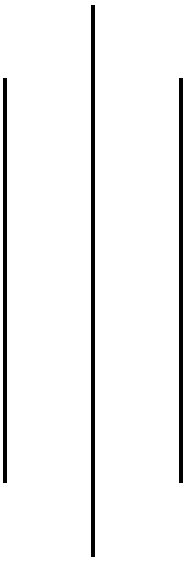
ORCHID INTERNATIONAL College

(Affiliated to Tribhuvan University)

Advanced Java Programming

Lab 004

Packaging and Multithreading



Submitted by:

Abhinna Ojha, 20788/075

BSc. CSIT - VII

Submitted to:

Mr. Krishna Pandey

Department of CSIT

1. Interfaces

1.1. Write a program to define a queue interface and have insert and delete methods in the interface. Implement these methods in a class.

```
/*
   Title:
      Interfaces

   Description:
      Write a program to define a queue interface and have insert and delete methods
      in the interface. Implement these methods in a class.

   Date modified; Author(s); Modification details
      2022-12-28; abhinna; Created the program
*/

import java.util.Scanner;

// class of individual members of queue
class QueueValue {
   int value;
   QueueValue next;
   public QueueValue(int value) {
      this.value = value;
      next = null;
   } // QueueValue(int value)
} // class QueueValue

// an interface to implement insert and delete methods
interface IQueue {
   void insert(QueueValue q);
   void delete();
} // interface IQueue

// implementation of interface on class
class Queue implements IQueue {
//    front and back pointers of the queue to keep track of the front and back of queue
//    for insert and delete
   QueueValue front, back;

//    initialising queue to be empty queue at the beginning
   public Queue() {
      front = null;
      back = null;
   } // public Queue()

//    overriding interface methods
   @Override
   public void insert(QueueValue q) {
```

```
//        if front = null, it is an empty queue
//        so, front and back are the newly inserted value
        if (front == null) {
            front = q;
            back = q;
        } // if (front == null)
        else {
//        replace the null on next pointer of current back to newly inserted queue value
//        then replace the current back value with newly inserted queue value
//            back.next = q; back = q; is also viable
            back.next = q;
            back = back.next;
        } // else of if (front == null)
    } // public void insert(QueueValue q)

    @Override
    public void delete() {
//        to delete, just replace the front pointer with the next of front so front changes
//        front defines the start of queue, so the old front cannot be accessed, thus is deleted
        front = front.next;

//        if front = null then queue becomes empty so assign back as null too
        if (front == null) {
            back = null;
        } // if (front == null)
    } // public void delete()

    public void display() {
        QueueValue i = front;

//        displaying values from front
//        once the next pointer of a value is null, queue has reached its end
//        also, once i = back the queue reaches is end, so it can be implemented that way too
        while (i != null) {
            System.out.println(i.value);
            i = i.next;
        } // while (i != null)
    } // public void display()
} // class Queue implements IQueue

// main class that has the main
public class Q1Interfaces {
    public static void main(String[] args) {
//        creating array of queue values for easier manipulation
//        making array of large size to avoid dynamic sizing for now
        QueueValue[] queueValue = new QueueValue[100];

//        initiating a counter to point to array position, updates on insert only
        int x = 0;
```

```
//        creating the queue
        Queue queue = new Queue();

//        defining infinite flag for while based infinite loop
        boolean infiniteFlag = true;

//        infinite loop until exit is hit
        while (infiniteFlag) {
            System.out.println("\n1.insert");
            System.out.println("2.delete");
            System.out.println("3.display");
            System.out.println("0.exit");
            System.out.print("Make your choice: ");

            int choice;
            Scanner scanner = new Scanner(System.in);
//            scanner may have exception if user inputs character or decimals instead of integers
            try {
                choice = Integer.parseInt(scanner.nextLine());
                switch (choice) {
                    case 0 ->
                            infiniteFlag = false;
                    case 1 -> {
                        System.out.print("Enter integer to insert: ");
                        // try for input of value for QueueValue(value)
                        try {
                            int value = Integer.parseInt(scanner.nextLine());
                            queueValue[x] = new QueueValue(value);
                            queue.insert(queueValue[x]);
                            x++;
                        } // try of value for QueueValue(value)
                        catch (Exception exception) {
                            System.out.println("Invalid input");
                        } // catch
                    } // case 1

                    case 2 ->
                        queue.delete();
                    // case 2
                    case 3 ->
                        queue.display();
                    // case 3
                } // switch (choice)
            } // try of choice
            catch (Exception exception) {
                System.out.println("Invalid choice.");
            } // catch (Exception exception)
        } // while (infiniteFlag)
    } // public static void main(String[] args)
} // public class Q1Interfaces
```

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2.4\lib\idea_rt.jar=10118:C:\Program Files

1.insert
2.delete
3.display
0.exit
Make your choice: 1
Enter integer to insert: 4

1.insert
2.delete
3.display
0.exit
Make your choice: 1
Enter integer to insert: 7

1.insert
2.delete
3.display
0.exit
Make your choice: 1
Enter integer to insert: 5
```

```
Enter integer to insert: 5

1.insert
2.delete
3.display
0.exit
Make your choice: 1
Enter integer to insert: 4

1.insert
2.delete
3.display
0.exit
Make your choice: 3
4
7
5
4

1.insert
2.delete
3.display
0.exit
```

```
0.exit
Make your choice: 2

1.insert
2.delete
3.display
0.exit
Make your choice: 3
7
5
4

1.insert
2.delete
3.display
0.exit
Make your choice: 2

1.insert
2.delete
3.display
0.exit
Make your choice: 3
```

```
Make your choice: 3
5
4

1.insert
2.delete
3.display
0.exit
Make your choice: 1
Enter integer to insert: 3

1.insert
2.delete
3.display
0.exit
Make your choice: 3
5
4
3

1.insert
2.delete
3.display
0.exit
```

```
2.delete
3.display
0.exit
Make your choice: 1
Enter integer to insert: 3

1.insert
2.delete
3.display
0.exit
Make your choice: 3
5
4
3

1.insert
2.delete
3.display
0.exit
Make your choice: 0

Process finished with exit code 0
```

2. Packages

2.1. Write a program to define functions for subtract, multiply, divide, factorial and reversing the digits of a number in a package, import this class in another package and use all the methods defined in the primary package.

```
/*
   Title:
      Packages

   Description:
      Write a program to define functions for subtract, multiply, divide, factorial and
      reversing the digits of a number in a package, import this class in another package
      and use all the methods defined in the primary package.

   Date modified; Author(s); Modification details
      2022-12-28; abhinna; Created the program
*/

// importing the package to use
import com.Lab004.ArithmeticOperations.*;

import java.util.Scanner;

public class Q2Packages {
   public static void main(String[] args) {
//        flag for infinite loop
      boolean infiniteFlag = true;

      while (infiniteFlag){
         System.out.println("\n1. Subtract");
         System.out.println("2. Multiply");
         System.out.println("3. Divide");
         System.out.println("4. Factorial");
         System.out.println("5. Reverse digits");
         System.out.println("0. Exit");
         System.out.print("Make your choice: ");

         int choice;
         Scanner scanner = new Scanner(System.in);

//          scanner is for integer only
         try {
            choice = Integer.parseInt(scanner.nextLine());
            switch (choice) {
               case 0 -> infiniteFlag = false;
                // case 0
```

```
//                  Subtract 2 numbers
                  case 1 -> {
//                      scanner for double only
                     try {
                        System.out.print("\nEnter a for a - b: ");
                        double a = Double.parseDouble(scanner.nextLine());

                        System.out.print("\nEnter b for a - b: ");
                        double b = Double.parseDouble(scanner.nextLine());

                        System.out.println(a + " - " + b + " = " + ArithmeticOperations.subtract(a,
b));
                     } // try
                     catch (Exception exception) {
                        System.out.println("Error in input.");
                     } // catch
                  } // case 1

//                  Multiply 2 numbers
                  case 2 -> {
//                      scanner for double only
                     try {
                        System.out.print("\nEnter a for a * b: ");
                        double a = Double.parseDouble(scanner.nextLine());

                        System.out.print("\nEnter b for a * b: ");
                        double b = Double.parseDouble(scanner.nextLine());

                        System.out.println(a + " * " + b + " = " + ArithmeticOperations.multiply(a,
b));
                     } // try
                     catch (Exception exception) {
                        System.out.println("Error in input.");
                     } // catch
                  } // case 2

//                  Multiply 2 numbers
                  case 3 -> {
//                      scanner for double only
                     try {
                        System.out.print("\nEnter a for a / b: ");
                        double a = Double.parseDouble(scanner.nextLine());

                        System.out.print("\nEnter b for a / b: ");
                        double b = Double.parseDouble(scanner.nextLine());

//                         error handling for divide by 0 by not allowing the error state to reach
                        if (b == 0) {
                           System.out.println("Divide by 0 error");
```

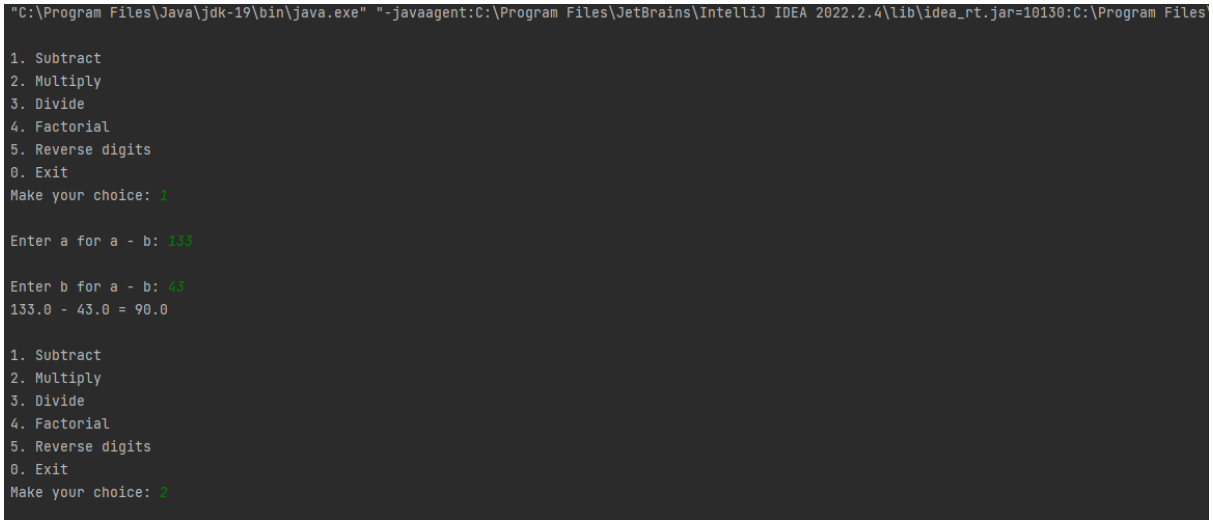
```
                    } // if (b == 0)
                    else {
                        System.out.println(a + " / " + b + " = " + ArithmeticOperations.divide(a,
b));
                    } // else of if (b == 0)
                } // try
                catch (Exception exception) {
                    System.out.println("Error in input.");
                } // catch
            } // case 3

//              Factorial
            case 4 -> {
//                  scanner for int only
                try {
                    System.out.print("\nEnter a for a!: ");
                    int a = Integer.parseInt(scanner.nextLine());

//                      error handling for -ve numbers by not allowing the error state to reach
                    if (a < 0) {
                        System.out.println("Factorial is only allowed for +ve numbers");
                    } // if (a < 0)
                    else {
                        System.out.println(a + "! " + " = " + ArithmeticOperations.factorial(a));
                    } // else of if (a < 0)
                } // try
                catch (Exception exception) {
                    System.out.println("Error in input.");
                } // catch
            } // case 4

//              Reverse
            case 5 -> {
//                  scanner for int only
                try {
                    System.out.print("\nEnter a for a's reverse digits: ");
                    int a = Integer.parseInt(scanner.nextLine());

//                      error handling for -ve numbers by not allowing the error state to reach
                    if (a < 0) {
                        System.out.println("Reverse is only allowed for +ve numbers");
                    } // if (a < 0)
                    else {
                        System.out.println(a + "'s reverse " + " is " +
ArithmeticOperations.reverse(a));
                    } // else of if (a < 0)
                } // try
                catch (Exception exception) {
                    System.out.println("Error in input.");
                } // catch
```

```
                } // case 5

            } // switch (choice)
        } // try
        catch (Exception exception) {
            System.out.println("Error in input.");
        } // catch
    } // while (infiniteFlag)
  } // public static void main(String[] args)
} // public class Q2Packages
```

com.Lab004.ArithmeticOperations

```
package com.Lab004.ArithmeticOperations;

public class ArithmeticOperations {
//    all methods are made static so that they can be used without instantiation
//    subtract operation
  public static double subtract(double a, double b) {
      return (a - b);
  } // public double subtract(double a, double b)

//    multiply operation
  public static double multiply(double a, double b) {
      return (a * b);
  } // public double multiply(double a, double b)

//    divide operation
  public static double divide(double a, double b) {
      return (a / b);
  } // public double divide(double a, double b)

//    factorial of a number
  public static double factorial(int a) {
      double fact = 1;
      if (a != 0 && a != 1) {
          for (int i = 1; i <= a; i++) {
              fact *= i;
          } // for (int i = 0; i <= a; i++)
      } // if (a == 0 || a == 1)

      return fact;
  } // public double factorial(int a)

//    reverse digits of number
  public static int reverse(int a) {
      int rev = 0, x = a;
```

```
        while (x != 0) {
            rev = rev * 10 + x % 10;
            x /= 10;
        } // while (a != 0)

        return  rev;
    } // public int reverse(int i)
} // public class ArithmeticOperations
```



```
4. Factorial
5. Reverse digits
0. Exit
Make your choice: 3

Enter a for a / b: 13

Enter b for a / b: 5
13.0 / 5.0 = 2.6

1. Subtract
2. Multiply
3. Divide
4. Factorial
5. Reverse digits
0. Exit
Make your choice: 4

Enter a for a!: -1
Factorial is only allowed for +ve numbers

1. Subtract
2. Multiply
```

```
2. Multiply
3. Divide
4. Factorial
5. Reverse digits
0. Exit
Make your choice: 6

1. Subtract
2. Multiply
3. Divide
4. Factorial
5. Reverse digits
0. Exit
Make your choice: 4

Enter a for a!: 6
6!  = 720.0

1. Subtract
2. Multiply
3. Divide
4. Factorial
5. Reverse digits
```

```
6!  = 720.0

1. Subtract
2. Multiply
3. Divide
4. Factorial
5. Reverse digits
0. Exit
Make your choice: 5

Enter a for a's reverse digits: 659
659's reverse  is 956

1. Subtract
2. Multiply
3. Divide
4. Factorial
5. Reverse digits
0. Exit
Make your choice: 0

Process finished with exit code 0
```

3. Exception

3.1. Write a program to demonstrate ArrayIndexOutOfBoundsException.

```
/*
   Title:
      Exception

   Description:
      Write a program to demonstrate ArrayIndexOutOfBoundsException.

   Date modified; Author(s); Modification details
      2022-12-28; abhinna; Created the program
*/

public class Q3Exception {
   public static void main(String[] args) {
//       creating an array of size 6 with values
      int[] arr = {13, 22, 34, 84, 51, 60};

//       running a loop of 10
      for (int i = 0; i < 10; i++) {
//           displaying array
//           but, there may be exception so using try-catch block too
         try {
            System.out.println(arr[i]);
         } // try
//          catching Array Index Out Of Bounds Exception
         catch (ArrayIndexOutOfBoundsException arrayIndexOutOfBoundsException) {
            System.out.println("ArrayIndexOutOfBoundsException occurred" +
arrayIndexOutOfBoundsException);
         } // catch (ArrayIndexOutOfBoundsException arrayIndexOutOfBoundsException)
      } // for (int i = 0; i < 10; i++)
   } // public static void main(String[] args)
} // public class Q3Exception
```

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2.4\lib\idea_rt.jar=10150:C:\Program File
13
22
34
84
51
60
ArrayIndexOutOfBoundsException occurredjava.lang.ArrayIndexOutOfBoundsException: Index 6 out of bounds for length 6
ArrayIndexOutOfBoundsException occurredjava.lang.ArrayIndexOutOfBoundsException: Index 7 out of bounds for length 6
ArrayIndexOutOfBoundsException occurredjava.lang.ArrayIndexOutOfBoundsException: Index 8 out of bounds for length 6
ArrayIndexOutOfBoundsException occurredjava.lang.ArrayIndexOutOfBoundsException: Index 9 out of bounds for length 6

Process finished with exit code 0
```

4. Thread

4.1. Write a program to print tables of 5 by creating a new thread and display 20 even numbers as a task of main thread.

```
/*
   Title:
      Thread

   Description:
      Write a program to print tables of 5 by creating a new thread and display 20 even
      numbers as a task of main thread.

   Date modified; Author(s); Modification details
      2022-12-28; abhinna; Created the program
*/

public class Q4Threads {
   public static void main(String[] args) {
//       creating object of TableOf5 to pass onn thread
      TableOf5 tableOf5 = new TableOf5();
//       passing tableOf5 on Thread class to create a thread
      Thread threadTable = new Thread(tableOf5);

//       creating main thread to call the Table of 5 and display 20 even numbers
      Thread mainThread = new Thread() {
//          overriding run is must
         @Override
         public void run() {
            int i = 0, even = 0;
//             calling child thread from main thread
            threadTable.start();

//             printing 20 even numbers b first feeding an even number, then increasing it by 2
            while (i < 20) {
               System.out.println(even + " is even.");
               i++;
               even += 2;
//                exception handling for Thread.sleep()
               try {
//                   using Thread.sleep(300) to display at 0.3 second intervals for better output
                  Thread.sleep(300);
               } // try
               catch (InterruptedException e) {
                  throw new RuntimeException(e);
               } // catch (InterruptedException e)
            } //  while (i < 20)
         } // public void run()
      }; // Thread mainThread = new Thread()
```

```
//         starting main thread
        mainThread.start();
    } // public static void main(String[] args)
} // public class Q4Threads


// creating a runnable thread class by implementing Runnable interface
class TableOf5 implements Runnable
{
//     overriding run method of Runnable interface
    @Override
    public void run() {
//         print table of 5 from thread
        for (int i = 1; i <= 10; i++) {
            System.out.println("5 * " + i + " = " + (5 * i));
//             exception handling for Thread.sleep()
            try {
//                 using Thread.sleep(300) to display at 0.3 second intervals for better output
                Thread.sleep(300);
            } // try
            catch (InterruptedException e) {
                throw new RuntimeException(e);
            } // catch (InterruptedException e)
        } // for (int i = 1; i <= 10; i++)
    } // public void run()
} // class Thread1 implements Runnable
```

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2.4\lib\idea_rt.jar=10152:C:\Program Files
0 is even.
5 * 1 = 5
2 is even.
5 * 2 = 10
4 is even.
5 * 3 = 15
6 is even.
5 * 4 = 20
8 is even.
5 * 5 = 25
10 is even.
5 * 6 = 30
12 is even.
5 * 7 = 35
14 is even.
5 * 8 = 40
16 is even.
5 * 9 = 45
18 is even.
5 * 10 = 50
20 is even.
22 is even.
```

```
22 is even.
24 is even.
26 is even.
28 is even.
30 is even.
32 is even.
34 is even.
36 is even.
38 is even.

Process finished with exit code 0
```