INTRODUCTION TO JAVA

Activity Book

NIIT

Introduction to Java

Activity Book

Introduction to Java_AB/18-M03-V1.0

COURSE DESIGN-LAB GUIDE

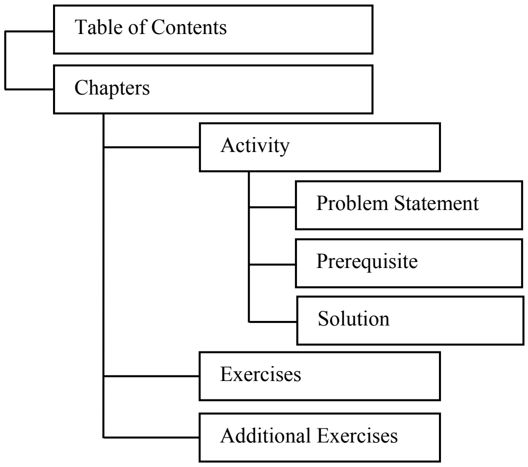


Table of Contents

Chapter 1 – Overview of Java

Activity 1.1: Creating and Executing a Class ---------- 1.2
Problem Statement ---------- 1.2
Solution ---------- 1.2
Exercises ---------- 1.4
Exercise 1---------- 1.4
Exercise 2---------- 1.4
Exercise 3---------- 1.5
Exercise 4---------- 1.5
Exercise 5---------- 1.5
Additional Exercises ---------- 1.6
Exercise 1---------- 1.6
Exercise 2---------- 1.6

Chapter 2 – Implementing Operators

Activity 2.1: Working with Operator---------- 2.2
Problem Statement ---------- 2.2
Solution ---------- 2.2
Exercises ---------- 2.4
Exercise 1---------- 2.4
Exercise 2---------- 2.4
Exercise 3---------- 2.4
Exercise 4---------- 2.4
Exercise 5---------- 2.4
Additional Exercises ---------- 2.6
Exercise 1---------- 2.6
Exercise 2---------- 2.6

Chapter 3 – Working with Conditional and Loop Constructs

Activity 3.1: Working with Conditional Constructs ------------------------------3.2
Problem Statement--3.2
Prerequisite ---3.2
Solution --3.2
Activity 3.2: Working with Loop Constructs--------------------------------------3.4
Problem Statement--3.4
Prerequisite ---3.4
Solution --3.4
Exercises---3.6
Exercise 1 ---3.6
Exercise 2 ---3.6
Exercise 3 ---3.6
Exercise 4 ---3.6
Exercise 5 ---3.7
Additional Exercises---3.8
Exercise 1 ---3.8

Chapter 4 – Working with Arrays, Enums, and Strings

Activity 4.1: Manipulating Arrays and Strings --------------------------------4.2
Problem Statement--4.2
Prerequisite ---4.2
Solution --4.2
Exercises---4.5
Exercise 1 ---4.5
Additional Exercises---4.6
Exercise 1 ---4.6

Chapter 5 – Implementing Inheritance and Polymorphism

Activity 5.1: Implementing Inheritance and Polymorphism --------------------5.2
Problem Statement--5.2
Solution --5.2
Exercises---5.6
Exercise 1 ---5.6
Exercise 2 ---5.6

Additional Exercises -- 5.7
Exercise 1-- 5.7

Chapter 6 – Handling Errors

Activity 6.1: Exception Handling-- 6.2
Problem Statement -- 6.2
Prerequisite--- 6.2
Solution -- 6.2
Activity 6.2: Implementing Assertions --- 6.6
Problem Statement -- 6.6
Solution -- 6.6
Exercises -- 6.9
Exercise 1-- 6.9
Exercise 2-- 6.9
Exercise 3-- 6.9

Chapter 7 – Working with Regular Expressions

Activity 7.1: Processing Strings Using Regex ------------------------------------- 7.2
Problem Statement -- 7.2
Prerequisite--- 7.2
Solution -- 7.2
Exercises -- 7.4
Exercise 1-- 7.4
Additional Exercises -- 7.5
Exercise 1-- 7.5

Chapter 8 – Working with Streams

Activity 8.1: Working with Input Stream and Output Stream ---------------- 8.2
Problem Statement -- 8.2
Prerequisite--- 8.2
Solution -- 8.2
Exercises -- 8.6
Exercise 1-- 8.6
Exercise 2-- 8.6
Additional Exercises -- 8.7
Exercise 1-- 8.7

NIIT

Overview of Java

CHAPTER 1

Activity 1.1: Creating and Executing a Class

Problem Statement

ProGame Inc. has recently introduced a new department that develops games. For this, the organization has hired junior programmers. They have been assigned a project to develop the HangmanGame application in Java. They need to create the class structure for the Hangman game. In addition, they need to write code to display a menu that a user will use to play the game, view the game instructions, or exit the game. Further, the menu should also accept the input from the player. Help the programmers to do the needful.

Solution

To achieve the preceding requirements, the programmers need to perform the following steps:

1. Create a Java application, **HangmanGame**.
2. Create a package, **game**, in the Java application, **HangmanGame**.
3. Create a class, **Hangman**, in the package, **game**.
4. Replace the code in the **Hangman.java** file with the following code:

```
package game;
import java.util.Scanner;
public class Hangman {
    public void showMenu()
    {
        int option;
        Scanner sc = new Scanner(System.in);
        System.out.println("---------Menu---------");
        System.out.println("1. Play");
        System.out.println("2. Instructions");
        System.out.println("3. Exit");
        System.out.print("\nChoose the option: ");
        option = sc.nextInt();
    }
public static void main(String[] args)
    {

        Hangman hg = new Hangman();
        hg.showMenu();
    }
}
```

5. Compile and execute the **HangmanGame** Java application.

After executing the preceding code, the following output is displayed:

```
---------Menu---------
1. Play
2. Instructions
3. Exit

Choose the option:
```

Note

Refer to the toolbook to create and execute the Java application in NetBeans.

Exercises

Exercise 1

Alchem Inc. is an emerging company in Singapore. Recently, the organization has seen a huge growth following which the management has decided to hire 100 employees. In order to organize the details of an ever rising count of employees in an efficient manner, the management has asked Shawn, the Head of the IT department, to create an application named EmployeeBook. This application will store details, such as, employee ID, employee name, department, designation, date of joining, date of birth, marital status, and date of marriage. This application will be useful for storing the details of the employees in an organized manner. It will also be useful in updating and accessing the personal details of the employees quickly. To achieve this, the application must provide the functionality to store the personal employee data that includes the employee name, mobile number, address, marital status, and date of birth. Ask the students to identify the following components:

- Various classes
- Class members
- Access modifiers

Exercise 2

Create a class named EmployeeDetails for the EmployeeBook application and display a menu similar to the following menu:

```
---------Menu---------
1. Enter Data
2. Update Data
3. Display Data
4. Exit

Choose the option:
```

In addition, the application should accept a menu input from the user, such as 1, 2, 3, and 4.

Note

*Create a Java application, **EmployeeBook**, in NetBeans and create a package, **alchem**. Further, create the **EmployeeDetails** class for exercise 2, inside the **alchem** package.*

Exercise 3

Predict the output of the following code:

```
class JumbledGame {
        final int choice;
        public static void main(String[] args) {
            JumbledGame obj1 = new JumbledGame ();
            System.out.println(obj1.choice);
        }
}
```

Exercise 4

Write a program to create a class named `SampleClass`. The `SampleClass` class should contain a variable named `counter` and assign it the value, `1`. Declare the `counter` variable as `public`. Create a method named `Display()` in the class. The `Display()` method should print the value of the `counter` variable. Create an object and access the `Display()` method using the object.

Exercise 5

Create a `Reservation` class with a member variable, `TicketID`, and a method, `ShowTicket()`. A constructor should initialize the `TicketID` variable, and the `ShowTicket()` method should display its value. In addition, create an object of the `Reservation` class to call these methods.

Note

*Create a Java application named **Exercises** and create packages with chapter names, such as **chapter01** and **chapter02**, in NetBeans. Further, in the **Exercises** application, create the classes for the exercises, 4 and 5, respectively, inside the package, **chapter01**.*

Additional Exercises

Exercise 1

ServeYourMoney bank offers various services to its customers. The bank has branches all over the country, and therefore, each branch has a unique id. The bank stores the details of its customers, such as customer name, account number, address, phone number, and email address. In addition, the bank stores the various details of its employees, such as employee name, employee id, address, phone number, and email id. The bank offers its customers a choice of accounts, such as savings account and current account. Ask the students to identify the various classes, their member variables, and data types.

Exercise 2

Create a `Grocery` class with a member variable, `weight`, and two methods, `weightNow()` and `checkWeight()`. The `weightNow()` method should initialize the `weight` variable and the `checkWeight()` method should display its value. In addition, create an object of the `Grocery` class to call the preceding methods.

Note

*You need to create a folder by your name in **D:\IJ**. In addition, create a Java application named **AdditionalExercises** and packages with names, such as **chapter01** and **chapter02**, in NetBeans. Further, in the **AdditionalExercises** application, create the classes for the exercise 2 inside the package, **chapter01**.*

Implementing Operators

CHAPTER 2

Activity 2.1: Working with Operators

Problem Statement

David needs to write a program to calculate the net payable salary. The program will take basic salary as an input from the user and calculate the total salary along with the following components:

- HRA, which is 50% of the basic salary
- Special allowance, which is 75% of the basic salary
- PF deduction, which is 12% of the basic salary

After calculating the preceding salary components, the net payable salary should be displayed. Help David to perform the preceding requirement.

Note

Net payable salary = (basic+HRA+special allowance) -PF

Solution

To solve the preceding requirement, David needs to perform the following steps:

1. Create a Java application, **SalaryCalculator**.
2. Create a package, **calculator** in the Java application, **SalaryCalculator**.
3. Create a class, **SalaryCalculator**, inside the package, **calculator**.
4. Replace the code in the **SalaryCaluculator.java** file with the following code:

```
package calculator;

import java.util.Scanner;

public class SalaryCalculator
{
   public static void main(String[] args)
   {
         int hra;
         int specialallowance;
         int pf;
         int netpayablesalary;
         int basicsalary;

         System.out.println("Enter your basic salary:");
         Scanner sc1 = new Scanner(System.in);
         basicsalary= sc1.nextInt();
```

```
            hra=(50*basicsalary)/100;
            specialallowance=(75*basicsalary)/100;
            pf=(12*basicsalary)/100;
            netpayablesalary=basicsalary+hra+specialallowance-pf;

            System.out.println("Basic salary      ="+basicsalary);
            System.out.println("HRA               ="+hra);
            System.out.println("Special allowance ="+specialallowance);
            System.out.println("PF                ="+pf);
            System.out.println("------------------------");
            System.out.println("Net payable salary="+netpayablesalary);
        }
    }
```

5. Compile and execute the **SalaryCalculator** Java application.

After executing the preceding code, the following output is displayed:

```
Enter your basic salary:
```

Once you provide the basic salary as `5000`, the following output is displayed:

```
Basic salary      =5000
HRA               =2500
Special allowance =3750
PF                =600
------------------------
Net payable salary=10650
```

Note
Refer the toolbook to create and execute the Java application in NetBeans.

Exercises

Exercise 1

John needs to write a program to calculate the total distance travelled by a vehicle. The program will take initial velocity, acceleration, and time as input from the user. To calculate the distance, John can use the following formula:

```
Distance = ut + (at²) /2
```

Here, `u` is the initial velocity (meters per second), `a` is the acceleration (meters per second^2), and `t` is the time (seconds). Help John to perform the preceding task.

Exercise 2

Mary needs to write a program to calculate the volume of a cylinder. The program will take radius and height of cylinder as input from the user. To calculate the volume, Mary can use the following formula:

```
Volume = πr²h
```

Here, `r` is the radius of the cylinder, `h` is the height of the cylinder, and value of pie (`π`) is `22/7`. Help Mary to perform the preceding task.

Exercise 3

Arnold wants to write a program to convert the temperature given in Celsius to Fahrenheit. The program will take temperature in Celsius as input from the user. To calculate the temperature, Arnold can use the following formula:

```
F = C * 9/5 +32
```

Here, `C` is the temperature in Celsius and `F` is the temperature in Fahrenheit. Help Arnold to perform the preceding task.

Exercise 4

Write a program to perform the bitwise AND, OR, NOT, and XOR operations on the numbers, 5 and 9.

Exercise 5

Consider the following expression:

```
4 * 4 + 8 / 4
```

When Sam executes the preceding expression by using the Java code, he gets an output value of 18. However, he wants that the output should be 12. Help Sam to achieve the preceding task.

Note

You need to create the classes of exercises 1, 2, 3, 4, and 5 inside the package, ***chapter02*** *inside the application,* ***Exercises****.*

Additional Exercises

Exercise 1

Write a program to display the postfix and prefix values of the given number, 5, by using increment operators.

Exercise 2

Write a program to perform left shift and right shift operations for 2 by 2 bits. In addition, perform unsigned right shift for -2 by 24 bits.

Note

You need to create the classes of exercises 1 and 2 inside the package, ***chapter02****, in the application,* ***AdditionalExercises****.*

Working with Conditional and Loop Constructs

CHAPTER 3

Activity 3.1: Working with Conditional Constructs

Problem Statement

Peter wants to display the following menu when the Hangman game starts:

1. Play Game
2. View Instructions
3. Exit Game

Thereafter, he wants a user to enter a choice, such as 1, 2, or 3. In addition, he wants the respective methods, `playGame()`, `instructGame()`, or `exitGame()`, to be invoked according to the user's input, 1, 2, or 3, respectively. Help Peter to achieve the preceding requirement.

Prerequisite

To perform the activity, you need to use the **Activity1.1.txt** file.

Solution

To achieve the preceding requirement, Peter needs to perform the following steps:

1. Ensure that the **HangmanGame** Java application is open and active.
2. Ensure that the **Hangman.java** file is open.
3. Open the **Activity1.1.txt** file.
4. Replace the code in the **Hangman.java** file with the code in the **Activity1.1.txt** file.
5. Add the following code snippet after the comment, `// Switch Case`, within the `showMenu()` method in the **Hangman.java** file:

```
switch(option)
{
    case 1: playGame();
            break;
    case 2: instructGame();
            break;
    case 3: exitGame();
            break;
    default: System.out.println("Incorrect menu option");
             showMenu();
             break;
 }
```

6. Add the following code snippet after the comment, `// Method definition`, in the **Hangman.java** file:

```
public void playGame()
{
    System.out.println("playGame method is invoked");
}

public void instructGame()
{
   System.out.println("instructGame method is invoked");
}

public void exitGame()
{
   System.out.println("exitGame method is invoked");
   System.exit(0);
}
```

7. Compile and execute the **HangmanGame** Java application.

   After executing the application, the following output is displayed:

```
---------Menu---------
1. Play
2. Instructions
3. Exit

Choose the option:
```

   Once you provide the option as `1`, the following output is displayed:

```
playGame method is invoked
```

Note
Refer to the toolbook to create and execute the Java application in NetBeans.

Activity 3.2: Working with Loop Constructs

Problem Statement

In the Hangman game, Peter wants a user to enter a letter. On the basis of the user's input, he wants to check whether the letter is present in the word or not. For this, he decides to compare the letter entered by the user with each letter of the corresponding word. If the letter is present in the word, it should display an appropriate message. In addition, he wants the menu to be displayed till the user wants to guess a letter. Help Peter to achieve the preceding requirement.

Prerequisite

To perform the activity, you need to use the **Activity3.1.txt** file.

Solution

To achieve the preceding requirement, Peter needs to perform the following steps:

1. Ensure that the **HangmanGame** Java application is open and active.
2. Ensure that the **Hangman.java** file is open.
3. Open the **Activity3.1.txt** file.
4. Replace the code in the **Hangman.java** file with the code in the **Activity3.1.txt** file.
5. Add the following code snippet after the comment, `// Instance variable`, in the **Hangman.java** file:

```
String word = "australia";
```

6. Replace the code after the comment, `//playGame functionality`, within the `playGame()` method in the **Hangman.java** file:

```
int i, flag =0;
String input, guess;
Scanner sc = new Scanner(System.in);
do
{
    System.out.print("\nEnter Your Guess: ");
    input = sc.nextLine();

    for(i=0;i<word.length();i++)
    {
        if(word.charAt(i) == input.charAt(0))
        {
            flag=1;
        }
    }
```

```
        if(flag==1)
        {
            System.out.println("This letter is present in the word");
        }
        else
        {
            System.out.println("This letter is not present in the word");
        }

        System.out.println("Do want to guess again(y/n):");
        guess=sc.nextLine();
        flag=0;

      }while(guess.equals("y")||guess.equals("Y"));
```

7. Compile and execute the **HangmanGame** Java application.

   After executing the application, the following output is displayed:

```
 ---------Menu---------
1. Play
2. Instructions
3. Exit

Choose the option:
```

   Once you provide the option as `1`, the following output is displayed:

```
Enter Your Guess:
```

   Once you provide the guess as `u`, the following output is displayed:

```
This letter is present in the word
Do want to guess again(y/n):
```

Note

Refer to the toolbook to create and execute the Java application in NetBeans.

Exercises

Exercise 1

In the EmployeeBook application, David wants to display the following menu when the application starts:

1. Enter Data
2. Display Data
3. Exit

Thereafter, he wants a user to enter a choice, such as 1, 2, or 3. Based on the input provided by the user, he wants the respective methods, `enterData()`, `displayData()`, or `exitMenu()`, to be invoked according to the user's input, 1, 2, or 3, respectively. Help David to achieve the preceding requirement.

Note
*You need to modify the class, **EmployeeDetails**, inside the package, **alchem**, in the application, **EmployeeBook**, created in Chapter 1.*

Exercise 2

Write a program to display the following numeric pattern:

12345

1234

123

12

1

Exercise 3

Write a program to identify whether a character entered by a user is a vowel or a consonant.

Exercise 4

Write a program to accept a number from a user and display all the prime numbers between one and the number entered by the user.

Exercise 5

Write a program to accept two numbers and check if the first is divisible by the second. In addition, an error message should be displayed if the second number is zero.

Note

You need to create the classes of exercises, 2, 3, 4, and 5, inside the package, ***chapter03****, in the application,* ***Exercises****.*

Additional Exercises

Exercise 1

Write a program to print the table of five.

Note
You need to create the class of exercise 1 inside the package, ***chapter03****, in the application,* ***AdditionalExercises****.*

NIIT

Working with Arrays, Enums, and Strings

CHAPTER 4

Activity 4.1: Manipulating Arrays and Strings

Problem Statement

In the Hangman game, when a user decides to play the Hangman game, the numbers of dashes or blanks that are equal to the length of a word are displayed. To implement this, Peter needs to store a group of words, randomly pick a word, calculate the length of the word, and finally displays the number of dashes or blanks. In addition, he wants to implement the functionality that takes an alphabetical character as an input from the user, check if the character entered by the user exists in the corresponding word, and display the letter in the appropriate dashes. However, if the character doesn't exist in the word, it is added to the list of missed letters. Help Peter to achieve the preceding requirement.

Prerequisite

To perform the activity, you need to use the **Activity3.2.txt** file.

Solution

To achieve the preceding requirement, Peter needs to perform the following steps:

1. Ensure that the **HangmanGame** Java application is open and active.
2. Ensure that the **Hangman.java** file in the **game** package is open.
3. Open the **Activity3.2.txt** file.
4. Replace the code in the **Hangman.java** file with the code in the **Activity3.2.txt** file.
5. Replace the statement after the comment, `// Instance variable`, with the following code snippet, in the **Hangman.java** file:

```
String word[] = {"japan", "qatar", "syria", "mongolia", "bahrain", "india"};
```

6. Add the following code snippet after the comment, `// import statements`:

```
import java.util.Random;
```

7. Replace the code after the comment, `//playGame functionality`, within the `playGame()` method in the **Hangman.java** file:

```
int len, i, count = 0, rnd, flag =0;
String choice, temp;

Random rd = new Random();
Scanner input = new Scanner(System.in);

rnd = rd.nextInt(6); /* generates a random number between -1 and 7 and
assigns the same to the variable rnd */
```

```
len = word[rnd].length();
char[] newString = new char[len];


StringBuffer wrgString = new StringBuffer();

 for(int j = 0; j < len; j++)
 {
      System.out.print("_ ");
 }
 System.out.println();
       do
        {
            flag =0;

            System.out.print("\n\nEnter your guess: ");
            String ch = input.nextLine().toLowerCase();
            count++;

            for (i = 0; i < len; i++)
            {
                if (word[rnd].charAt(i) == ch.charAt(0))
                {
                    newString[i] = word[rnd].charAt(i);
                    flag = 1;
          }
       }
            if (flag == 0)
            {
          flag=1;
                wrgString.append(ch + ",");
          System.out.println("\nMisses: " + wrgString);
            }

            System.out.println(newString);
            temp = new String(newString);

            if (word[rnd].equals(temp))
            {
          System.out.println("---------- Congrats :) You won ------------");
          System.out.println("Do you want to play again (Y/N)");
          choice = input.nextLine();
                if (choice.equalsIgnoreCase("y"))
          {
                 playGame();
          }
          else
          {
                 showMenu();
          }

            }
        }while(flag!=0);
```

Note

The method, `int nextInt(int)`*, of* `java.util.Random` *class is used generate a random number between* `-1` *and the value passed as argument to the method.*

8. Compile and execute the **HangmanGame** application.

After executing the application, the following output is displayed:

```
 ---------Menu---------
1. Play
2. Instructions
3. Exit

Choose the option:
```

Once you provide the option as `1`, the following output is displayed:

```
_ _ _ _ _ _ _ _


Enter your guess:
```

Note

Number of dashes may vary as per the functionality of the game.

Once you provide the guess as `a`, the following output is displayed:

```
a

Enter your guess:
```

Note

The preceding output will only be displayed if the entered letter is present in the word corresponding to the dashes. However, if the letter is not present in the word corresponding to dashes it will show the following output:

```
Misses: <letter1, letter2,..., letterN>

Enter your guess:
```

Exercises

Exercise 1

In the EmployeeBook application, David wants to implements the following functionalities:

- When user selects the option to enter data, the employee details, such as employee ID, employee name, department, designation, date of joining, date of birth, marital status, and date of marriage should be captured. However, date of marriage can be captured only if the employee is married.
- When user selects the option to view data, the employee id of the employee for which the data needs to view should be entered and validated. If the employee id is correct, the details should be displayed else an error message should be displayed.
- When user selects the option to exit, the application must terminate.

Help David to implement the preceding functionalities.

Note

*You need to modify the class, **EmployeeDetails** inside the package, **alchem**, in the application, **EmployeeBook** created in Chapter 3.*

Additional Exercises

Exercise 1

Sam has been assigned the task to create a Java application that a user can use to know the degrees of directions shown by a compass. In this application, he needs to restrict the user to specify the directions as NORTH, SOUTH, EAST, WEST, NORTHEAST, SOUTHEAST, SOUTHWEST, or NORTHWEST. Create an enum and declare a class to test the enum that Sam can use in his application.

Note
*You need to create the class for exercises 1 inside the package, **chapter04**, in the application, **AdditionalExercises**.*

NIIT

Implementing Inheritance and Polymorphism

CHAPTER 5

Activity 5.1: Implementing Inheritance and Polymorphism

Problem Statement

The management of LearnMore University is planning to automate the University management system. Therefore, Steve Wilkinson, the programmer, has decided to create a Java program that accepts the student details, such as the first name, last name, age, course enrolled, and student ID. In addition, he also needs to accept the employee details, such as first name, last name, age, salary, department name, designation, and employee ID. Steve must ensure the reusability of code. The program must offer a choice to accept either the student's or employee's details. Help Steve to develop the program.

Solution

To achieve the preceding requirements, Steve needs to perform the following steps:

1. Create a Java application, **Information**.
2. Create a package, **university**, in the application, **Information**.
3. Create a class, **Person**, inside the package, **university**.
4. Replace the code in **Person.java** with the following code:

```
package university;
import java.util.Scanner;
public class Person
{

    String firstName;
    String lastName;
    int age;

    public void getDetail()
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the Basic Details");
        System.out.println("Enter the First Name");
        firstName = input.nextLine();
        System.out.println("Enter the Last Name");
        lastName = input.nextLine();
        System.out.println("Enter the Age");
        age = input.nextInt();
    }
}
```

5. Create a class, **Employee**, inside the package, **university**.

6. Replace the code in **Employee.java** with the following code:

```
package university;
import java.util.Scanner;
class Employee extends Person
{
    double Salary;
    String desg;
    String dept;
    public void getDetail()
    {
        super.getDetail();
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the Professional Details");
        System.out.println("Enter the Department");
        dept = input.nextLine();
        System.out.println("Enter the Designation");
        desg = input.nextLine();
        System.out.println("Enter the Salary");
        Salary = input.nextInt();
        showDetail();
    }

    public void showDetail()
    {
        System.out.println("The details entered are: \n");
        System.out.println("First Name: " + super.firstName);
        System.out.println("Last Name: " + super.lastName);
        System.out.println("Age: " + super.age);
        System.out.println("Department: " + dept);
        System.out.println("Designation: " + desg);
        System.out.println("Salary: " + Salary);
    }
}
```

7. Create a class, **Student**, inside the package, **university**.
8. Replace the code in **Student.java** with the following code:

```
package university;
import java.util.Scanner;
class Student extends Person
{
    String University;
    String stream;
    int StudentID;
    public void getDetail()
    {
        super.getDetail();
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the Academic Details");
        System.out.println("Enter the Course enrolled");
        stream = input.nextLine();
        System.out.println("Enter the Student ID");
```

```
            StudentID = input.nextInt();
            showDetail();
        }
        public void showDetail()
        {
            System.out.println("The details entered are: \n");
            System.out.println("First Name: " + super.firstName);
            System.out.println("Last Name: " + super.lastName);
            System.out.println("Age: " + super.age);
            System.out.println("Course Enrolled: " + stream);
            System.out.println("Student ID: " + StudentID);
        }
    }
```

9. Create a class, **MainMenu**, inside the package, **university**.
10. Replace the code in **MainMenu.java** with the following code:

```
    package university;
    import java.util.Scanner;
    public class MainMenu
    {
        public static void main(String[] args) {
            Scanner choice = new Scanner(System.in);
            System.out.println("Menu");
            System.out.println("1. Student Details");
            System.out.println("2. Employee Details");
            System.out.println("Enter the choice: ");
            int a = choice.nextInt();
            switch (a)
            {
                case 1:
                    Student st = new Student();
                    st.getDetail();
                    break;
                case 2:
                    Employee emp1 = new Employee();
                    emp1.getDetail();
                    break;
            }
        }
    }
```

11. Compile and execute the **Information** Java application.

After executing the application, the following output is displayed:

```
    Menu
    1. Student Details
    2. Employee Details
    Enter the choice:
```

Note

After specifying the choice as `1` or `2`, you can enter the desired details for a student or an employee, respectively.

Note

Refer to the toolbook to create and execute the Java application in NetBeans.

Exercises

Exercise 1

Furniture and Fittings Company (FFC) manufactures chairs and bookshelves. The customers provide their specifications for the desired furniture item to the company. The furniture items in the company have similar characteristics, such as price, width, and height. However, for some furniture items, specific details need to be provided by the customers. Recently, the business of the company has increased significantly. To handle the increased customer's orders, FFC has decided to computerize the order processing system. For this, you need to develop a Java program that accepts the furniture attributes from the customers, as per the specified furniture items, such as bookshelves and chairs. You need to develop the program to accept and display the various choices of customers.

Exercise 2

Create a library management system program that will be used to store the details of the software books and hardware books. For software books, the various pieces of information, such as the author name, title, price, stock, software version, software name, and number of pages, should be accepted.

For hardware books, the various pieces of information, such as the author name, title, price, number of pages, stock, hardware category, and publisher, should be accepted. After accepting the details in either of the book categories, the entered details must be displayed. You need to ensure that code reusability is implemented in the program.

Note

*You need to create the classes of exercises 1 and 2 inside the package, **chapter05**, inside the application, **Exercises**.*

Additional Exercises

Exercise 1

Define a `FixedAccount` class containing the method, `calculateInterest()`, which calculates the interest on the balance for a fixed account. In addition, extend the `FixedAccount` class and override the `calculateInterest()` method to calculate the interest for the savings account. The interest rate for the fixed and savings account are different.

Note

*You need to create the classes of exercise 1 inside the package, **chapter05**, in the application, **AdditionalExercises**.*

NIIT

Handling Errors

CHAPTER 6

Activity 6.1: Exception Handling

Problem Statement

In the Hangman game, Peter wants to implement exception handling, such that if a user enters a menu input other than 1, 2, or 3 in the game menu option, an appropriate user-defined exception should be generated. In addition, a user-defined exception should be generated if the player inputs multiple characters, instead of a single character to identify the possible alphabet in the word.

Prerequisite

To perform the activity, you need to use the **Activity4.1.txt** file.

Solution

To achieve the preceding requirements, Peter needs to perform the following steps:

1. Ensure that the **HangmanGame** Java application is open and active.
2. Create a new exception class named **WrongInputException** in the **game** package.
3. Replace the code in the **WrongInputException** file with the following code:

```
package game;
public class WrongInputException extends RuntimeException
{
    WrongInputException()
    {
        System.out.println("Please provide a single character only..!!");
    }
}
```

4. Create a new exception class named **MenuInputException** in the **game** package.
5. Replace the code in the **MenuInputException** file with the following code:

```
package game;
public class MenuInputException extends RuntimeException
{
    MenuInputException()
{

        System.out.println("Please provide a valid input (1-3)");
   }
 }
```

6. Open the **Activity4.1.txt** file.
7. Open the **Hangman.java** file.
8. Replace the code in the **Hangman.java** file with the code in the **Activity4.1.txt** file.

9. Replace the statements after the comment, `// try-catch block`, within the `showMenu()` method with the following statements:

```
option=0;
        try
        {
            option = sc.nextInt();
        }
       catch(RuntimeException e)
        {
        System.out.println("Please provide a valid numeric input");
       showMenu();
        }
        // Switch Case
        switch(option)
        {
            case 1: playGame();
                    break;
            case 2: instructGame();
                   break;
            case 3: System.exit(0);
                    break;

            default:
        try
{
          throw new MenuInputException();
          }
        catch (Exception e)
             {
                showMenu();
             }
        }
```

10. Replace the statements after the comment, `//Implement user defined exception`, within the `playGame()` method with the following code snippet:

```
try
{
String ch = input.nextLine().toLowerCase();
if(ch.length()!=1)
{
throw new WrongInputException();
}
count++;
for (i = 0; i < len; i++)
{
if (word[rnd].charAt(i) == ch.charAt(0))
{
newString[i] = word[rnd].charAt(i);
flag = 1;
}
}
```

```
    if (flag == 0)
    {
    flag=1;
    wrgString.append(ch + ",");
    System.out.println("\nMisses: " + wrgString);
    }

    System.out.println(newString);
    temp = new String(newString);

    if (word[rnd].equals(temp))
    {
    System.out.println("---------- Congrats :) You won ------------");
    System.out.println("Do you want to play again (Y/N)");
    choice = input.nextLine();

    if (choice.equalsIgnoreCase("y"))
    {
    playGame();
    }
    else
    {
    showMenu();
    }

    }
    }
    catch (WrongInputException e)
    {
    //System.out.println(e);
    flag = 1;
    }
    }
    while(flag!=0);
```

11. Compile and execute the **HangmanGame** Java application.

After executing the application, the following output is displayed:

```
    ---------Menu---------
    1. Play
    2. Instructions
    3. Exit

    Choose the option:
```

Once you provide the option as `a`, the following output is displayed:

```
    Please provide a valid numeric input
    ---------Menu---------
    1. Play
    2. Instructions
    3. Exit

    Choose the option:
```

Once you provide the option as `7`, the following output is displayed:

```
Please provide a valid input (1-3)
---------Menu---------
1. Play
2. Instructions
3. Exit

Choose the option:
```

Once you provide the option as `1`, the following output is displayed:

```
     _ _ _ _ _


Enter your guess:
```

Once you specify the guess as `aaa`, the following output is displayed:

```
Please provide a single character only..!!


Enter your guess:
```

Note
The output may vary as per the functionality of the game.

Note
Refer to the toolbook to create and execute the Java application in NetBeans.

Activity 6.2: Implementing Assertions

Problem Statement

Create a calculator application in Java that will accept two numbers. Further, the calculator application should be able to perform the following operations one at a time on the two numbers:

- Addition
- Subtraction
- Multiplication
- Division

You need to implement assertions and assert that both the numbers should be greater than 0. Further, the operator used to perform the calculations should only be +,-, *, or, /.

Solution

To achieve the preceding requirements, you need to perform the following steps:

1. Create a Java application, **CalculatorApplication**.
2. Create a package, **calculator**, in the application, **CalculatorApplication**.
3. Create a class, **CalculatorApp**, inside the package, **calculator**.
4. Enable assertions for the calculator package.
5. Ensure that the **CalculatorApp.java** file is open.
6. Replace the code in the **CalculatorApp.java** file with the following code:

```
package calculator;
import java.util.Scanner;
class CalculatorApp
{

    public static void main(String[] args)
    {

        int num1, num2, res = 0;
        Scanner numbers = new Scanner(System.in);
        Scanner operator = new Scanner(System.in);
        String op;
        System.out.println("Enter the 1st number");
        num1 = numbers.nextInt();
        assert (num1 > 0) : "The number should be greater than 0.";
        System.out.println("Enter the 2nd number");
        num2 = numbers.nextInt();
        assert (num2 > 0) : "The number should be greater than 0.";
        System.out.println("enter the operator");
        op = operator.nextLine();
        assert ((op.equals("+")) || (op.equals("-")) || (op.equals("*")) ||
(op.equals("/"))) : "The operator is not valid.";

        if (op.equals("+"))
        {
            res = num1 + num2;
        }

        else if (op.equals("-"))
        {

            if (num1 > num2)
            {
                res = num1 - num2;
            } else
            {
                res = num2 - num1;
            }
        }

        else if (op.equals("*"))
        {
            res = num1 * num2;
        }

        else if (op.equals("/"))
        {
            res = num1 / num2;
        }
        else
        {
            System.out.println("Wrong operator");
        }
        System.out.println("The result is " + res);
    }
}
```

7. Compile and execute the **CalculatorApplication** Java application.

After executing the preceding code, the following output is displayed:

```
Enter the 1st number
```

Once you enter the 1st number as `8`, the following output is displayed:

```
Enter the 2nd number
```

Once you enter the 2nd number as `5`, the following output is displayed:

```
enter the operator
```

Once you enter the operator as `&`, the following output is displayed:

```
Exception in thread "main" java.lang.AssertionError: The operator is not
valid.
    at calculator.CalculatorApp.main(CalculatorApp.java:21)
Java Result: 1
```

Note

The line number in the `CalculatorApp.java:21` statement may vary.

Exercises

Exercise 1

Consider the scenario of the EmployeeDetails application. In this application, David needs to implement user-defined exceptions for the following cases:

- If the menu input entered by the user is less than 1 or greater than 3, an appropriate message should be displayed to the user and the application must restart.
- The employee id entered does not start with the alphabet “e”.

Note
You need to modify the class, ***EmployeeDetails****, inside the package,* ***alchem****, in the application,* ***EmployeeBook****, created in Chapter 4.*

Exercise 2

Sammy is a programmer at NewAxis Technologies. He needs to write a program, which will accept the employee details from a user. A user should be able to enter the name, contact details, and age. The minimum and maximum ages are 20 and 55, respectively. If the age of an employee is less than 20 or greater than 55, a user-defined exception should be thrown. Help Sammy to perform this task.

Exercise 3

Create a program to divide two numbers. The program should provide the functionality, such that if a user tries to perform the division operation by 0, the program should terminate with a customized message.

Note
You need to create the classes of exercises 2 and 3 inside the package, ***chapter06****, inside the application,* ***Exercises****.*

Working with Regular Expressions

CHAPTER 7

Activity 7.1: Processing Strings Using Regex

Problem Statement

Create a method, which should have the email-id and password fields as parameters. When a user calls the method, the email and password entered by the user should be validated. The email-id field should be specified in the format, such as xyz@xyz.co.in or xyz@xyz.com.

In addition, the format of the password should adhere to the following conventions:

- There should be one special character.
- There should be one uppercase letter.
- There should be one lowercase letter.
- The password should be eight characters long.

Prerequisite

To perform the activity, you need to use the **LoginForm.txt** file.

Solution

To achieve the preceding requirements, you need to perform the following steps:

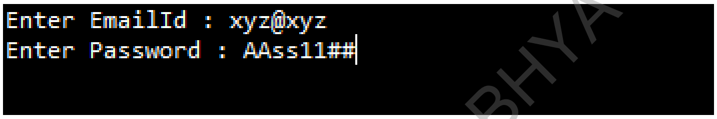
1. Create a Java application, **LoginFormApplication**.
2. Create a package, **loginformvalidate**, in the application, **LoginFormApplication**.
3. Create a class, **LoginForm**, inside the package, **loginformvalidate**.
4. Open the **LoginForm.txt** file.
5. Replace the code in the **LoginForm.java** file with the code in the **LoginForm.txt** file.
6. Add the following code after the `//Validation Code` comment:

```
Pattern pattern = Pattern.compile("^[a-z]+[a-z.0-9-]+@[a-z.-]+(\\.[A-Za-z0-
9]+)$");
 Matcher matcher = pattern.matcher(email);
 boolean emailValidator=matcher.matches();
Pattern passwordPattern = Pattern.compile("^(?=.*[0-9])(?=.*[a-z])(?=.*[A-
Z])(?=.*[@#$%^&+=]).{8,}$");
        Matcher matcherPassword = passwordPattern.matcher(password);
        boolean passwordValidator=matcherPassword.matches();
        if(emailValidator==true && passwordValidator==true)
        {
            System.out.println("Email Id and password are in correct
format...");
            return true;
        }
        else if(emailValidator==false)
        {
            System.out.println("Email Id is in incorrect format...");
```

```
                return false;
            }
            else if(passwordValidator==false)
            {
                System.out.println("Password is in incorrect format...");
                return false;
            }
            else
            {
                System.out.println("Email Id or Password is in incorrect
    format...");
                return false;
            }
```

7. Compile and execute the **LoginFormApplication** application.

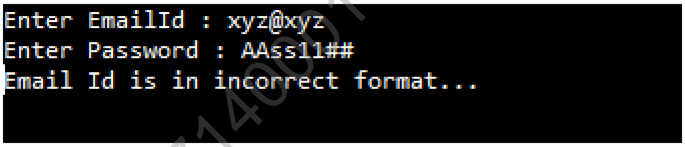
   After executing the application, the following output is displayed:



```
Enter EmailId : xyz@xyz
Enter Password : AAss11##
```

The Output of the LoginFormApplication Application

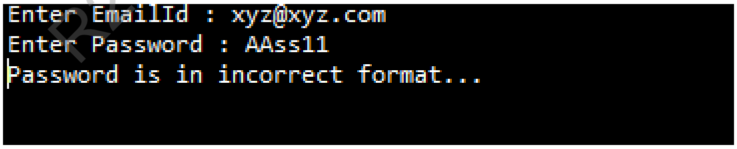
Once you have typed the text, **xyz@xyz**, in the **Email-id** and **AAss11##** in the **Password** and click the **Enter** button. The following output is displayed:



```
Enter EmailId : xyz@xyz
Enter Password : AAss11##
Email Id is in incorrect format...
```

The Output of the LoginFormApplication Application

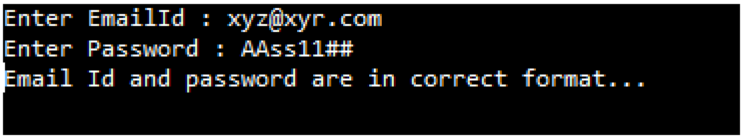
Now type the text **xyz@xyz.com** in the **Email-id** and **AAss11** in the **Password** and click the **Enter** button. The following output is displayed:



```
Enter EmailId : xyz@xyz.com
Enter Password : AAss11
Password is in incorrect format...
```

The Output of the LoginFormApplication Application

And now, type the text **xyz@xyz.com** in the **Email-id** and **AAss11##** in the **Password** and click the **Enter** button. The following output is displayed:



```
Enter EmailId : xyz@xyr.com
Enter Password : AAss11##
Email Id and password are in correct format...
```

The Output of the LoginFormApplication Application

Exercises

Exercise 1

Write a program that accepts a Web portal name from the user. In addition, the program should validate the format of the Web portal name. The Web portal name should have the following fields:

- Protocol, such as https or http.
- Domain, such as xyz.
- The top level domain should be more than one character, such as .co.

Note

You need to create the classes of the exercise 1 inside the package, ***chapter07****, inside the application,* ***Exercises****.*

Additional Exercises

Exercise 1

You need to develop an application that accepts a name and contact number as inputs from the user. In addition, you need to validate these inputs based on the following constraints:

- The name should contain only letters.
- The contact number should contain only numeric values in the (xxx-xxxx-xxxx) format.

Note

You need to create the classes of the exercise 1 inside the package, ***chapter07****, in the application,* ***AdditionalExercises****.*

Working with Streams

CHAPTER 8

Activity 8.1: Working with Input Stream and Output Stream

Problem Statement

David is a Java developer at MacroBusiness Inc. He has been assigned a task to develop the application to find the factorial. In this application, he needs to implement the following functionalities:

- The User should be presented with a menu, which has the options to Calculate factorial, view the instructions, view user's factorial, and exit.
- Every time, when the user play the game, their score should be updated in a text file. In addition, if a player decides to view the factorial Score, it should be displayed.

Help David to achieve this task.

Prerequisite

To perform the activity, you need to use the **FileOperation.txt**, **GameMenu.txt**, **GetScore.txt**, **Factorial.txt** and **Main.txt** files.

Solution

To achieve the preceding task, David needs to perform the following steps:

1. Create a Java application, **GameApp**.
2. Create a package, **game**, in the application, **GameApp**.
3. Create a class, **GameMenu**, inside the package, **game**.
4. Open the **GameMenu.txt** file.
5. Replace the code in the **GameMenu.java** file with the code in the **GameMenu.txt** file.
6. Create a class, **GetScore**, inside the package, **game**.
7. Open the **GetScore.txt** file.
8. Replace the code in the **GetScore.java** file with the code in the **GetScore.txt** file.
9. Create a class, **Factorial**, inside the package, **game**.
10. Open the **Factorial.txt** file.
11. Replace the code in the **Factorial.java** file with the code in the **Factorial.txt** file.
12. Create a class, **Main**, inside the package, **game**.
13. Open the **Main.txt** file.
14. Replace the code in the **Main.java** file with the code in the **Main.txt** file.
15. Create a class, **FileOperation**, inside the package, **game**.
16. Open the **FileOperation.txt** file.
17. Replace the code in the **FileOperation.java** file with the code in the **FileOperation.txt** file.

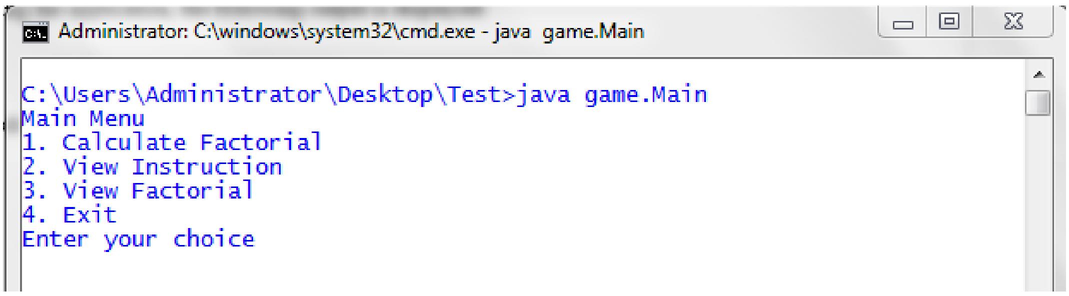
18. Add the following code after the `//Reading Data` comment:

```
name = player.toLowerCase();
String factorial = "";

    try (BufferedReader bf = new BufferedReader(new FileReader("Score.txt")))
{
                String stringSearch = name;

                String line;

                while ((line = bf.readLine()) != null) {
                        String txt[] = line.split(" ");
                        for (int i = 0; i < txt.length; i++) {
                                if (txt[i].equals(stringSearch)) {

                                        factorial=txt[i+1];

                                }
                        }
                }


        }

        catch (IOException e) {

        }
        return factorial;
```

19. Add the following code after the `//Writing Data` comment:

```
try(BufferedWriter bw = new BufferedWriter(new FileWriter("Score.txt",
true))) {
                bw.append("\n" + player1.toLowerCase() + " " + fact);
                bw.append("\n ........................................................");
        }
```

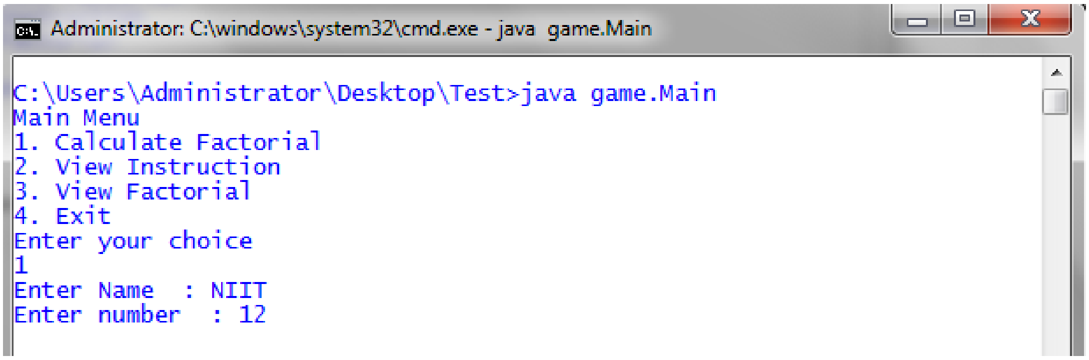
20. Compile and execute the **GameApp** application.
After executing the application, the following output is displayed:



```
Administrator: C:\windows\system32\cmd.exe - java  game.Main

C:\Users\Administrator\Desktop\Test>java game.Main
Main Menu
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
```

The Output of the GameApp Application

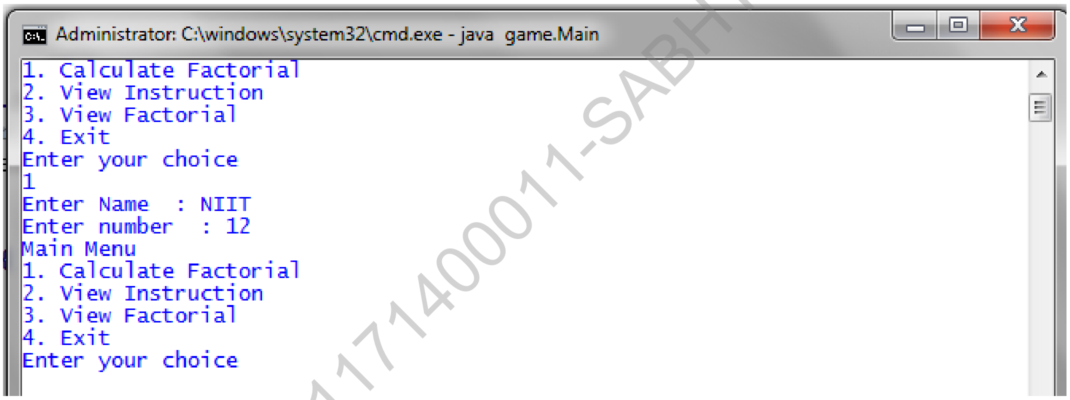
Once you enter the choice 1. The following output is displayed:



```
C:\Users\Administrator\Desktop\Test>java game.Main
Main Menu
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
1
Enter Name  : NIIT
Enter number  : 12
```

The Output of the GameApp Application

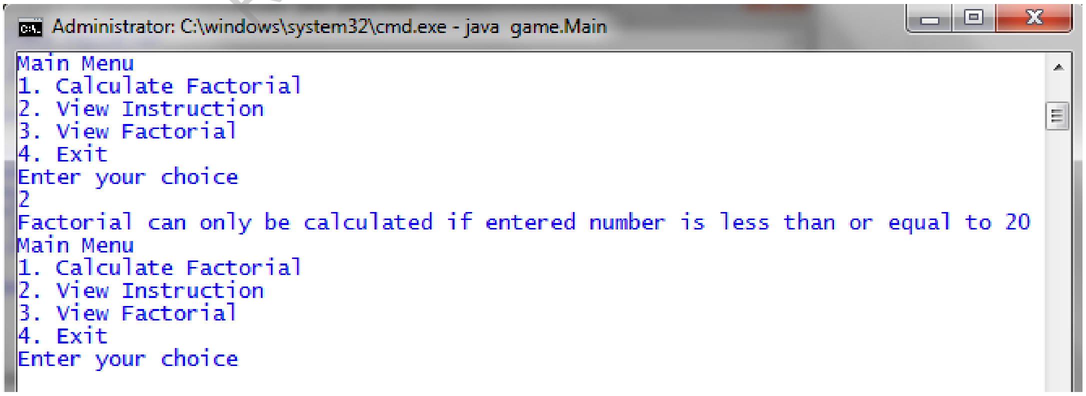
Once you have typed user name **NIIT** in the **Enter Name field** and and **12** in the **Enter Number** field and press **Enter**. The following output is displayed:



```
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
1
Enter Name  : NIIT
Enter number  : 12
Main Menu
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
```

The Output of the GameApp Application

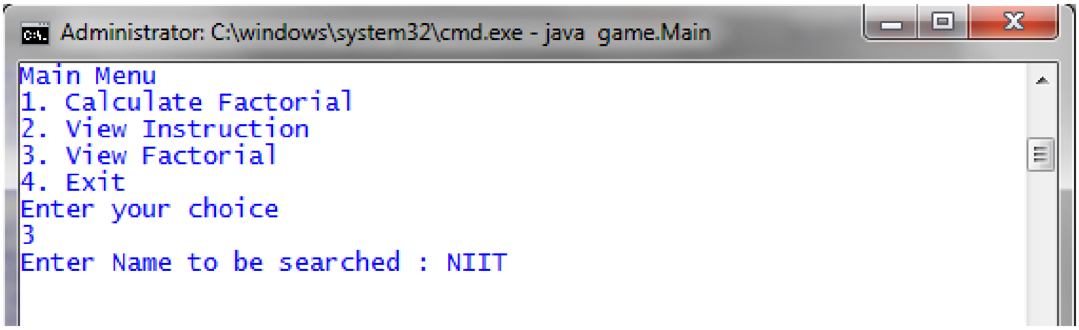
If you enter your choice as **2**. The following output is displayed:



```
Main Menu
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
2
Factorial can only be calculated if entered number is less than or equal to 20
Main Menu
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
```

The Output of the GameApp Application

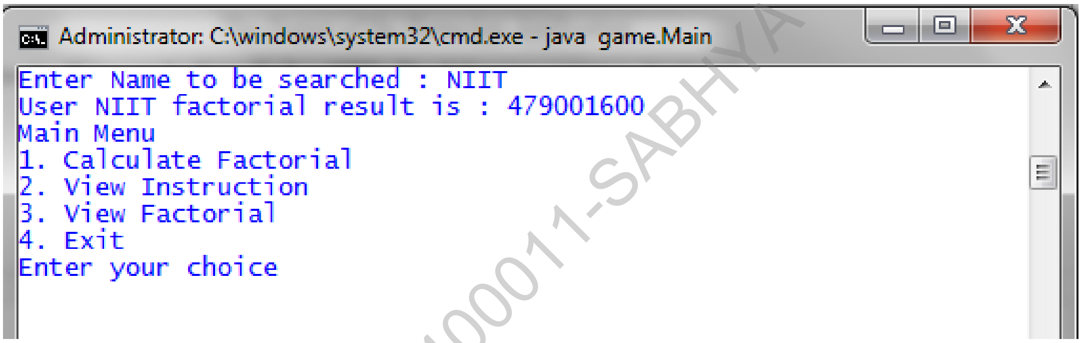
And if you enter the choice as 3. The following output is displayed:



```
Administrator: C:\windows\system32\cmd.exe - java  game.Main
Main Menu
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
3
Enter Name to be searched : NIIT
```

The Output of the GameApp Application

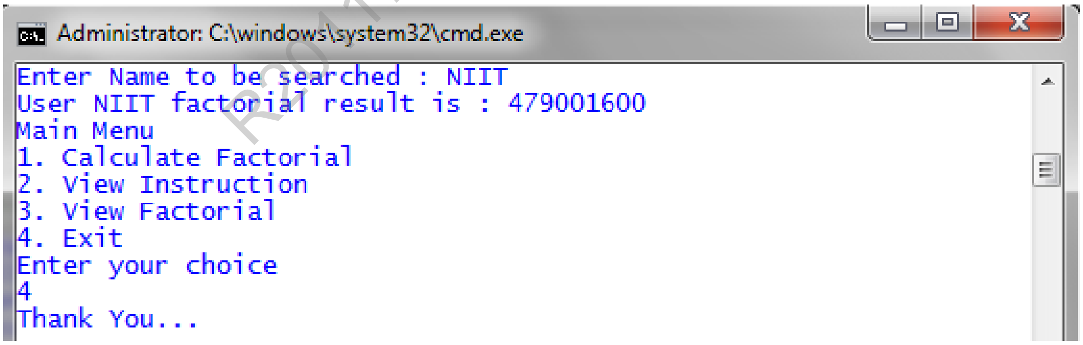
Once you enter the **Name to be Searched** as NIIT. The following output is displayed:



```
Administrator: C:\windows\system32\cmd.exe - java  game.Main
Enter Name to be searched : NIIT
User NIIT factorial result is : 479001600
Main Menu
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
```

The Output of the GameApp Application

Once you Enter the **Choice 4** . The following output will be displayed:



```
Administrator: C:\windows\system32\cmd.exe
Enter Name to be searched : NIIT
User NIIT factorial result is : 479001600
Main Menu
1. Calculate Factorial
2. View Instruction
3. View Factorial
4. Exit
Enter your choice
4
Thank You...
```

Note

You can enter a Number less than 20 in the Enter Number field.

Exercises

Exercise 1

You need to write a program which provides an option to either input the book details or display the book details. In addition, on selecting the input option, the book details, such as the book name, author name, and price, should be accepted and written into a text file. Further, if the display option is selected, all the book details should be displayed.

Exercise 2

You need to develop an application that accepts the name of a file and the word to be searched in that file. Based on the input that the user has provided, the application should display the number of occurrences of that word in the file.

Note

In order to perform the search in a document in the exercise, 2, you can use an existing document.

Note

You need to create the classes of the exercises, 1 and 2, inside the package, ***chapter08****, inside the application,* ***Exercises****.*

Additional Exercises

Exercise 1

Write a program that accepts an absolute file path, such as D:\Textfile.txt, and the content to write into the file from the user. The program should continue accepting the content from the user until the user inputs the text, end.

Note
*You need to create the classes of the exercise, 1, inside the package, **chapter08**, in the application, **AdditionalExercises**.*