

### **Exercise - 1 (Basics)**

**a) Write a JAVA program to display default value of all primitive data type of JAVA**

**program:**

```
class DefaultValues {
    static boolean a;
    static byte b;
    static short c;
    static int d;
    static long e;
    static float f;
    static double g;
    static char h;
    static String i;
    public static void main(String args[]) {
        System.out.println("Default value of boolean datatype is "+a);
        System.out.println("Default value of byte datatype is "+b);
        System.out.println("Default value of short datatype is "+c);
        System.out.println("Default value of int datatype is "+d);
        System.out.println("Default value of long datatype is "+e);
        System.out.println("Default value of float datatype is "+f);
        System.out.println("Default value of double datatype is "+g);
        System.out.println("Default value of char datatype is "+h);
        System.out.println("Default value of string datatype is "+i);
    }
}
```

**Output:**

Default value of boolean datatype is false

Default value of byte datatype is 0

Default value of short datatype is 0

Default value of int datatype is 0

Default value of long datatype is 0

Default value of float datatype is 0.0

Default value of double datatype is 0.0

Default value of char datatype is

Default value of string datatype is null

**b) Write a java program that display the roots of a quadratic equation  $ax^2 + bx = 0$ . Calculate the discriminate D and basing on value of D, describe the nature of root.**

**Program:**

```
import java.util.Scanner;
class Quadratic {
    public static void main(String args[]) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a value");
        double a = s.nextDouble();
        System.out.println("Enter b value");
        double b = s.nextDouble();
        System.out.println("Enter c value");
        double c = s.nextDouble();
        double d = ((b*b)-(4*a*c));
        if(d>0) {
            System.out.println("Roots are real and distinct");
            double x1 = (-b + Math.sqrt(d))/(2*a);
            double x2 = (-b - Math.sqrt(d))/(2*a);
            System.out.println("Roots are "+x1+" and "+x2);
        } else if(d==0) {
            System.out.println("Roots are equal and positive");
            double x = -b/(2*a);
            System.out.println("Equal root is "+x);
        } else {
            System.out.println("Roots are imaginary");
        }
    }
}
```

**Output:**

Enter a value

4

Enter b value

16

Enter c value

2

Roots are real and distinct

Roots are -0.12917130661302934 and -3.8708286933869704

**c) Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers.**

**Program:**

```
import java.util.Scanner;
class Racers {
    public static void main(String args[]) {
        Scanner s = new Scanner(System.in);
        float r1,r2,r3,r4,r5;
        System.out.println("Enter racer 1 speed");
        r1 = s.nextFloat();
        System.out.println("Enter racer 2 speed");
        r2 = s.nextFloat();
        System.out.println("Enter racer 3 speed");
        r3 = s.nextFloat();
        System.out.println("Enter racer 4 speed");
        r4 = s.nextFloat();
        System.out.println("Enter racer 5 speed");
        r5 = s.nextFloat();
        double avg = (r1+r2+r3+r4+r5)/5.0;
        if(r1>avg)
            System.out.println("First Racer Qualified");
        if(r2>avg)
            System.out.println("Second Racer Qualified");
        if(r3>avg)
            System.out.println("Third Racer Qualified");
        if(r4>avg)
            System.out.println("Fourth Racer Qualified");
        if(r5>avg)
            System.out.println("Fifth Racer Qualified");
    }
}
```

**Output:**

```
Enter racer 1 speed
100
Enter racer 2 speed
85
Enter racer 3 speed
49
Enter racer 4 speed
58
Enter racer 5 speed
23
First Racer Qualified
Second Racer Qualified
```

## **Exercise - 2 (Operations, Expressions, Control-flow, Strings)**

**a) Write a JAVA program to search for an element in a given list of elements using binary search mechanism.**

### **Program:**

```
import java.util.Scanner;
class BinarySearchExp {
    public static void binarySearch(int arr[], int first, int last, int key) {
        while (first <= last) {
            int mid = (first + last) / 2;
            if (arr[mid] < key) {
                first = mid + 1;
            } else if (arr[mid] == key) {
                System.out.println("Element is found at index " + mid);
                break;
            } else {
                last = mid - 1;
            }
        }
        if (first > last)
            System.out.println("Element is not found");
    }

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of elements: ");
        int n = sc.nextInt();
        int arr[] = new int[n];
        System.out.print("Enter " + n + " elements in ascending order: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.print("Enter element to be searched: ");
        int key = sc.nextInt();
        binarySearch(arr, 0, (n - 1), key);
        sc.close();
    }
}
```

### **Output:**

```
Enter number of elements: 4
Enter 4 elements in ascending order: 1 2 3 4
Enter element to be searched: 3
Element is found at index 2
```

### **Output:**

```
Enter number of elements: 4
Enter 4 elements in ascending order: 1 2 3 4
Enter element to be searched: 5
Element is not found
```

**b) Write a JAVA program to sort for an element in a given list of elements using bubble sort**

**Program:**

```
import java.util.Scanner;

class BubbleSortExample {
    public static void bubbleSort(int[] arr) {
        int n = arr.length, temp;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        int arr[], n;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of elements:");
        n = sc.nextInt();
        arr = new int[n];
        System.out.print("Enter " + n + " elements to be sorted: ");
        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt();
        System.out.print("Array Before Bubble Sort: ");
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");
        System.out.println();
        bubbleSort(arr);
        System.out.print("Array After Bubble Sort: ");
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");
        sc.close();
    }
}
```

**Output:**

Enter number of elements:5

Enter 5 elements to be sorted: 13 10 12 10 3

Array Before Bubble Sort: 13 10 12 10 3

Array After Bubble Sort: 3 10 10 12 13

**c) Write a JAVA program to sort for an element in a given list of elements using merge sort.**

**Program:**

```
import java.util.Scanner;
public class Merge_Sort {
    public static void merge(int a[], int beg, int mid, int end) {
        int n1 = mid - beg + 1;
        int n2 = end - mid;

        int LeftArray[] = new int[n1];
        int RightArray[] = new int[n2];

        for (int i = 0; i < n1; i++)
            LeftArray[i] = a[beg + i];
        for (int j = 0; j < n2; j++)
            RightArray[j] = a[mid + 1 + j];

        int i = 0, j = 0, k = beg;

        while (i < n1 && j < n2) {
            if (LeftArray[i] <= RightArray[j])
                a[k++] = LeftArray[i++];
            else
                a[k++] = RightArray[j++];
        }

        while (i < n1)
            a[k++] = LeftArray[i++];

        while (j < n2)
            a[k++] = RightArray[j++];
    }

    static void mergeSort(int a[], int beg, int end) {
        if (beg < end) {
            int mid = (beg + end) / 2;
            mergeSort(a, beg, mid);
            mergeSort(a, mid + 1, end);
            merge(a, beg, mid, end);
        }
    }

    public static void main(String args[]) {
        int arr[], n;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of elements:");
        n = sc.nextInt();

        arr = new int[n];
        System.out.print("Enter " + n + " elements to be sorted: ");
        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt();
    }
}
```

```
        System.out.print("Array Before Merge Sort: ");
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");
        System.out.println();

        mergeSort(arr, 0, n - 1);

        System.out.print("Array After Merge Sort: ");
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");

        sc.close();
    }
}
```

**Output:**

Enter number of elements:5

Enter 5 elements to be sorted: 13 10 12 10 3

Array Before Merge Sort: 13 10 12 10 3

Array After Merge Sort: 3 10 10 12 13

**d) Write a JAVA program using StringBuffer to delete, remove character.**

**Program:**

```
public class DeleteChar {  
    public static void main(String args[]) {  
        StringBuffer sb = new StringBuffer("Java Programming");  
        System.out.println("The original String : " + sb);  
        sb.deleteCharAt(4);  
        System.out.println("The string after removing a character: " + sb);  
    }  
}
```

**Output:**

The original String : Java Programming

The string after removing a character: JavaProgramming



### **Exercise - 3 (Class, Objects)**

**a) Write a JAVA program to implement class mechanism. Create a class, methods and invoke them inside main method.**

#### **Program:**

```
class Test {
    int a,b;
    void SetData(int i,int j) {
        a = i;
        b = j;
    }
    void DisplayData() {
        System.out.println("Value of a is "+a);
        System.out.println("Value of b is "+b);
    }
}
class AccessTest {
    public static void main(String args[]) {
        Test o1 = new Test();
        o1.SetData(10,20);
        o1.DisplayData();
    }
}
```

#### **Output:**

Value of a is 10

Value of b is 20

**b) Write a JAVA program to implement constructor.**

**Program:**

**//default constructor**

```
class Rectangle {
    int length, breadth;
    Rectangle() {
        System.out.println("Constructing Rectangle....");
        length = 10;
        breadth = 20;
    }
    int rectArea() {
        return length*breadth;
    }
}
class Area {
    public static void main(String args[]) {
        Rectangle r1 = new Rectangle();
        System.out.println("Area of Rectangle is "+r1.rectArea());
    }
}
```

**Output:**

Constructing Rectangle....

Area of Rectangle is 200

**// parameterised constructor**

```
class Rectangle {
    int length, breadth;
    Rectangle(int l,int b) {
        System.out.println("Constructing Rectangle....");
        length = l;
        breadth = b;
    }
    int rectArea() {
        return length*breadth;
    }
}
class Area1 {
    public static void main(String args[]) {
        Rectangle r1 = new Rectangle(10,20);
        System.out.println("Area of Rectangle is "+r1.rectArea());
    }
}
```

**Output:**

Constructing Rectangle....

Area of Rectangle is 200

#### **Exercise - 4 (Methods)**

**a) Write a JAVA program to implement constructor overloading.**

##### **Program:**

```
class Add {
    int a,b;
    Add() {
        a = 10;
        b = 20;
    }
    Add(int a1,int b1) {
        a = a1;
        b = b1;
    }
    void add() {
        System.out.println("Addition of two numbers is "+(a+b));
    }
}

class Conoverloading {
    public static void main(String args[]) {
        Add obj = new Add();
        obj.add();
        Add obj1 = new Add(2,3);
        obj1.add();
    }
}
```

##### **Output:**

Addition of two numbers is 30

Addition of two numbers is 5

**b) Write a JAVA program implement method overloading.**

**Program:**

```
class Addition {
    int add(int a,int b) {
        return a+b;
    }
    int add(int a,int b,int c) {
        return a+b+c;
    }
}
class Meoverloading {
    public static void main(String args[]) {
        Addition obj = new Addition();
        System.out.println("Addition of two numbers: "+obj.add(1,2));
        System.out.println("Addition of three numbers: "+obj.add(1,2,3));
    }
}
```

**Output:**

Addition of two numbers: 3

Addition of three numbers: 6

### **Exercise - 5 (Inheritance)**

**a) Write a JAVA program to implement Single Inheritance**

#### **Program:**

```
class A {  
    int x=10;  
    public void showX() {  
        System.out.println("X = "+x);  
    }  
}  
class B extends A {  
    int y = 20;  
    public void showY() {  
        System.out.println("Y = "+y);  
    }  
}  
class SingleLevel {  
    public static void main(String args[]) {  
        B b = new B();  
        b.showX();  
        b.showY();  
    }  
}
```

#### **Output:**

X = 10

Y = 20

**b) Write a JAVA program to implement multi level Inheritance**

**program:**

```
class A {
    int x=10;
    public void showX() {
        System.out.println("X = "+x);
    }
}
class B extends A {
    int y = 20;
    public void showY() {
        System.out.println("Y = "+y);
    }
}
class C extends B {
    int z = 30;
    public void showZ() {
        System.out.println("Z = "+z);
    }
}
class MultiLevel {
    public static void main(String args[]) {
        C c = new C();
        c.showX();
        c.showY();
        c.showZ();
    }
}
```

**Output:**

X = 10

Y = 20

Z = 30

**c) Write a java program for abstract class to find areas of different shapes**

**program:**

```
import java.util.*;
abstract class Shape {
    Scanner s = new Scanner(System.in);
    float s1,s2,a;
    final float pi = 3.14f;
    public abstract void get_input();
    public abstract void cal_area();
    public void show_area() {
        System.out.println("Area is :"+a);
    }
}
class Rect extends Shape {
    public void get_input() {
        System.out.println("Enter L and B values:");
        s1 = s.nextFloat();
        s2 = s.nextFloat();
    }
    public void cal_area() {
        a = s1*s2;
    }
}
class circle extends Shape {
    public void get_input() {
        System.out.println("Enter radius of the circle");
        s1 = s.nextFloat();
    }
    public void cal_area() {
        a = pi*s1*s1;
    }
}
class Mainclass {
    public static void main(String args[]) {
        Shape s;
        s = new Rect();
        System.out.println("Rectangle");
        s.get_input();
        s.cal_area();
        s.show_area();

        s = new circle();
        System.out.println("Circle");
        s.get_input();
        s.cal_area();
        s.show_area();
    }
}
```



**Output:**

Rectangle

Enter L and B values:

10 3

Area is :30.0

Circle

Enter radius of the circle

3

Area is :28.26

### **Exercise - 6 (Inheritance - Continued)**

**a) Write a JAVA program give example for “super” keyword.**

#### **Program:**

```
class A {
    int x = 10;
    public void show() {
        System.out.println("A : x = "+x);
    }
}
class B extends A {
    int x = 20;
    public void show() {
        super.show();
        System.out.println("B : x = "+x);
    }
}
class C extends B {
    int x = 30;
    public void show() {
        super.show();
        System.out.println("C : x = "+x);
    }
}
class SuperKey {
    public static void main(String args[]) {
        C c = new C();
        c.show();
    }
}
```

#### **Output:**

A : x = 10

B : x = 20

C : x = 30

**b) Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?**

**Program:**

```
import java.util.Scanner;
interface Internal {
    void get_InternalMarks();
}
interface External {
    void get_ExternalMarks();
}
interface Marks extends Internal, External {
    void show_Marks();
}
class Results implements Marks {
    float i1, e1;
    Scanner sc = new Scanner(System.in);
    public void get_InternalMarks() {
        System.out.println("Enter internal marks:(0-40)");
        i1 = sc.nextFloat();
    }
    public void get_ExternalMarks() {
        System.out.println("Enter external marks:(0-60)");
        e1 = sc.nextFloat();
    }
    public void show_Marks() {
        System.out.println("Total marks = " + (i1 + e1));
    }
}
class Interface1 {
    public static void main(String args[]) {
        Marks m = new Results();
        m.get_InternalMarks();
        m.get_ExternalMarks();
        m.show_Marks();
    }
}
```

**Output:**

```
Enter internal marks:(0-40)
40
Enter external marks:(0-60)
60
Total marks = 100.0
```

### **Exercise - 7 (Exception)**

**a) Write a JAVA program that describes exception handling mechanism program:**

```
import java.util.*;
class TryCatch {
    public static void main(String args[]) {
        int a,b,c;
        a = b = c = 0;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a and b values");
        a = s.nextInt();
        b = s.nextInt();
        try {
            c = a/b;
            System.out.println("a/b = "+c);
        } catch(Exception e) {
            System.out.println("Division by 0 is not possible");
        }
    }
}
```

#### **Output-1:**

Enter a and b values

5 2

a/b = 2

#### **output-2:**

Enter a and b values

4 0

Division by 0 is not possible

**b) Write a JAVA program Illustrating Multiple catch clauses**

**program:**

```
class MultiCatch {  
    public static void main(String args[]) {  
        int no,d,r;  
        no = d = r = 0;  
        try {  
            no = Integer.parseInt(args[0]);  
            d = Integer.parseInt(args[1]);  
            r = no/d;  
            System.out.println("result = "+r);  
        } catch(ArrayIndexOutOfBoundsException e1) {  
            System.out.println("Entered less than two values");  
        } catch(NumberFormatException e2) {  
            System.out.println("Conversion of alphabets to int not  
possible");  
        } catch(ArithmeticException e3) {  
            System.out.println("division by zero");  
        } catch(Exception e) {  
            System.out.println(e);  
        }  
    }  
}
```

**Output-1:**

```
>java MultiCatch  
Entered less than two values
```

**Output-2:**

```
>java MultiCatch 5 4  
result = 1
```

**output-3:**

```
>java MultiCatch 5 0  
division by zero
```

**output-4:**

```
>java MultiCatch 5 a  
Conversion of alphabets to int not possible
```

### **Exercise – 8 (Runtime Polymorphism)**

**a) Write a JAVA program that implements Runtime polymorphism program:**

```
class Shape {
    void draw() {
        System.out.println("Drawing...");
    }
}
class Rectangle extends Shape {
    void draw() {
        System.out.println("Drawing Rectangle...");
    }
}
class Circle extends Shape {
    void draw() {
        System.out.println("Drawing Circle...");
    }
}
class TestPoly {
    public static void main(String args[]) {
        Shape s;
        s = new Rectangle();
        s.draw();
        s = new Circle();
        s.draw();
    }
}
```

### **Output:**

Drawing Rectangle...  
Drawing Circle..



**b) Write a Case study on run time polymorphism, inheritance that implements in above problem**

Overridden methods allow java to support run-time polymorphism; polymorphism is essential to object-oriented program because it allows a general class to specify methods that will be common to all its derivatives, while allowing subclasses to define specific implementation of some or all of those methods. Overridden methods are another way that java implements the “one interface multiple methods” aspect of polymorphism.

The fundamental idea of polymorphism is that a compiler does not know which method (Super class method or subclass method) to call during the compile time. It only knows during runtime (based on object). This is called as “late binding”. This type of polymorphism is called as polymorphism by interface.

The super class provides all elements that a subclass can use directly. It also defines methods in subclass the flexibility to define its own methods, yet still enforces a consistent interfaces. This by combining inheritance with overridden methods, a super class can define the general form of the methods that will be used by all of its sub classes.

Dynamic, runtime polymorphism is one of the most powerful mechanisms that object-oriented design brings to bear on code reuse and robustness. The ability of existing code libraries to call methods on instances of new classes without recompiling while maintaining a clean abstract interface is a profoundly powerful tool.



### **Exercise – 9 (User defined Exception)**

**a) Write a JAVA program for creation of Illustrating throw  
Program:**

```
import java.util.Scanner;

public class Main {
    public static void main(String args[]) {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter age: ");
        int age = s.nextInt();
        try {
            if (age < 18) {
                throw new ArithmeticException("Person is not eligible to vote");
            } else {
                System.out.println("Person is eligible to vote!!");
            }
        } catch (Exception e) {
            System.out.println(e);
        }
        System.out.println("rest of the code...");
    }
}
```

#### **Output:**

```
Enter age: 16
java.lang.ArithmeticException: Person is not eligible to vote
rest of the code...
```

```
Enter age: 20
Person is eligible to vote!!
rest of the code...
```

**b) Write a JAVA program for creation of illustrating finally**

**Program:**

```
import java.util.Scanner;
class ExceptionFinally {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a and b values:");
        int a = sc.nextInt();
        int b = sc.nextInt();
        try {
            System.out.println("inside try block");
            System.out.println(a / b);
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception");
        } finally {
            System.out.println("finally : i execute always.");
        }
    }
}
```

**Output:**

```
Enter a and b values:13 10
inside try block
1
finally : i execute always.
```

**Output:**

```
Enter a and b values:4 0
inside try block
Arithmetic Exception
finally : i execute always.
```

**c) Write a JAVA program for creation of Java Built-in Exceptions**

**Program:**

```
class MultiCatch {  
    public static void main(String args[]) {  
        int no,d,r;  
        no = d = r = 0;  
        try {  
            no = Integer.parseInt(args[0]);  
            d = Integer.parseInt(args[1]);  
            r = no/d;  
            System.out.println("result = "+r);  
        } catch(ArrayIndexOutOfBoundsException e1) {  
            System.out.println("Entered less than two values");  
        } catch(NumberFormatException e2) {  
            System.out.println("Conversion of alphabets to int not  
possible");  
        } catch(ArithmeticException e3) {  
            System.out.println("division by zero");  
        } catch(Exception e) {  
            System.out.println(e);  
        }  
    }  
}
```

**Output-1:**

```
>java MultiCatch  
Entered less than two values
```

**Output-2:**

```
>java MultiCatch 5 4  
result = 1
```

**output-3:**

```
>java MultiCatch 5 0  
division by zero
```

**output-4:**

```
>java MultiCatch 5 a  
Conversion of alphabets to int not possible
```

**d)Write a JAVA program for creation of User Defined Exception**

**Program:**

```
import java.util.Scanner;
class MinorException extends Exception {
    public MinorException(String msg) {
        super(msg);
    }
}
class SeniorException extends RuntimeException {
    public SeniorException(String msg) {
        super(msg);
    }
}
public class Main {
    public static void main(String args[]) {
        int age = 0;
        Scanner sc = new Scanner(System.in);
        try {
            System.out.print("Enter age:");
            age = sc.nextInt();
            if (age < 18)
                throw new MinorException("you are not eligible for marriage");
            else if (age >= 18 && age < 40)
                System.out.println("you are eligible for marriage");
            else
                throw new SeniorException("you are too late");
        } catch (MinorException me) {
            System.out.println(me);
        } catch (SeniorException sce) {
            System.out.println(sce);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

**Output:**

```
Enter age:18
you are eligible for marriage
```

**Output:**

```
Enter age:12
MinorException: you are not eligible for marriage
```

**Output:**

```
Enter age:60
SeniorException: you are too late
```

### Exercise – 10 (Threads)

a) Write a JAVA program that creates threads by extending Thread class. First thread display “Good Morning “every 1 sec, the second thread displays “Hello “every 2 seconds and the third display “Welcome” every 3 seconds ,(Repeat the same by implementing Runnable)

#### Program:

```
class A extends Thread {
    public void run() {
        try {
            while (true) {
                sleep(1000);
                System.out.println("good morning");
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
class B extends Thread {
    public void run() {
        try {
            while (true) {
                sleep(2000);
                System.out.println("hello");
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
class C extends Thread {
    public void run() {
        try {
            while (true) {
                sleep(3000);
                System.out.println("welcome");
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
class Exp10 {
    public static void main(String args[]) {
        A a1 = new A();
        B b1 = new B();
    }
}
```

```
        C c1 = new C();
        a1.start();
        b1.start();
        c1.start();
    }
}
```

**Output:**

good morning  
hello  
good morning  
good morning  
welcome  
hello  
good morning  
good morning  
hello  
welcome  
good morning  
good morning  
hello  
good morning

## ii) Creating multiple threads using Runnable interface

### Program:

```
class A implements Runnable {
    public void run() {
        try {
            while (true) {
                Thread.sleep(1000);
                System.out.println("good morning");
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
class B implements Runnable {
    public void run() {
        try {
            while (true) {
                Thread.sleep(2000);
                System.out.println("hello");
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
class C implements Runnable {
    public void run() {
        try {
            while (true) {
                Thread.sleep(3000);
                System.out.println("welcome");
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
class Exp10 {
    public static void main(String args[]) {
        Thread t1 = new Thread(new A());
        Thread t2 = new Thread(new B());
        Thread t3 = new Thread(new C());
        t1.start();
        t2.start();
    }
}
```

```
        t3.start();  
    }  
}
```

**Output:**

good morning  
hello  
good morning  
welcome  
good morning  
hello  
good morning  
good morning  
welcome  
hello  
good morning  
good morning  
hello  
good morning  
welcome  
good morning



**b) Write a program illustrating isAlive and join ()**

**Program:**

```
class MyRunnableClass implements Runnable {
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println(Thread.currentThread().getName() + " i - " + i);
        }
    }
}

public class Exp10_2 {
    public static void main(String[] args) {
        Thread t1 = new Thread(new MyRunnableClass(), "A");
        Thread t2 = new Thread(new MyRunnableClass(), "B");
        t1.start();
        t2.start();
        System.out.println("A Alive - " + t1.isAlive());
        System.out.println("B Alive - " + t2.isAlive());
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            System.out.println(e);
        }
        System.out.println("A Alive - " + t1.isAlive());
        System.out.println("B Alive - " + t2.isAlive());
        System.out.println("Processing finished");
    }
}
```

**Output:**

```
A Alive - true
B Alive - true
A i - 0
A i - 1
A i - 2
A i - 3
B i - 0
B i - 1
A i - 4
B i - 2
B i - 3
B i - 4
A Alive - false
B Alive - false
Processing finished
```

**c) Write a Program illustrating Daemon Threads.**

**Program:**

```
class Thread1 extends Thread {
    public void run() {
        if (Thread.currentThread().isDaemon()) {
            System.out.println("Daemon thread executing");
        } else {
            System.out.println("user(normal) thread executing");
        }
    }

    public static void main(String[] args) {
        Thread1 t1 = new Thread1();
        Thread1 t2 = new Thread1();
        t1.setDaemon(true);
        t1.start();
        t2.start();
    }
}
```

**Output:**

Daemon thread executing  
user(normal) thread executing

### **Exercise - 11 (Threads continuity)**

#### **a) Write a JAVA program Producer Consumer Problem**

##### **Program:**

```
import java.util.Scanner;
class Buffer {
    String data;
    boolean avail = false;
    public synchronized void put(String data) {
        while (avail == true) {
            try {
                wait();
            } catch (InterruptedException ie) {
                System.out.println(ie);
            }
        }
        this.data = data;
        System.out.println("Produced : " + data);
        avail = true;
        notify();
    }
    public synchronized String get() {
        while (avail == false) {
            try {
                wait();
            } catch (InterruptedException ie) {
                System.out.println(ie);
            }
        }
        avail = false;
        notify();
        return data;
    }
}
class Producer extends Thread {
    String data;
    Scanner s = new Scanner(System.in);
    Buffer buf;
    public Producer(Buffer buf) {
        super("Producer");
        this.buf = buf;
    }
    public void run() {
        try {
            while (true) {
                System.out.println("Enter data");
                data = s.nextLine();
                buf.put(data);
                Thread.sleep(500);
            }
        } catch (Exception e) {
```

```

        System.out.println(e);
    }
}
}
class Consumer extends Thread {
    Buffer buf;
    public Consumer(Buffer buf) {
        super("Consumer");
        this.buf = buf;
    }
    public void run() {
        try {
            while (true) {
                System.out.println("Consumed: " + buf.get());
                Thread.sleep(500);
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
}
class MainDemo {
    public static void main(String args[]) {
        Buffer buf = new Buffer();
        Producer p = new Producer(buf);
        Consumer c = new Consumer(buf);
        p.start();
        c.start();
    }
}

```

**Output:**

```

Enter data
BESTIES
Produced : BESTIES
Consumed: BESTIES
Enter data
12
Produced : 12
Consumed: 12
Enter data
13
Produced : 13
Consumed: 13
Enter data
10
Produced : 10
Consumed: 10

```

**b) Write a case study on thread Synchronization after solving the above producer consumer problem**

The fact that a thread can execute in parallel with other threads is the main power of the concurrent programming. But this property causes usually a lot of trouble for programmers which must assure a correct sharing of the resources used in common by different threads. It means that the threads that execute in parallel and use common critical resources must be synchronized at some points to assure a correct functioning of the system. The correct functioning of a parallel or concurrent program is possible only if the conflicts of simultaneously performing a critical section operation by several threads are avoided. A critical section operation is an operation (a portion of the program code) that cannot be done in parallel by several processes or threads, like incrementing a shared counter, writing in a shared buffer, modifying a shared object, etc.

The synchronization means that a thread must wait for another thread to leave the critical section and to enter into this section using some security measures, e.g. locking the critical section when entering it. To show a situation when the synchronization is strictly necessary considers the following program. The program creates two threads that use a shared object to count the total number of iterations done by both threads.

You can use wait, notify and notifyAll methods to communicate between threads in Java. For example, if you have two threads running in your program e.g. Producer and Consumer then producer thread can communicate to the consumer that it can start consuming now because there are items to consume in the queue. Similarly, a consumer thread can tell the producer that it can also start putting items now because there is some space in the queue, which is created as a result of consumption. A thread can use wait() method to pause and do nothing depending upon some condition. For example, in the [producer-consumer problem](#), producer thread should wait if the queue is full and consumer thread should wait if the queue is empty.

If some thread is waiting for some condition to become true, you can use notify and notifyAll methods to inform them that condition is now changed and they can wake up. Both notify() and notifyAll() method sends a notification but notify sends the notification to only one of the waiting thread, no guarantee which thread will receive notification and notifyAll() sends the notification to all threads. So if only one thread is waiting for an object lock, also known as a monitor then both notify and notifyAll will send the notification to it. If [multiple threads](#) are waiting on a monitor then notify will only inform one of the lucky thread and rest will not receive any notification, but notifyAll will inform all threads.

### **Exercise – 12 (Packages)**

**a) Write a JAVA program illustrate class path**

**Program:**

```
package mypack;  
  
public class Factorial {  
    public int fact(int a) {  
        if (a == 1)  
            return 1;  
        else  
            return a * fact(a - 1);  
    }  
}
```

**Output:**

**compile the program of the specific class path**

This program will not run it does not contain main method.

**b) Write a case study on including in class path in your os environment of your package.**

Classpath is used for storing the path of the third-party and user-defined classes. Whenever we execute/compile any class file, jdk tools javac and java, search the package/class file in the user classpath which is the current directory by default. If the classes are not in the current directory, then we need to set the classpath.

The classpath can be set in two ways:

1. It is an environment variable which can be set using the *System* utility in the control panel or at the DOS prompt as shown.

```
Set CLASSPATH = %CLASSPATH%;c:\pack;
```

%Classpath% is used to keep the existing path intact and append our new path to it. Now L3 of both the above cases will execute.

2. Use classpath option `-classpath` or `-cp` of javac/java tools to override the user-defined classpath and find the user-defined specific package/classes used in the Java source files.

```
//syntax: javac -cp path of the directory/package used in java  
source file  
followed by name of the java source file  
C:\pack\packexample> javac -cp c:\javaeg DemoClass.java
```

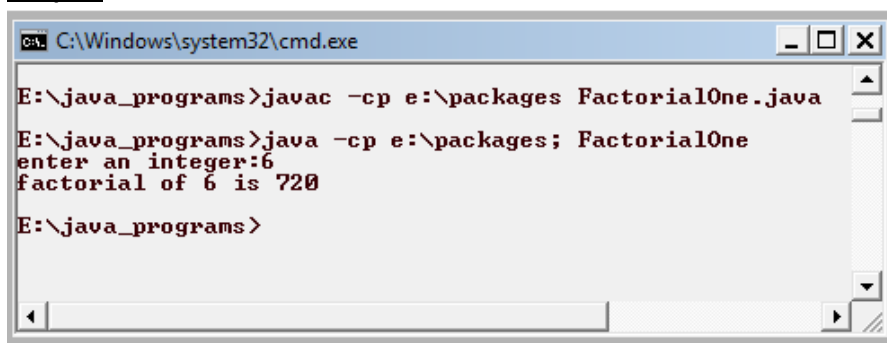
`-cp` specifies that the user-defined package/classes used in DemoClass.java will be found at c:\javaeg.

c) Write a JAVA program that import and use the defined your package in the previous Problem

**Program:**

```
import mypack.*;
import java.util.Scanner;
class FactorialOne {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("enter an integer:");
        int a = sc.nextInt();
        Factorial i = new Factorial();
        System.out.println("factorial of " + a + " is " + i.fact(a));
    }
}
```

**Output:**



```
C:\Windows\system32\cmd.exe

E:\java_programs>javac -cp e:\packages FactorialOne.java
E:\java_programs>java -cp e:\packages; FactorialOne
enter an integer:6
factorial of 6 is 720
E:\java_programs>
```