5

Input in Java

Learning Scope

Introduction about IPO cycle, Comment in Java programming; Single line comment (//), multiline comment (/*.....*/), documentation comment (/**.....*/), Java Programming by using Function Argument; Java Programming by using Input StreamReader class; Input by using Scanner Class; Import System Package; Creating Scanner Object; Tokens in a Text Line; Testing and Debugging; Types of Errors; Syntax Error; Logical Error; RunTime Error.

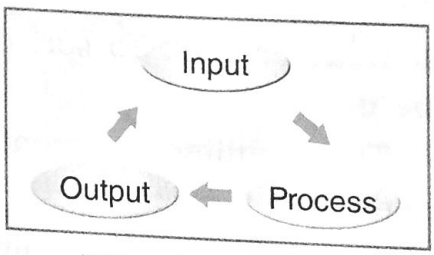
INTRODUCTION

Data processing is an important aspect of computer operation. It includes accepting values from the user, processing data and displaying the result.

Input → Process → Output

Data Processing

The block representation shown above is the cycle of data processing called the *IPO cycle*. Whenever you want to write a program in any computer language, the initial task is to provide values to the computer for processing.



IPO cycle in computer

In this chapter, we are going to discuss the ways of providing values (data) to the computer in Java language.

Before introducing input statements, let us focus on a non-executable statement that enables the users to understand the program logic. It is basically a statement of remark called the *Comment*.

Comment Statements in Java Programming

Sometimes, it becomes difficult for a user to understand the logic used in a program, especially when somebody else has developed it. In such cases, the programmer should mention the purpose and the action being taken in different steps. This can be made possible only by applying comment statements in the program.

There are three ways to give a comment in Java programming. However, users are expected to give a specific comment wherever necessary while writing a Java program. Following are the categories of comment statements:

1. Single line comment (//)
2. Multi line comment (/* ..... */)
3. Documentation comment (/** ..... .*/)

2023/8/12 22:34

Single Line Comment

A single line comment is given in a program when a comment can be expressed in a single line. It may be used as a separate line or in continuation of a statement.

A single line comment starts with (//) followed by the message.

An example of a single line comment is illustrated below:

```
//My first Java program
class Sample
{
public static void main(String args[])
    {
    int a=5;
    int c=a*a; //Square of the number
    System.out.println("Square of the number is"+c);
    }
}
```

Single Line Comment

Multiline Comment

Sometimes, the programmer may need to explain the logic of a program in detail. In such a situation, using single line comments in the program repeatedly is a tedious process. To sort out the problem, a multiline comment statement can be used.

The multiline comment starts with (/*) followed by the comment lines and ends with (*/).

An example is illustrated below:

```
/* My first Java program
It accepts a number (say, 5),
calculates and displays the square of the number */
class Sample
{
public static void main(String args[])
    {
    int a=5;
    int c=a*a;
    System.out.println("Square of 5 is"+c);
    }
}
```

Multiline Comment

Documentation Comment

During the course of programming, the user may need to include some information about himself and his program. This documentation may not be a discussion about the program logic but might serve some information to be viewed by other users who may need to execute your program. The documentation comment starts with (/**) followed by the set of comment lines and ends with (*/). An example is shown below:

```
/**My name is Akanksha Gautam.
 I study in RajendraVidyalaya in Class X B.
 This is my first Java Program.
 */
class test
{
public static void main(String args[])
  {
  int a=5;
  int c=a*a;
  System.out.println("Square of the number is"+c);
  }
}
```

Documentation Comment

Now, we are going to discuss the various ways to input the data values to a program. They are listed as follows:

1. By using a Function Argument (BlueJ system)
2. By using a InputStreamReader Class
3. By using a Scanner Class

1. INPUT BY USING A FUNCTION ARGUMENT

This is one of the methods to accept the value from the user at the time of execution of the program. The variables whose values are to be input must be provided as arguments of the main( ) function.

For example, **public static void main(int a, int b)**

The above function will accept two integer numbers from the user. The output of the program is obtained on the screen after processing. Some of the programs are illustrated below:

Prog. 1 Write a program in Java to find and display the sum and average of three numbers by using function argument.

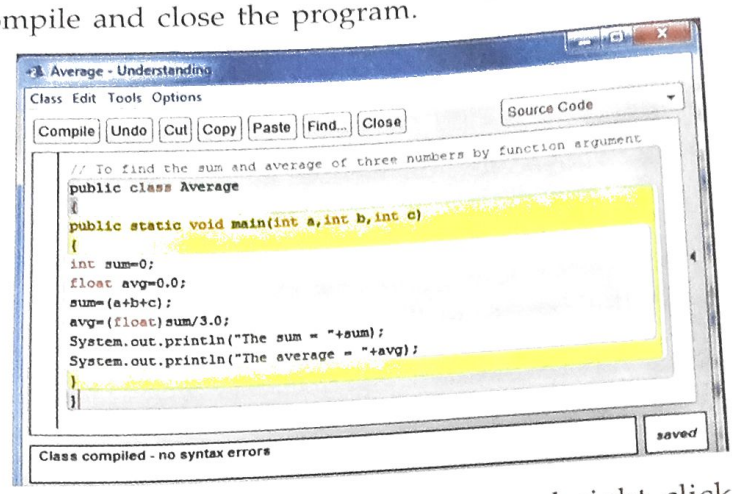
```
// To find the sum and average of three numbers by using function argument
public class Average
{
public static void main (int a, int b, int c)
  {
```

2023/8/12 22:35

```
    int sum=0;
    float avg=0.0;
    sum=(a+b+c);
    avg=sum/3.0;
    System.out.println("The sum ="+sum);
    System.out.println("The average ="+avg);
    }
}
```

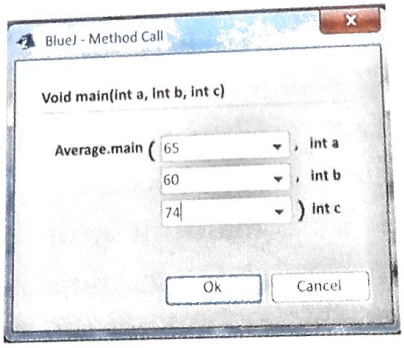
Execution of the program:

Step 1 : Compile and close the program.

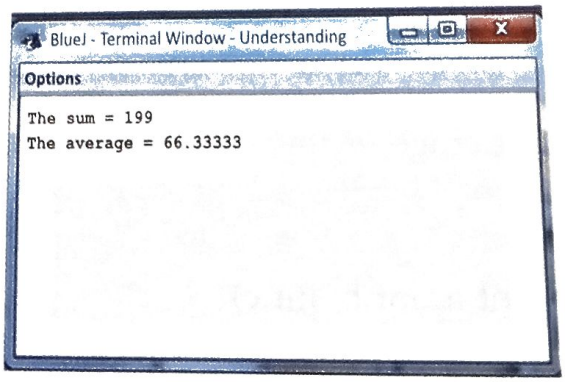


Step 2 : Select the icon of the class Average and right click. A drop down menu will appear. Select void main(int a, int b, int c) from drop down list.

Step 3 : A small Method Call window will appear on the screen. Enter the values of a, b and c one by one and click 'Ok'.



Thus, the desired output of the program is obtained on the output screen.



Prog. 2 Write a program in Java to find and display the simple interest and amount by using the function argument when Principal, Interest and Time are input from the console.

```
// To find the simple interest and the amount by using function argument
public class Interest
{
public static void main(int p, int r, int t)
    {
    double si=0, amt=0;
    si = (p*r*t)/100.0;
    amt=p+si;
    System.out.println("The simple interest ="+si);
    System.out.println("The amount ="+amt);
    }
}
```

Prog. 3 Write a program in Java to find and display the area and perimeter of a right–angled triangle by using function argument. Take perpendicular and base as inputs from the console.

```
// To find the area and perimeter of a right–angled triangle
public class Triangle
{
public static void main(int p, int b)
    {
    double d, ar=0, peri=0;
    d=Math.sqrt(p*p+b*b);
    ar=1.0/2.0*p*b;
    peri=(p+b+d);
    System.out.println("The Area ="+ar);
    System.out.println("The perimeter ="+peri);
    }
}
```

Prog. 4 Write a program to input a three digit number and display all the digits by using arithmetical operators.

Sample Input: 472

Sample Output: 4
7
2

```
// A program to display all the digits of a three digits number
class Digits
{
    public static void main(int num)
    {
        int h,t,u,d;
```

```
        h=num/100;
        d=num/10;
        t=d%10;
        u=num%10;
        System.out.println("The digits are:");
        System.out.println(h);
        System.out.println(t);
        System.out.println(u);
    }
}
```

Input by Using InputStreamReader Class

This system provides user-friendly environment to input the data values in Java programming. The user may have option to change the data during execution of the program. This system uses a combination of two classes, viz. **'InputStreamReader'** and **'BufferedReader'**. The significance of these classes is discussed below:

- **InputStreamReader class**
  The data given to the computer is converted into machine code, i.e. binary form. This machine code needs to be transferred to the CPU in the form of a binary bit pattern. Hence, the system tries to streamline the bits so that they do not get disturbed or hampered and cause any input error. The streamlining of bits is done by using InputStreamReader class.
- **BufferedReader class**
  A computer system has a high speed buffer device (a temporary storage device) in between input unit and processor. In order to synchronise the speeds of I/O devices and processor, buffer allows storing of input streamlined bit pattern so that the instant supply of data can be done to the CPU without any delay. Thus, the BufferedReader class is used.

Process to Input Data by Using InputStreamReader Class

(*i*) Include **java.io** package by using import java.io.*;
This enables java.io package to be used in your program. Basically, this package is imported to avail facility of using input/output functions like readLine() or println().

(*ii*) Create class by using **public class <class name>**
The class name may be any name normally selected and suitable to the operation to be performed in your program.

(*iii*) Declare main function header as:
public static void main(String args[])through IOException
Here,
public: It is the keyword used to specify that the function has no restriction on being used anywhere throughout the system.
static: This keyword states that the function does not require object creation.

void: It refers to the non-returnable type function, i.e. the function will not return any value.

main: It is the function name. The main function is called by default. As soon as the command to execute a program is issued, the control by default reaches to the main function for its execution. This is why the main function is important.

String args[]: This is an args[ ] array that receives the values input from the keyboard as a String data.

Throws IOException: Sometimes, the program experiences I/O errors even if the programming statements have correct syntax. It eliminates the I/O errors (if any) that creep in while programming.

(iv) **InputStreamReader read = new InputStreamReader(System.in);**
This statement creates an object of InputStreamReader type (say, read) to accumulate streamlined bit pattern from the input device.

(v) **BufferedReader in = new BufferedReader(read);**
This statement is used to create an object 'in' of BufferedReader type to store the input data entered from the keyboard temporarily.

Syntax to Accept Numeric Data Values

- To accept an integer up to 32 bits:

```
int n;
System.out.println("Enter a number");
n = Integer.parseInt(in.readLine( ));
```

- To accept a decimal fraction up to 32 bits:

```
float n;
System.out.println("Enter a real number");
n = Float.parseFloat(in.readLine( ));
```

- To accept a decimal fraction up to 64 bits:

```
double n;
System.out.println("Enter a real number");
n = Double.parseDouble(in.readLine( ));
```

Syntax to Accept a Character

```
char ch;
System.out.println("Enter a character");
ch = (char)(in.read( ));
```

Syntax to Accept a String

```
String str;
System.out.println("Enter a String");
str = in.readLine( );
```

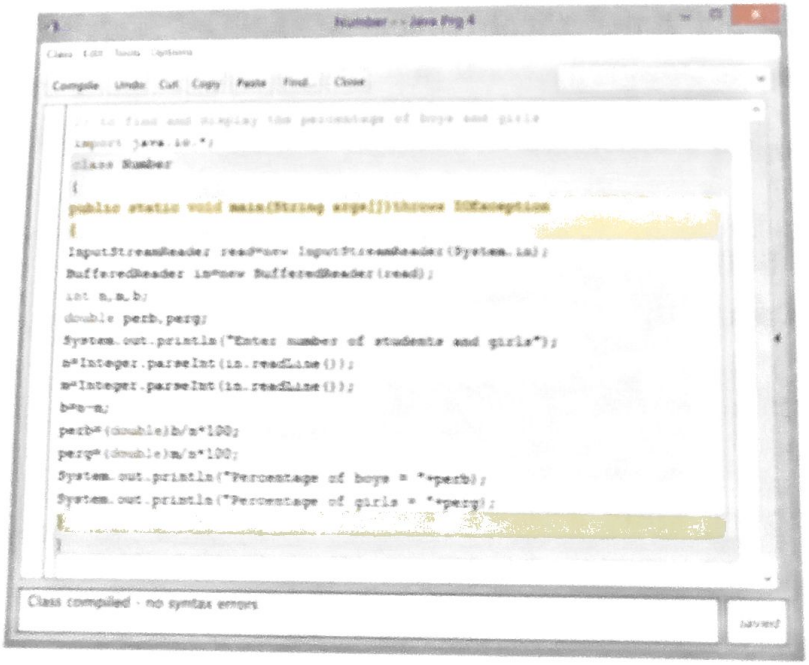
SOLVED PROGRAMS BASED ON INPUT STREAM READER

Prog. 1 In a class of 'n' number of students, the number of girls is 'm'. Write a program to input the values of n and m. Find and display the percentage of boys and girls in the class.

```
// to find and display the percentage of boys and girls
import java.io.*;
class Number
{
public static void main(String args[]) throws IOException
    {
    InputStreamReader read=new InputStreamReader(System.in);
    BufferedReader in=new BufferedReader(read);
    int n,m,b;
    double perb,perg;
    System.out.println("Enter number of students and girls");
    n=Integer.parseInt(in.readLine());
    m=Integer.parseInt(in.readLine());
    b=n-m;
    perb=(double)b/n*100;
    perg=(double)m/n*100;
    System.out.println("Percentage of boys ="+perb);
    System.out.println("Percentage of girls ="+perg);
    }
}
```

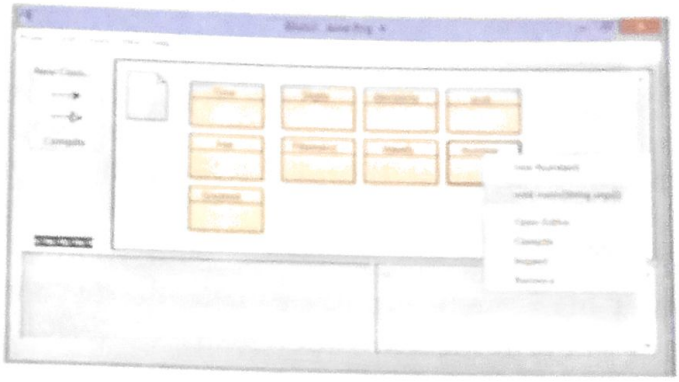
Compilation and Execution

Step 1 : Enter the program on BlueJ environment. Compile the program to make it error free. A 'Class compiled – no syntax errors' message will appear on lower pane of the screen. Click Close button.

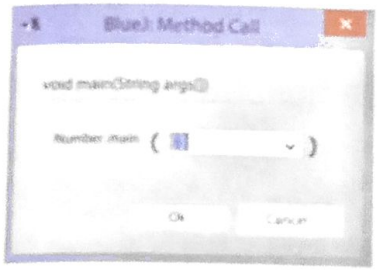


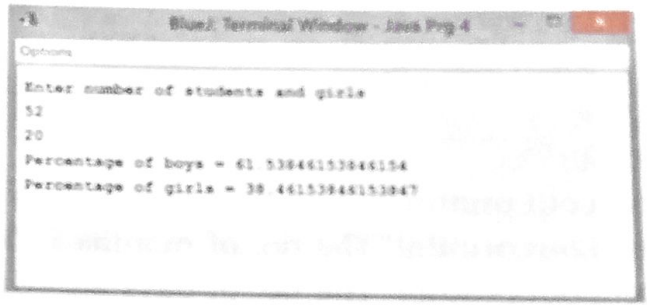
2023/8/12 22:35

Step 2 : Select class icon 'Number' and click right button of the mouse. A drop down menu will appear on the screen. Select and click void main(String args[]) option.



Step 3 : A Method Call window will appear on the screen. Click 'Ok' button.





Step 4 : Enter number of students and number of girls.

Step 5 : Press enter key to get the required output on the screen.

Prog. 2 Write a program in Java to accept the diagonal of a square. Find and display the area and perimeter by using Input stream.

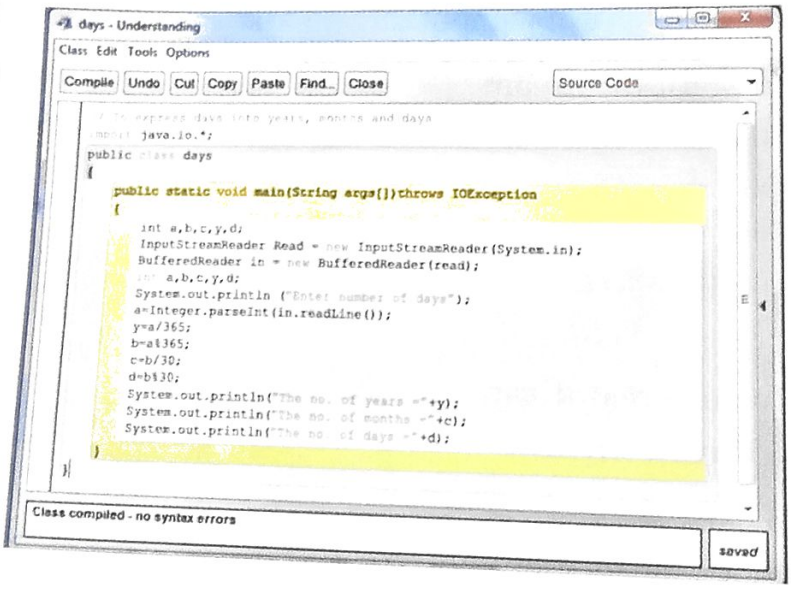
```
// To find the area and perimeter by Input Stream
import java.io.*;
public class Square
{
public static void main(String args[])throws IOException
  {
  InputStreamReader read=new InputStreamReader(System.in);
  BufferedReader in=new BufferedReader(read);
  double a,d,p,ar;
  System.out.println("Enter the diagonal of a square");
  d=Double.parseDouble(in.readLine());
  a=d/Math.sqrt(2);
  ar=a*a;
  p=4*a;
  System.out.println("The area = "+ar);
  System.out.println("The perimeter = "+p);
  }
```

2023/8/12 22:36

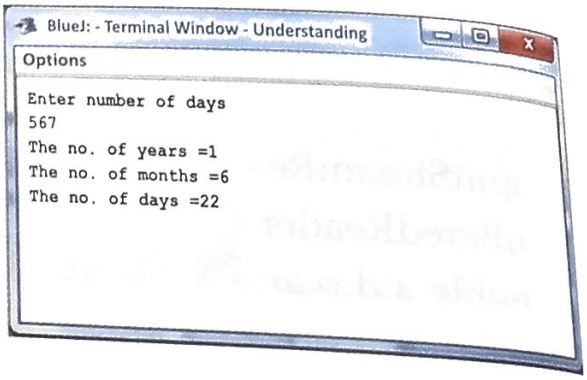
Prog. 3 Write a program in Java to accept the number of days and display it after converting into number of years, months and days.

```
// To express days into years, months and days
import java.io.*;
public class days
{
public static void main(String args[ ])throws IOException
  {
   int a,b,c,y,d;
   InputStreamReader read =new InputStreamReader(System.in);
   BufferedReader in =new BufferedReader(read);
   System.out.println("Enter the number of days");
   a=Integer.parseInt(in.readLine());
   y=a/365;
   b=a%365;
   c=b/30;
   d=b%30;
   System.out.println("The no. of years = "+y);
   System.out.println("The no. of months = "+c);
   System.out.println("The no. of days = "+d);
  }
}
```

Compilation of the Program:



Output of the program:



Prog. 4 An employee contributes 12% of salary to the provident fund. Of the remaining salary, he spends 10% on house rent and 20% on education of the children. Write a program to input salary and calculate:

i. the amount deducted as provident fund

ii. the amount spent on rent and education

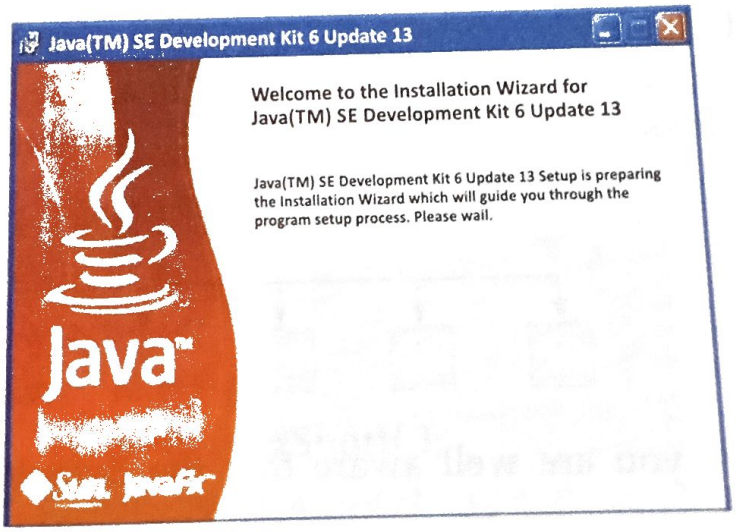
Display the output accordingly.

```
// A program to calculate the amount spent on PF, education and on rent
import java.io.*;
class Fund
{
public static void main(String args[])throws IOException
  {
    InputStreamReader read= new InputStreamReader(System.in);
    BufferedReader in= new BufferedReader(read);
    int sal;
    double pf,rsal,ed,ren;
    System.out.println("Enter salary:");
    sal=Integer.parseInt(in.readLine());
    pf=sal*12.0/100.0;
    rsal=sal-pf;
    ed=rsal*20.0/100.0;
    ren=rsal*10.0/100.0;
    System.out.println("Amount cotributed in PF :"+pf);
    System.out.println("Amount spent in education :"+ed);
    System.out.println("Amount spent on rent:"+ren);
  }
}
```

SCANNER CLASS

Input by Using Scanner Class

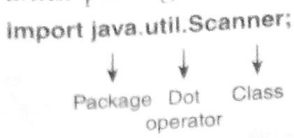
The Scanner Class is the latest development in Java, which allows the user to read values of various types. The Scanner Class is defined within a Java Package (i.e. **java.util.***). There are more methods in class Scanner than you will need in this course. We only cover a small useful subset, which allows you to input the values from either the keyboard or file without going through any conversion system. It also does not require mentioning the limit of values in advance by the programmers. The end of values can be determined by a character or string. However, the Scanner Class necessarily needs JDK 1.5 or more along with BlueJ version 2.5 or more. The methods which are defined under this class don't support the lower JDK and BlueJ version. In order to include Scanner Class in Java programming, you need to install JDK 1.5 and supporting BlueJ version, which is shown below:



Once your system is equipped with the set-up files of JDK and BlueJ, you are now ready to execute Java program by using a Scanner Class. This class is helpful to input different types of numeric and non-numeric data. However, the output statement remains the same i.e. System.out.print( ) or System.out.println( ). The data values are accepted through the object as it happens with InputStreamReader.

Import System Package

The scanner class is available in the system package java.util. You must import java.util package to avail the facilities available in the scanner class. The process to import java.util package is shown below:



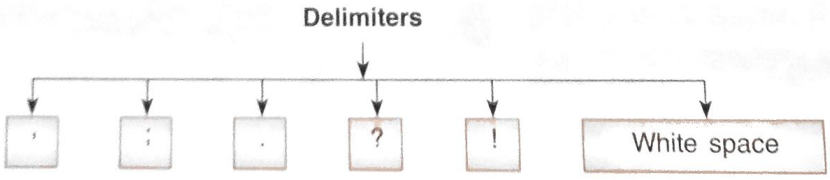
Or, import java.util.*;

Creating Scanner Object

After importing java.util package, you have got the facility to avail scanner class and create object which can hold a set of values of different types. The syntax to create a scanner object is shown below:

- Declare an object of type Scanner say **'sc'** (You can use any object name of your choice).
- **new** keyword is used to have dynamic allocation of scanner object.
- The scanner class has a special constructor which is automatically invoked to store values in the scanner object from console.
- The parameter **'System.in'** allows the constructor to receive data from the keyboard.

The Scanner Class basically works on the principle of 'Tokens' in the input. A token is a series of characters that ends with a delimiter. Some delimiters which can be used as token separator are comma (,), full stop (.), semi-column (;), whitespace, etc. Scanner class uses white space as default token separator. A whitespace character can be a blank, a tab character, a carriage return, or the end of the file. Thus, if we read a line that has a series of numbers separated by blanks, the scanner will take each number as a separate token.



Now, you are well aware that the default delimiter is referred to be white

spaces but other delimiters can also be set as the scanner object in order to separate the tokens. The process is shown below:

Syntax: **<Scanner Object> . useDelimiter (delimiter);**

For example, sc.useDelimiter (",");

Advantages of Input through Scanner Class

- User may not mention set of data being input from the console. The end of data elements can be determined through a special token.
- InputStreamReader and BufferedReader need not to be mentioned.
- Program logic is simple because Scanner class has various methods to manipulate input data.
- String manipulation is quite easy as each word can be obtained as a token and handled separately.
- It provides an easier approach to read records from a data file.

Tokens in a Text Line

Sample Input: Understanding Computer Applications for Class 9

With reference to earlier discussion, the different tokens are:

Understanding
Computer
Applications
for
Class
9

Thus, a Scanner Class accepts the data in terms of 'Tokens'. These tokens are considered to be string types and are manipulated through various methods available in the Scanner class.

Now, we are going to discuss the methods which are used to accept the numeric values from a set of values input to the Scanner object.

The numeric values may be on one line with blanks between each value or may be on separate lines, the ***whitespace*** characters (blanks or carriage returns) act as default token separators. The next method returns the next token as a string, regardless of what is typed in. If any token has not been extracted from Scanner object then first token itself is be treated to be the next token.

Methods to Read Tokens from Scanner Object

- **nextShort( )**

  It receives the next token from scanner object as a short type value and stores in short type variable.

  Syntax:

  short <variable> = <scanner Object>.nextShort( );

  e.g.,

  Scanner in = new Scanner(System.in);

  short p=in.nextShort(); Or, **short p;**
  p=in.nextShort( );

  It means that the variable p will store a short type data through the Scanner object.

2023/8/12 22:36

- **nextInt();**
  It receives the next token from the Scanner object which can be expressed as an integer and stored in integer type variable.
  Syntax:
  int <variable> = <Scanner Object>.nextInt( );
  For example,
  Scanner in = new Scanner(System.in);
  int p= in.nextInt();
  It means that the variable p will store an integer value through the Scanner object.
- **nextFloat();**
  It receives the next token which can be expressed as a floating type value and stores it into floating type variable.
  Syntax:
  float <variable> = <Scanner Object>.nextFloat( );
  For example,
  Scanner in = new Scanner(System.in);
  float p= in.nextFloat();
  It means that the variable p will store a float type value through the Scanner object.
- **nextLong();**
  This method receives the next token expressed as a long type value and stores it in a long type variable.
  Syntax:
  long <variable> = <Scanner Object>.nextLong( );
  For example,
  Scanner in = new Scanner(System.in);
  long p= in.nextLong();
  It means that the variable p will store a long type value through the Scanner object.

At a Glance

Data types (Integer) in Java

Type	Size in bytes	Size in bits
byte	1 byte	8 bits
short	2 bytes	16 bits
int	4 bytes	32 bits
long	8 bytes	64 bits

- **nextDouble();**
  This method receives the next token which can be expressed as a double type value and stores it as a double type variable.
  Syntax:
  double <variable> = <Scanner Object>.nextDouble( );
  For example,
  Scanner in = new Scanner(System.in);
  double p= in.nextDouble();
  It means the variable p will store a double type value through the Scanner object.
- **next();**
  This method accepts the next token from the Scanner object as a string. A token can be defined by a set of characters between two simultaneous white spaces or delimiters.

2023/8/12 22:36

ch can be expressed as

through the Scanner

floating type value

rough the Scanner

At a Glance

(Integer) in Java

ze in tes	Size in bits
yte	8 bits
ytes	16 bits
tes	32 bits
tes	64 bits

a double type

anner object.

a string. A
neous white

Syntax:
String <variable> = <Scanner Object>.next( );
For example,
Scanner in = new Scanner(System.in);
String str1= in.next();
It means that the variable str1 will store the next token as a string through the Scanner object.

- **next.charAt(0);**
  Java does not have a specific function to accept a character through a Scanner class. The next() function is itself modified in order to accept a character. The next() function receives a character as a string. The character can be extracted by using charAt(0) function to be stored in the resulting variable.
  Syntax: char<variable> = <Scanner object>.next().charAt(0)
  For example,
  Scanner in = new Scanner(System.in);
  char c=in.next().charAt(0);
  It means the variable c will store a character through the Scanner object.
- **nextLine();**
  The application of next( ) method is to accept a token as a string from the Scanner object. When a line of text is input into the scanner object, the next( ) method will receive the first token. Rest part of the text line can be obtained as a token by using nextLine( ) method. In case, next( ) method is not used before, the nextLines( ) method results in entire line of text.
  Syntax:
  String <variable> = <Scanner Object>.nextLine( );
  For example,
  Scanner in = new Scanner(System.in);
  String str2= in.nextLine();
  It means that the variable str2 will store a line of text through the Scanner object.

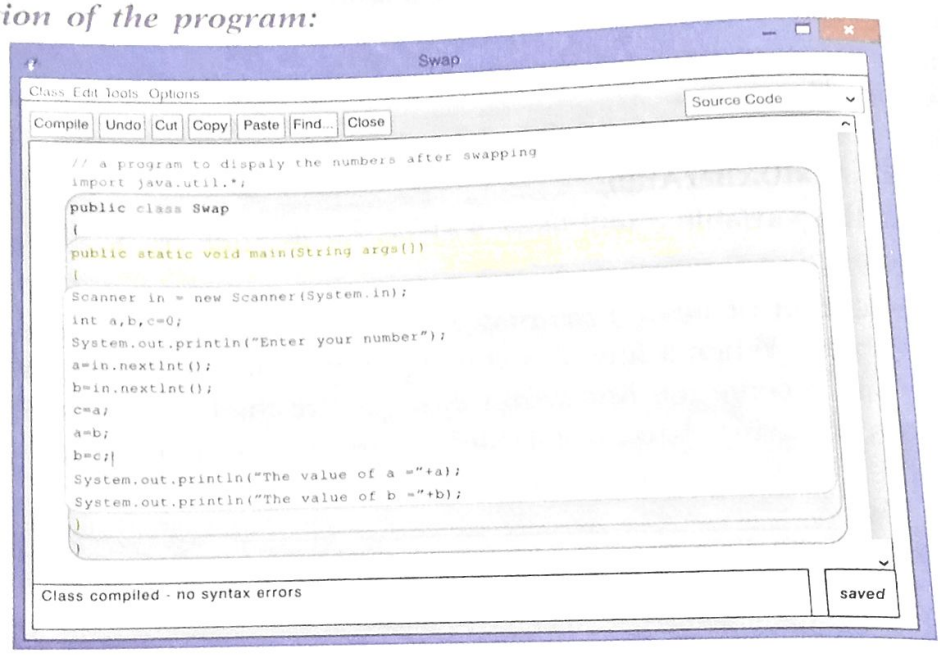
SOLVED PROGRAMS BY USING SCANNER CLASS

Prog. 1 Write a program in Java to input two numbers. Display the numbers after swapping them by using a third variable.
Sample Input: a= 95, b=45
Sample Output: a=45, b=95

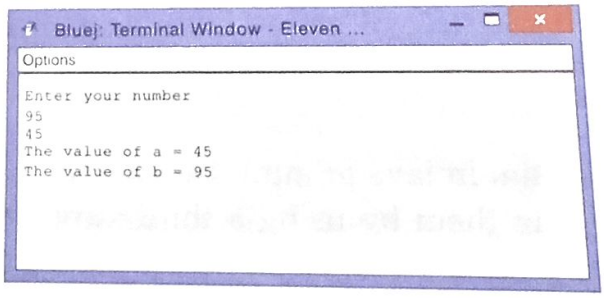
```
// a program to display the numbers after swapping
import java.util.*;
public class Swap
{
public static void main(String args[])
  {
  Scanner in = new Scanner(System.in);
  int a,b,c=0;
```

```
    System.out.println("Enter your number");
    a=in.nextInt();
    b=in.nextInt();
    c=a;
    a=b;
    b=c;
    System.out.println("The value of a="+a);
    System.out.println("The value of b="+b);
    }
}
```

Compilation of the program:



Output of the Program:



Prog. 2 Write a program in Java to input the name and basic salary of an employee. Calculate and display the gross salary and net salary when:

da= 30% of basic

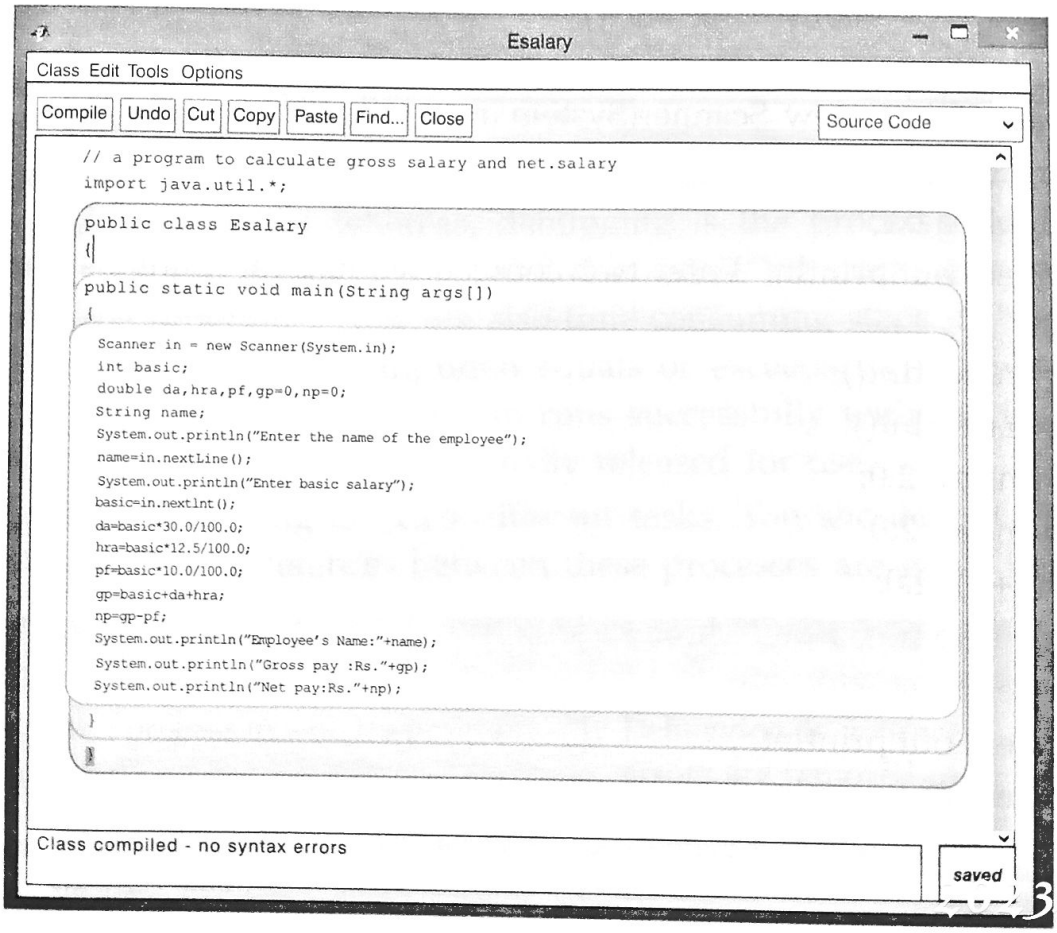
hra= 12.5% of basic

pf= 10% of basic

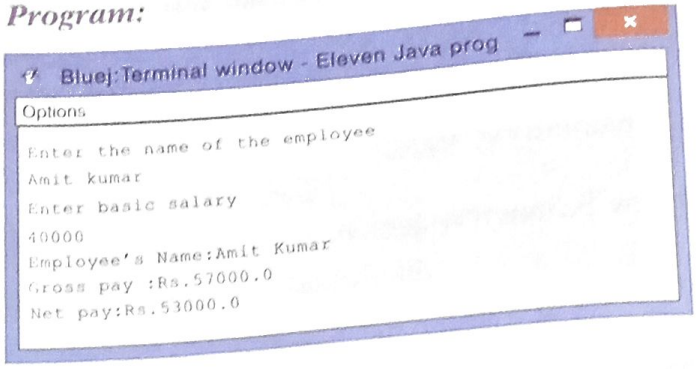
gross= basic + da + hra

net pay = gross - pf

```
// A program to calculate gross salary and net salary
import java.util.*;
public class Esalary
{
public static void main(String args[])
   {
   Scanner in = new Scanner(System.in);
   int basic;
   double da,hra,pf,gp=0,np=0;
   String name;
   System.out.println("Enter the name of the employee");
   name=in.nextLine();
   System.out.println("Enter basic salary");
   basic=in.nextInt();
   da=basic*30.0/100.0;
   hra=basic*12.5/100.0;
   pf=basic*10.0/100.0;
   gp=basic+da+hra;
   np=gp-pf;
   System.out.println("Employee's Name:"+name);
   System.out.println("Gross pay :Rs."+gp);
   System.out.println("Net pay:Rs."+np);
   }
}
```



Output of the Program:



Prog. 3 Write a program to display three rational numbers between any two consecutive natural numbers, taking inputs from the console.

Sample Input: 7 and 8

Sample Output: 7.25
7.5
7.75

[**Hint:** Rational number between m and n can be calculated as $\frac{m+n}{2}$]

```
// A program to calculate and rational numbers between two natural numbers
import java.util.*;
class Rational_Num
{
public static void main(String args[])
  {
      Scanner in = new Scanner(System.in);
      int m,n;
      double a,b,c;
      System.out.println("Enter two consecutive natural numbers one by
      one:");
      m=in.nextInt();
      n=in.nextInt();
      a=(m+n)/2.0;
      b=(m+a)/2.0;
      c=(a+n)/2.0;
      System.out.println("Three rational numbers are :");
      System.out.println(b);
      System.out.println(a);
      System.out.println(c);
  }
}
```


2023/8/12 22:37

Prog. 4 A two digit number can be obtained by using the expression (10a+b), where a and b are the ten's and unit's digits respectively. The number after reversing the digits will be (10b + a). Write a program to input two digits of a number (a and b) and display the result when the sum of number and the reversed number is divided by the difference between its digits.

Sample Input: 62

Sample Output: = 22

```
// A program to display result
import java.util.*;
class Result
{
public static void main(String args[])
   {
      Scanner in = new Scanner(System.in);
      int a,b,res;
      System.out.println("Enter tens digit:");
      a=in.nextInt();
      System.out.println("Enter unit digit:");
      b=in.nextInt();
      res=((10*a+b)+(10*b+a))/(a-b);
      System.out.println("The answer :"+res);
   }
}
```

TESTING AND DEBUGGING

In general, testing is the process of making sure that the program performs the intended task or not whereas, debugging is the process of locating and eliminating program errors. Testing and debugging are vital steps of removing errors from the programs. They are also time consuming steps. In fact, the time spent in testing and debugging often equals or exceeds the time spent in the creation of the program. If a program runs successfully with the test data and produces the correct result, it is normally released for use.

Testing and debugging are two different tasks. You should not be confused between them. The differences between these processes are given below:

Testing	Debugging
1. Testing is a process in which a program is validated.	1. Debugging is a process in which the errors are removed from the program.
2. Testing is complete when all desired verifications against specifications have been performed.	2. Debugging is finished when there are no errors and the program is ready for execution.

Types of Errors

When you run a program then you have to be very careful while giving inputs. It may happen that you may not get the desired result due to incorrect input of data and thus get incorrect output. Generally, there are three types of errors that occur in a computer program, i.e.:

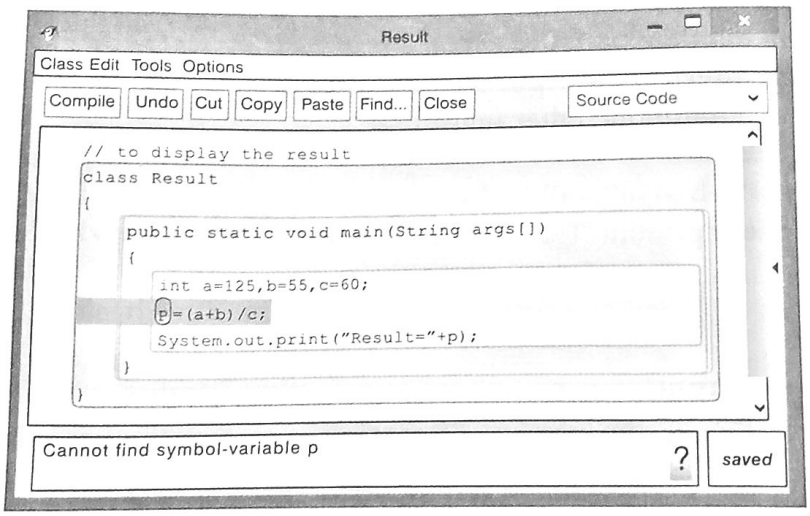
1. Syntax error
2. Logical error
3. Runtime error

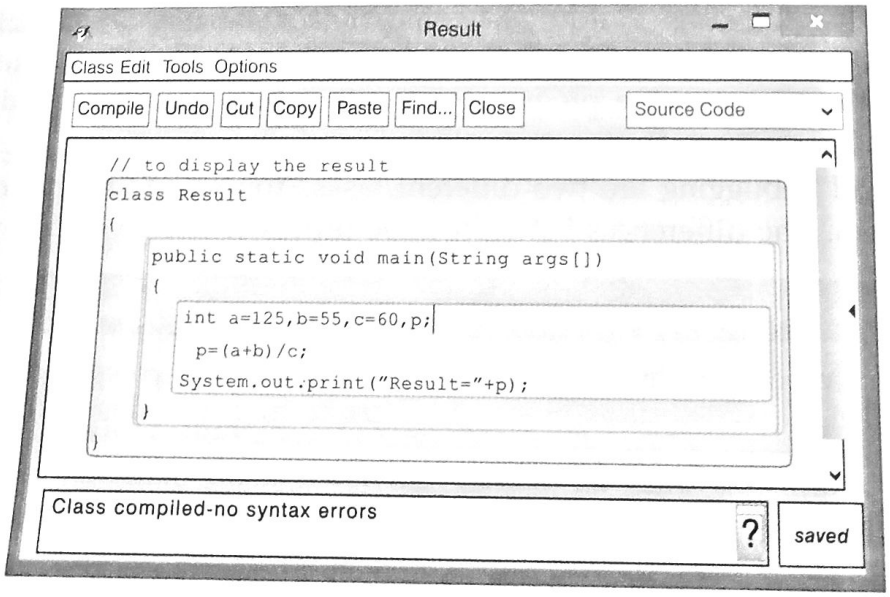
Syntax error

These errors occur when the rules or the grammar of the programming language are not followed. Such programming errors typically involve incorrect punctuation, incorrect word sequences, undefined terms or erroneous use of terms.

Example:

A sample program which shows the syntax error at the time of compilation.





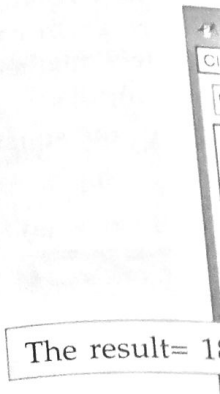
Here, you can notice tha
shows a syntax error (can
data type of the variable '
Thus, the program is er

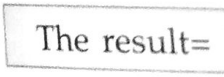
Logical error

A logical error is concern
computer actually does n
the instructions and exe
The problem is that the
Hence, a logical error is

Example:

Write a Java progran
numbers by the third





Here, you can notice that the variable p is not declared. Thus, the program shows a syntax error (cannot find symbol-variable p) and after declaring the data type of the variable 'p', it shows 'No syntax error'.

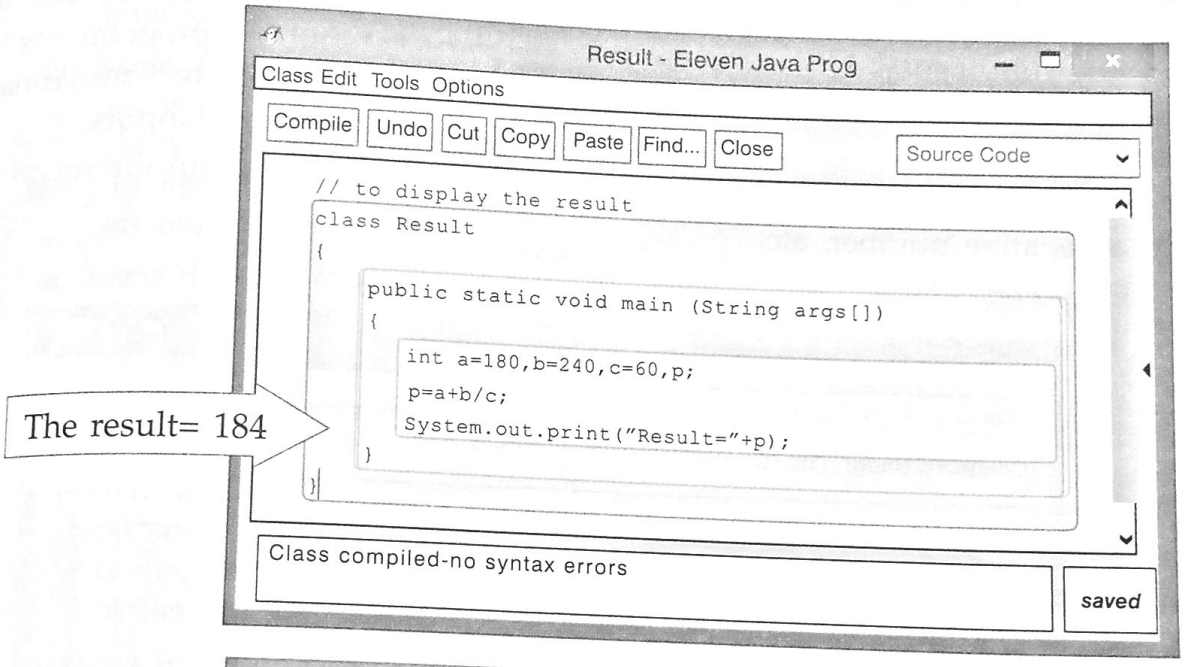
Thus, the program is error free and ready to execute.

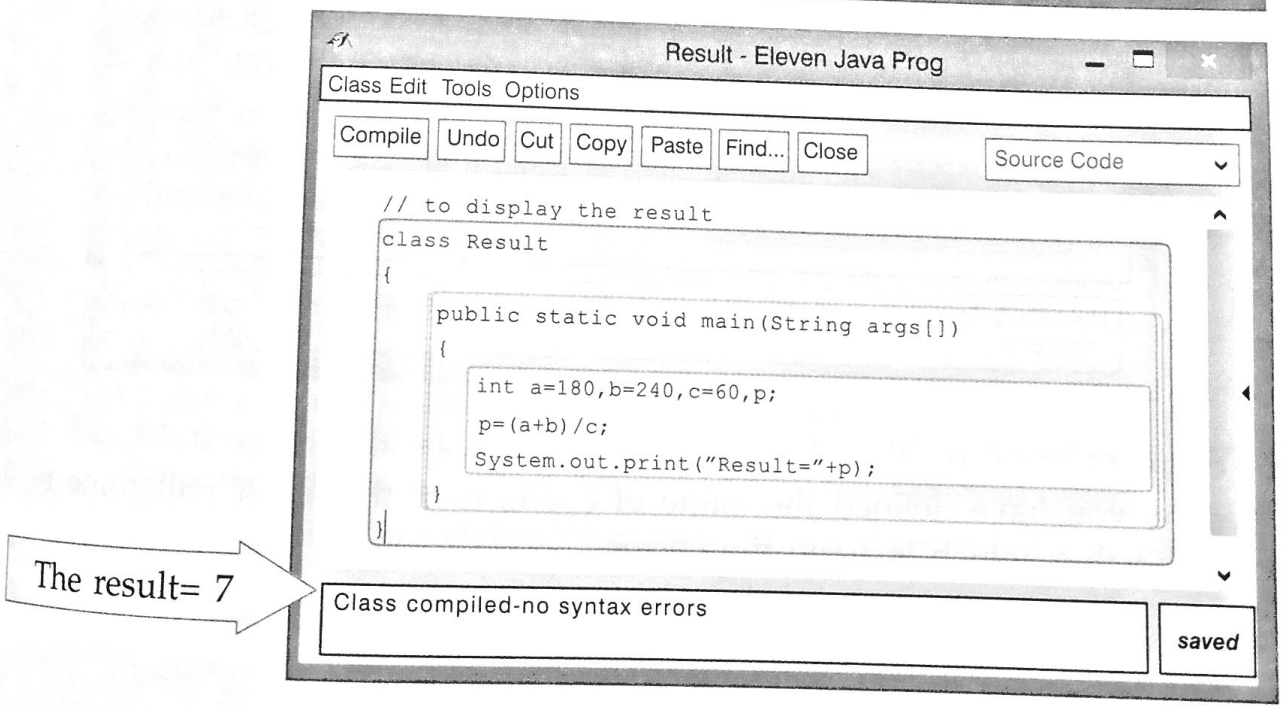
Logical error

A logical error is concerned with an error in planning the program's logic. The computer actually does not know that an error has been made. It simply follows the instructions and executes the results, but the output may not be correct. The problem is that the logic being given does not produce the desired results. Hence, a logical error is the error produced due to wrong logic in the program.

Example:

Write a Java program to take three numbers. Divide the sum of first two numbers by the third and display the result.





Here, you can notice that the result in both the cases are different. However, the program is compiled. It is because in first case, the user has instructed to add a to the result of $\frac{b}{c}$, which yielded incorrect result according to the given condition.

Differences between Syntax error and Logical error

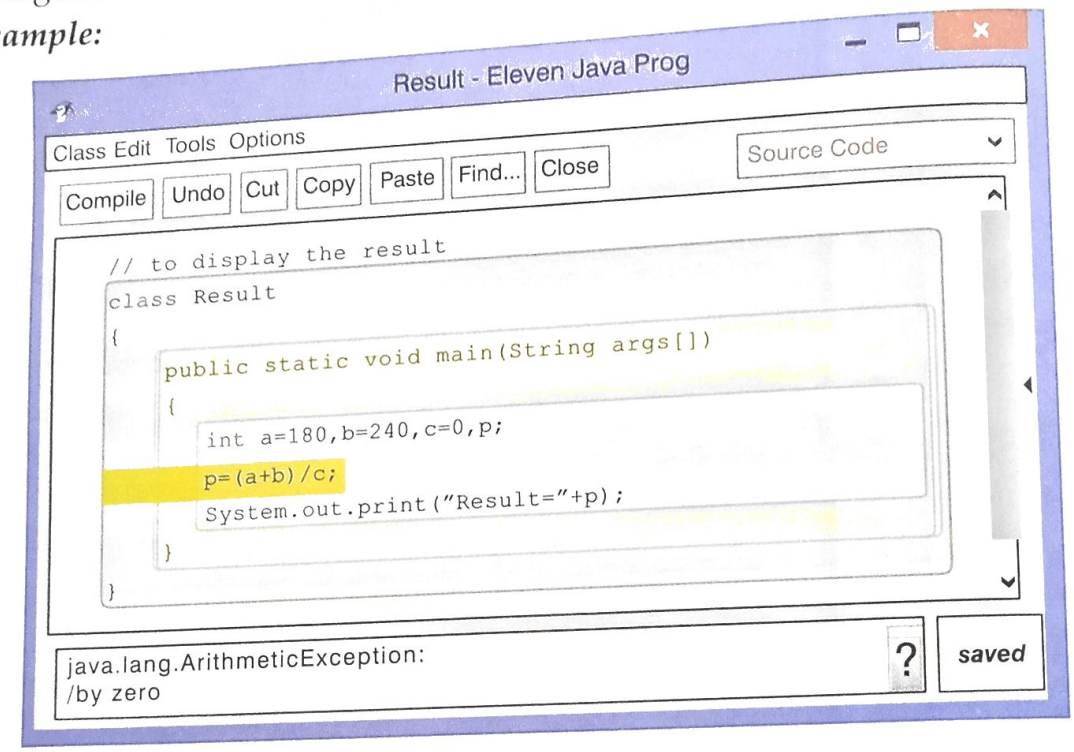
Syntax error	Logical error
1. It is the grammatical error in the statements.	1. It is the error in programming logic.
2. The program is unable to give the output.	2. The program doesn't give the desired output.

Run Time error

Sometimes, you find the program is not producing the desired results, even after errors (syntax as well as logical) are removed and the program has been compiled successfully. These errors may occur due to incorrect mathematical operations or by entering incorrect data values at the time of inputs.

For example: dividing a number by zero or trying to find out the square root of a negative number, etc.

Example:



The expression p = (a + b)/c;

Here, you have defined the value of c = 0, then the result will come to an infinite value, which is a run time error.