try again”);
```

```
switch(grade) {
case ‘A’: jTextField1.setText(“Well done”);
          break;
case ‘B’: jTextField1.setText(“Nice effort”);
          break;
case ‘C’: jTextField1.setText(“Keep it up”);
          break;
default: jTextField1.setText(“Try again”);}
```

Iteration (looping) statements

- Iteration or looping allow a set of instructions to be executed repeatedly until a certain condition is true or false.
- As per placing of condition to be tested, a loop may be Entry-controlled or Exit-controlled loop. In Entry controlled loop, a condition is tested (pre test) before executing the statements. Whereas in Exit-controlled statements are executed first then condition is tested (post test), which ensures at least on time execution of statements.
- As per number of execution, a loop may be Counter-controlled or Sentinel loop. Counter controlled loops are executed fixed number of times, but sentinel loop may be stopped any time by giving its sentinel value. i.e. number of execution can not be forecasted.
- A body of loop contains a block, having statements to be executed repeatedly.

The for .. loop

In simple use, a for.. Loop is Entry-controlled and counter controlled loop having fixed number of iteration.

```
for (initialization exp (s) ; condition ; update exp (s) )
  { ……….
     ……….
  }
```

Looping statements

```
for (int i=1; i<=10 ; i++ ) {
   System.out.println (“”+i);
  }
```

```
//loop to find even nos. up to 50
for (int i=0; i<=50 ; i=i+2)
 System.out.println (“”+i);
```

```
//loop to find factorial
int fact=1,num=5;
for (int i=1; i<=num ; i++)
    fact=fact * i;
System.out.println (“factorial=”+fact);
```

```
//loop to get sum of first 10 nos.
int sum=0;
for (int i=1; i<=10 ; i++ ) {
   sum=sum+ i;
  }
System.out.println (“”+sum);
```

Variations of for.. loop

- Multiple initialization and update expression

  A for.. Loop may contain multiple initialization and/or update expressions separated by (,)

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

- Optional Expressions

  In for loop initialization, condition and update section may be omitted. Note that (;) must be present.

```
for ( ; i<=n ; )
```

- Infinite loop

  For.. Loop may be endless (infinite) when defined condition is always true or it is omitted.

```
for ( ; ; )
```

- Empty loop

  for.. Loop may not contain any statement in looping body i.e. Empty loop. If (;) placed after for.. Loop then empty loop is created.

```
for (int i=1; i<=300 ; i++) ;
```

- Variable declared inside for.. Loop can't accessed outside the loop. Since it is out of scope.

The while.. loop

In simple use, a while .. Loop is Entry-controlled and counter controlled or sentinel looping statement, which executes statements until defined condition is true.

```
while (condition)
  { ……….
    ……….
  }
```

Looping statements

```
//While loop to print 10 numbers
int i=1;
while ( i<=10) {
  i=i+1;
  System.out.println (“”+i);
 }
```

```
//while loop to find factorial of 5
int fact=1,num=5,i=1;
while (i<=num)
 { fact=fact * i;
   i++;
 }
 System.out.println (“factorial=”+fact);
```

A while loop also may be empty or infinite

The do..while loop

Unlike for.. and while.. Loop, do..while loop is Exit-controlled and counter controlled or sentinel looping statement, which executes statements until defined condition is true. It always executes at least once.

```
do
  { ……….
    ……….
  } while (condition);
```

Looping statements

```
// do.. Loop to print A – Z letters
char ch ='A';
do {
   System.out.println (""+i);
   ch++;
} while (ch<='Z');
```

```
//do.. loop to print 1-10 numbers
int i=1;
do
 {System.out.println (""+i);
    i++;
 } while (i<=10);
```

A do…while loop also may be empty or infinite

Nested loop

When a loop (for.. , while.. & do..while) is placed inside another loop, then it is called nesting of loop. In nesting loop, for each cycle of outer loop, the inner loop will be executed completely.

```
for(……)
  { for(……)
      {……
       ……
      }
  }
```

Outer loop

Inner loop

Any combination of looping statement (for, while or do..while) can be nested.

If outer and inner loop is executed m and n time, then total execution will mxn times.

```
for(int i=1; i<=5;i++)
{for (int j=1; j<=i; j++)
  {System.out.print ("*");
   }
   System.out.println();
}
```

```
*
* *
* * *
* * * *
* * * * *
```

Which loop is better ?

- Java offers three looping statements i.e. for.., while.. and do..while. There are some situation where one loop is more appropriate than others.
- The for loop is best suited where number of execution is known in advance. (fixed execution)
- The while and do.. are more suitable in the situations where it is not known that when loop will terminate. (unpredictable times of execution).
- The do.. Loop ensures at least one time execution of the loop, since it is Exit-controlled loop.

Conversion of for & while loop

```
//for loop to print 10 numbers
for (int i=1; i<=10 ; i++ )
{
   System.out.println (“”+i);
 }
```

```
//While loop to print 10 numbers
int i=1;
while ( i<=10)
{ i=i+1;
  System.out.println (“”+i);
 }
```

Jump statements in Java

- Java offers three jump statements (**return**, **break** and **continue**), which transfers the control else where unconditionally.
- The return statement can be used any where in the program. It transfers the control to calling module or Operating System. However Java provides System.exit() method to stop the execution of program.
- The break is used with for.., while, do.. and switch statements which enables the program to skip over some part of the code and places control just after the nearest closing of block. It is used to terminate the loop.
- The continue statement is used within looping statement (not with switch) and works like break i.e. it also skips the statements. Unlike break, it forces the next iteration of loop by skipping the in between code and continues the loop.

Break and Continue the loop

```
While (condition 1)
{ statement 1;
   if (condition 2)
      break;
   .......
  statement 2;
}
Statement 3;
```

```
for (ini;cond;update)
{ statement 1;
   if (condition)
      break;
   .......
  statement 2;
}
Statement 3;
```

```
Do
{ statement 1;
   if (condition)
      break;
   .......
  statement 2;
} While (test condition)
Statement 3;
```

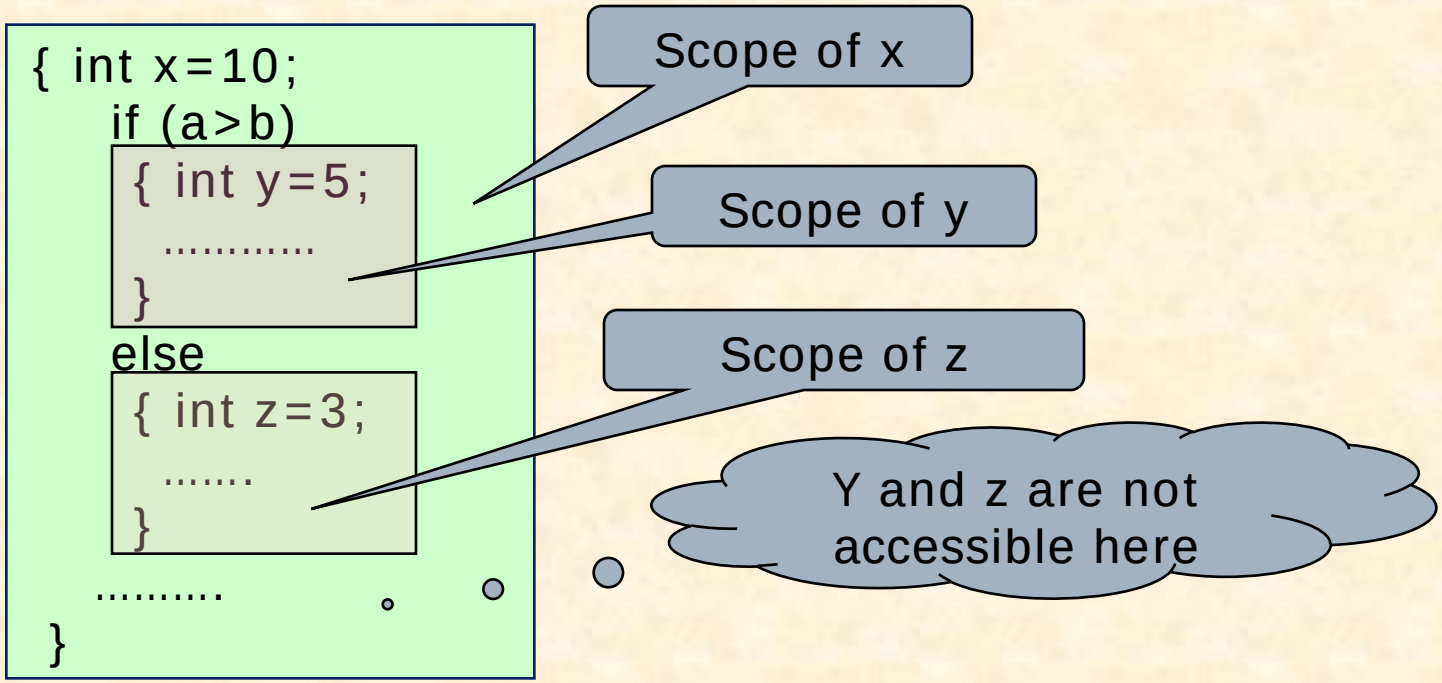
```
While (condition 1)
{ statement 1;
   if (condition 2)
      continue;
   .......
  statement 2;
}
Statement 3;
```

```
for (ini;cond;update)
{ statement 1;
   if (condition)
      continue;
   .......
  statement 2;
}
Statement 3;
```

```
Do
{ statement 1;
   if (condition)
      continue;
   .......
  statement 2;
} While (test condition)
Statement 3;
```

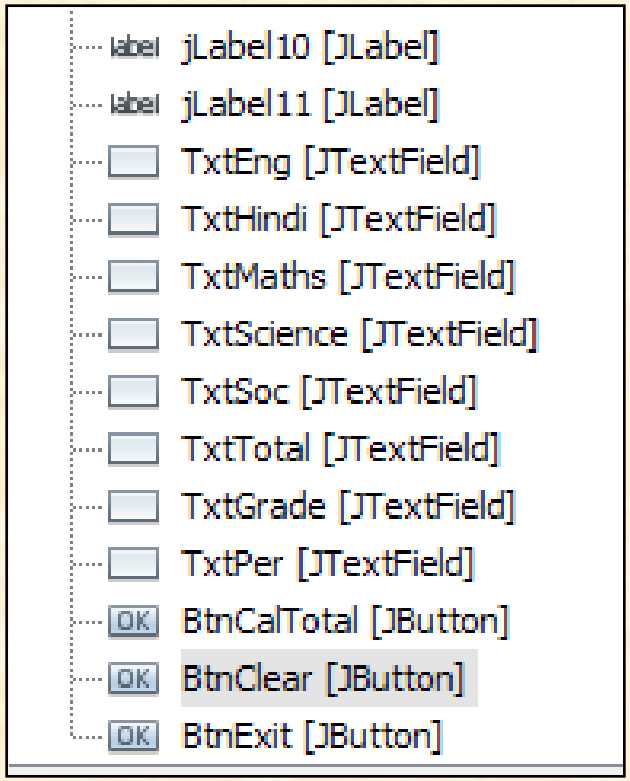
Scope of a variable

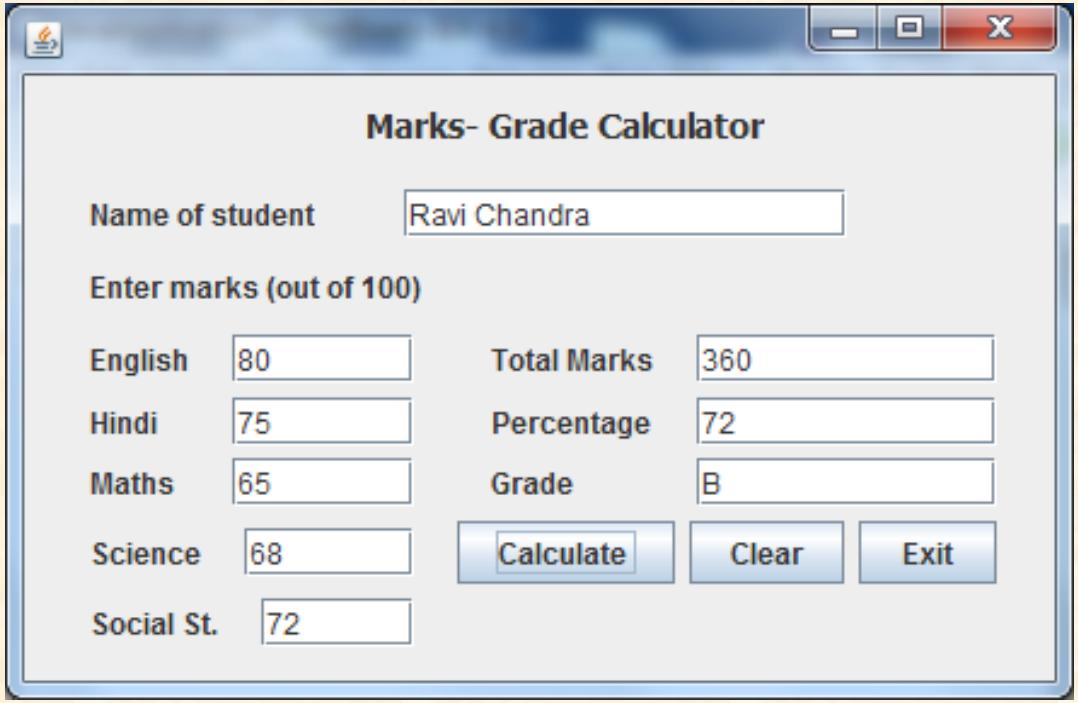
- In Java, a variable can be declared any where in the program but before using them.
- The area of program within which a variable is accessible, known as its scope.
- A variable can be accessed within the block where it is declared.



Demo Application- if..else

Develop a Grade Calculator application, which accepts marks of five subjects and calculate total marks, percentage and grade as per the criteria given.
80% or more - ‘A’ grade , 60 %-79% - ‘B’ Grade, 40%-59% - ‘C’ grade and less than 40% - ‘D’ grade.
The application should generate an error message, if entered marks for any subject is more than 100.





Demo Application- if..else

```
private void BtnCalTotalActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int eng, hin, math,sci, soc, tot, per;
    char grade;
    eng= Integer.parseInt(TxtEng.getText());
    hin= Integer.parseInt(TxtHindi.getText());
    math= Integer.parseInt(TxtMaths.getText());
    sci= Integer.parseInt(TxtScience.getText());
    soc= Integer.parseInt(TxtSoc.getText());
    if(eng>100 || hin>100 || math>100||sci>100||soc>100)
        JOptionPane.showMessageDialog(null,"Enter Mark out of 100");
    else
    {       tot=eng+hin+math+sci+soc;
            per=tot*100/500;
            if(per>=80)
                grade='A';
            else if(per>=60)
                grade='B';
            else if(per>=40)
                grade='C';
            else
                grade='D';
            TxtTotal.setText(""+tot);
            TxtPer.setText(""+per);
            TxtGrade.setText(""+grade);
    }
}
```

Read entered marks and place them on variables for Calculation.

Error message will be generated in a dialog box

Calculate Total, Percentage, Grade and print them accordingly.