403 - Java Programming

Index

403 - Java Programming

Q1. Write a program to accept 5 command line arguments and raise a custom exception if any argument is not from the list ("BCA","MCA","BBA","MBA","OTHER").

```
package practical_assignment;
import java.util.Scanner;
public class P1 {

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String[] arr = new String[5];

      for (int i=0;i<5;i++){
         System.out.print("Enter string " + (i + 1) + " : ");
         checkCourse(sc.next());
      }

      System.out.println("All course inputs are valid");
   }

   static void checkCourse(String s) {
      if (!(s.equalsIgnoreCase("BCA") ||
            s.equalsIgnoreCase("MCA") ||
            s.equalsIgnoreCase("BBA") ||
            s.equalsIgnoreCase("MBA") ||
            s.equalsIgnoreCase("OTHER"))) {

         throw new CourseException("Invalid Course: " + s);
      }
   }
}

/* Custom Exception */
class CourseException extends RuntimeException {
   CourseException(String message) {
      super(message);
   }
}
```

```
Enter string 1 : bca
Enter string 2 : bba
Enter string 3 : mba
Enter string 4 : mca
Enter string 5 : abc
Exception in thread "main" practical_assignment.CourseException Create breakpoint : Invalid Course: abc
    at practical_assignment.P1.checkCourse(P1.java:32)
    at practical_assignment.P1.main(P1.java:19)
```

Q2. Write a program to demonstrate throw and throws keyword.

```
package practical_assignment;
public class P2 {

   static void check(int age) throws ArithmeticException {
      if (age < 18)
         throw new ArithmeticException("Not eligible");
      else
         System.out.println("Eligible");
   }

   public static void main(String[] args) {
      try {
         check(16);
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

```
java.lang.ArithmeticException: Not eligible


Process finished with exit code 0
```

Q3. Write a program to demonstrate custom exception called NagetiveNumberArgumentException.

```
package practical_assignment;

import java.util.Scanner;

class NagetiveNumberArgumentException extends Exception {
   NagetiveNumberArgumentException(String message) {
      super(message);
   }
}
public class P3 {
   static void checkNumber(int num) throws NagetiveNumberArgumentException {
      if (num < 0) {
         throw new NagetiveNumberArgumentException("Negative number is not allowed");
      } else {
         System.out.println("Valid number: " + num);
      }
   }

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      try {
         System.out.print("Enter a number: ");
         int number = sc.nextInt();
         checkNumber(number);
      } catch (NagetiveNumberArgumentException e) {
         System.out.println(e);
      } catch (Exception e) {
         System.out.println("Please provide a valid integer");
      }
   }
}
```

```
Enter a number: -20
practical_assignment.NagetiveNumberArgumentException: Negative number is not allowed
```

Q4. Write a program to input strings and display those which start with 'A' or 'a' using Scanner.

```
package practical_assignment;
import java.util.Scanner;

public class P4 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("How many strings? ");
      int n = sc.nextInt();
      sc.nextLine();



      for (int i = 0; i < n; i++) {
         System.out.print("Enter strings:");
         String s = sc.nextLine();

         if (s.startsWith("A") || s.startsWith("a")) {
            System.out.println(s);
         }
      }
   }
}
```

```
How many strings? 2
Enter strings:Akash
Akash
Enter strings:yash
```

Q5. Write a program to input two strings search similar characters from both string and replace it with '*'.

```
package practical_assignment;
import java.util.Scanner;

public class P5 {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      System.out.print("Enter first string: ");
      String s1 = sc.nextLine();

      System.out.print("Enter second string: ");
      String s2 = sc.nextLine();

      char[] a = s1.toCharArray();
      char[] b = s2.toCharArray();

      for (int i = 0; i < a.length; i++) {
         for (int j = 0; j < b.length; j++) {
            if (a[i] == b[j]) {
               a[i] = '*';
               b[j] = '*';
            }
         }
      }

      System.out.println("Modified first string: " + new String(a));
      System.out.println("Modified second string: " + new String(b));

      sc.close();
   }
}
```

```
Enter first string: yash
Enter second string: harsh
Modified first string: y***
Modified second string: **r*h
```

Q6. Write a java application which accepts two strings. Merge both the strings using alternate characters of each one. For example: If String1 is: "Very", and String2 is: "Good". Then result should be: "VGeoroyd". in java.

```
package practical_assignment;
import java.util.Scanner;

public class P6 {
   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      System.out.print("Enter first string: ");
      String s1 = sc.nextLine();

      System.out.print("Enter second string: ");
      String s2 = sc.nextLine();

      StringBuilder result = new StringBuilder();

      int i = 0;
      while (i < s1.length() || i < s2.length()) {

         if (i < s1.length()) {
            result.append(s1.charAt(i));
         }

         if (i < s2.length()) {
            result.append(s2.charAt(i));
         }

         i++;
      }
      System.out.println("Merged String: " + result);
   }
}
```

```
Enter first string: aceg
Enter second string: bdfh
Merged String: abcdefgh
```

Q7. Create package stores. Under it create a class called stock with member variable (item_no, item_name, stock_availible, and cost). Under the default package create a class called sales with field name (qty_sold) and it is the child class of stores class. Write a program to print the current stock of each item and perform addition.

```
package practical_assignment;

public class P7 {
   public static void main(String[] args) {

      Sales s = new Sales(101, "Gold Ring", 50, 2500, 10);

      s.displayStock();
      s.printCurrentStock();
      s.addition();
   }
}
```

```
package practical_assignment;
import practical_assignment.stores.Stock;

/*
 Sales class in default package
 Child class of Stock
*/

public class Sales extends Stock {

   int qty_sold;

   public Sales(int item_no, String item_name, int stock_available, double cost, int qty_sold) {
      super(item_no, item_name, stock_available, cost);
      this.qty_sold = qty_sold;
   }

   void printCurrentStock() {
      int currentStock = stock_available - qty_sold;
      System.out.println("Quantity Sold: " + qty_sold);
      System.out.println("Current Stock: " + currentStock);
   }

   void addition() {
      double totalAmount = qty_sold * cost;
      System.out.println("Total Sales Amount: " + totalAmount);
   }

}
```

```
package practical_assignment.stores;

/*
 Stock class inside stores package
*/

public class Stock {

    protected int item_no;
    protected String item_name;
    protected int stock_available;
    protected double cost;

    public Stock(int item_no, String item_name, int stock_available, double cost) {
        this.item_no = item_no;
        this.item_name = item_name;
        this.stock_available = stock_available;
        this.cost = cost;
    }

    public void displayStock() {
        System.out.println("Item No: " + item_no);
        System.out.println("Item Name: " + item_name);
        System.out.println("Stock Available: " + stock_available);
        System.out.println("Cost per Item: " + cost);
    }
}
```

```
Item No: 101
Item Name: Gold Ring
Stock Available: 50
Cost per Item: 2500.0
Quantity Sold: 10
Current Stock: 40
Total Sales Amount: 25000.0
```

Q8. Write an application that executes three threads from one thread class. One thread displays "JAVA" every 1 second. another display "PAPER" every 2 seconds and last one displays "COLLEGE"every 3 seconds. Create the thread by using Runnable Interface.

```
package practical_assignment;
class PrintMessage implements Runnable {

   String message;
   int time;

   PrintMessage(String message, int time) {
      this.message = message;
      this.time = time;
   }

   public void run() {
      try {
         while (true) {
            System.out.println(message);
            Thread.sleep(time);
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}

public class P8 {
   public static void main(String[] args) {

      Thread t1 = new Thread(new PrintMessage("JAVA", 1000));
      Thread t2 = new Thread(new PrintMessage("PAPER", 2000));
      Thread t3 = new Thread(new PrintMessage("COLLEGE", 3000));

      t1.start();
      t2.start();
      t3.start();
   }
}
```

```
JAVA
COLLEGE
PAPER
JAVA
JAVA
PAPER
COLLEGE
JAVA
PAPER
JAVA
JAVA
PAPER
```

Q9. Write a java code which accepts name a of 10 students. Sort the names of students in ascending order. Display the names of students using thread class at interval of one seconds.

```
package practical_assignment;
import java.util.Scanner;
import java.util.Arrays;

class DisplayNames extends Thread {

   String[] names;

   DisplayNames(String[] names) {
      this.names = names;
   }

   public void run() {
      try {
         for (String name : names) {
            System.out.println(name);
            Thread.sleep(1000);   // 1 second delay
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}

public class P9 {
   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);
      String[] names = new String[10];

      System.out.println("Enter names of 10 students:");

      for (int i = 0; i < 10; i++) {
         names[i] = sc.nextLine();
      }

      Arrays.sort(names);   // Ascending order
      DisplayNames t = new DisplayNames(names);
      t.start();
      sc.close();
   }
}
```

```
walter white
lalo salamanca
Ned stark
Gustavo fring
eleven
professor
mike
robert
robb stark
thomas shelby
Gustavo fring
Ned stark
eleven
lalo salamanca
mike
professor
robb stark
robert
thomas shelby
walter white
```

Q10. Write a program to demonstrate thread priorities.

```
package practical_assignment;

class PriorityThread extends Thread {

   public void run() {
      System.out.println(
            "Thread Name: " + getName() +
                  " | Priority: " + getPriority()
      );
   }
}

public class P10 {
   public static void main(String[] args) {

      PriorityThread t1 = new PriorityThread();
      PriorityThread t2 = new PriorityThread();
      PriorityThread t3 = new PriorityThread();

      t1.setName("Low Priority Thread");
      t2.setName("Normal Priority Thread");
      t3.setName("High Priority Thread");

      t1.setPriority(Thread.MIN_PRIORITY);     // 1
      t2.setPriority(Thread.NORM_PRIORITY);    // 5
      t3.setPriority(Thread.MAX_PRIORITY);     // 10

      t1.start();
      t2.start();
      t3.start();
   }
}
```

```
Thread Name: High Priority Thread | Priority: 10
Thread Name: Normal Priority Thread | Priority: 5
Thread Name: Low Priority Thread | Priority: 1
```

Q11. Write a java code (application) which display current Date and Time Use thread class to execute the code.

```
package practical_assignment;
import java.util.Date;

class DateTimeThread extends Thread {

   public void run() {
      Date d = new Date();
      System.out.println("Current Date and Time: " + d);
   }
}
public class P11 {

   public static void main(String[] args) {

      DateTimeThread t = new DateTimeThread();
      t.start();
   }
}
```

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaage
Current Date and Time: Sat Feb 07 10:37:33 IST 2026

Process finished with exit code 0
```

Q12. Write a java application which accepts 10 names of student and their age. Sort names and age in descending order. Display the names of students using thread class at interval of one second.

```
package practical_assignment;
import java.util.Scanner;
import java.util.Arrays;


class Student {
   String name;
   int age;

   Student(String name, int age) {
      this.name = name;
      this.age = age;
   }
}

class DisplayStudents extends Thread {

   Student[] students;

   DisplayStudents(Student[] students) {
      this.students = students;
   }

   public void run() {
      try {
         for (Student s : students) {
            System.out.println(s.name + " - " + s.age);
            Thread.sleep(1000);   // 1second delay
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}
public class P12 {
   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);
      Student[] students = new Student[10];

      System.out.println("Enter name and age of 10 students:");

      for (int i = 0; i < 10; i++) {
```

```
            System.out.print((i+1)+" Name: ");
            String name = sc.nextLine();
            System.out.print((i+1)+" Age: ");
            int age = sc.nextInt();
            sc.nextLine();   // consume newline
            students[i] = new Student(name, age);
        }

        // Sort in descending order of age
        Arrays.sort(students, (a, b) -> b.age - a.age);

        DisplayStudents t = new DisplayStudents(students);
        t.start();

        sc.close();
    }
}
```

```
Enter name and age of 10 students:
1 Name: walter
1 Age: 52
2 Name: robb
2 Age: 25
3 Name: ned
3 Age: 48
4 Name: lalo
4 Age: 35
5 Name: gustavo
5 Age: 50
6 Name: mike
6 Age: 65
7 Name: elevel
7 Age: 11
8 Name: luffy
8 Age: 20
9 Name: zoro
9 Age: 25
10 Name: oden
10 Age: 40
```

```
mike - 65
walter - 52
gustavo - 50
ned - 48
oden - 40
lalo  - 35
robb - 25
zoro - 25
luffy - 20
elevel - 11
```

Q13. Write a java program that creates Singly Link List to perform create, insert, delete and display node using menu driven program.

```
package practical_assignment;
import java.util.Scanner;

class Node {
   int data;
   Node next;

   Node(int data) {
      this.data = data;
      this.next = null;
   }
}

public class P13 {
   static Node head = null;

   // Create / Insert at beginning
   static void insert(int data) {
      Node newNode = new Node(data);
      newNode.next = head;
      head = newNode;
      System.out.println("Node inserted");
   }

   // Delete from beginning
   static void delete() {
      if (head == null) {
         System.out.println("List is empty");
      } else {
         head = head.next;
         System.out.println("Node deleted");
      }
   }

   // Display list
   static void display() {
      if (head == null) {
         System.out.println("List is empty");
         return;
      }

      Node temp = head;
      System.out.print("Linked List: ");
      while (temp != null) {
```

```
        System.out.print(temp.data + " -> ");
        temp = temp.next;
    }
    System.out.println("NULL");
}

public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);
    int choice, value;

    do {
        System.out.println("\n--- Singly Linked List Menu ---");
        System.out.println("1. Insert Node");
        System.out.println("2. Delete Node");
        System.out.println("3. Display List");
        System.out.println("4. Exit");
        System.out.print("Enter your choice: ");

        choice = sc.nextInt();

        switch (choice) {
            case 1:
                System.out.print("Enter value: ");
                value = sc.nextInt();
                insert(value);
                break;

            case 2:
                delete();
                break;

            case 3:
                display();
                break;

            case 4:
                System.out.println("Exiting program");
                break;

            default:
                System.out.println("Invalid choice");
        }

    } while (choice != 4);
```

```
        sc.close();
    }
}
```

```
--- Singly Linked List Menu ---
1. Insert Node
2. Delete Node
3. Display List
4. Exit
Enter your choice: 1
Enter value: 5
Node inserted

--- Singly Linked List Menu ---
1. Insert Node
2. Delete Node
3. Display List
4. Exit
Enter your choice: 3
Linked List: 5 -> NULL

--- Singly Linked List Menu ---
1. Insert Node
2. Delete Node
3. Display List
4. Exit
Enter your choice: 4
Exiting program
```