KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of the UGC Act, 1956

School of Computer Engineering

WT LAB - 9

Submitted By :

Name : ISHU KUMAR

Roll No. : 2006270

Section: IT-04

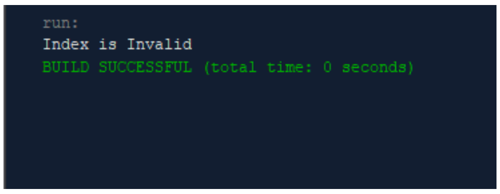
Branch : Information Technology

Q-1 Write a Java program to generate an ArrayIndexOutofBoundsException and handle it

Code :- **ISHU KUMAR 2006270**

```
 package lab.pkg9;


      public class Q1 {

    public static void main(String[] args) {
        int A[] = {30, 20, 10, 40, 0};
        try {
              System.out.println(A[5]);
           } catch (ArrayIndexOutOfBoundsException e) {
              System.out.println("Index is Invalid");
           }
    }

 }
```

OUTPUT:



```
run:
Index is Invalid
BUILD SUCCESSFUL (total time: 0 seconds)
```

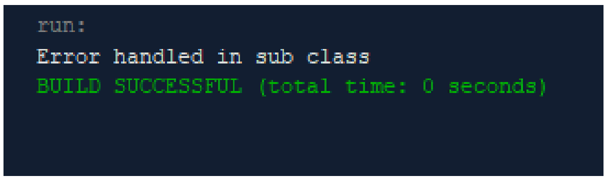
Q-2 A subclass exception must appear before super-class exception. Justify this with suitable Java programs.

Code :-

```
package lab.pkg9;

public class Q2 {

   public static void main(String[] args) {
      int A[] = {30, 20, 10, 40, 0};

      try {
         int c = A[0] / A[4];
```

```
            System.out.println("Division is " + c);
        } catch (ArithmeticException e) {
            System.out.println("Error handled in sub class");
        } catch (Exception e) {
            System.out.println("Error handled in super class");
        }
    }

}
```

OUTPUT:



```
run:
Error handled in sub class
BUILD SUCCESSFUL (total time: 0 seconds)
```

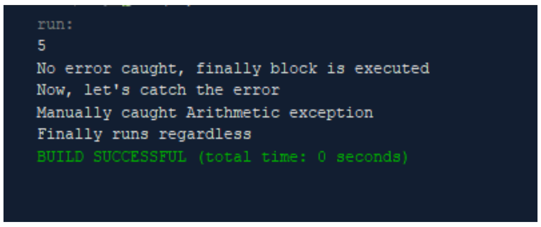
ISHU KUMAR 2006270

Q-3 Write a Java program to illustrate try..catch..finally block.

CODE:

```
class q3 {
    public static void main(String args[]) {
        try {
            int data = 30 / 5;
            System.out.println(data);
        } catch (Exception e) {
            System.out.println(e);
        }

        finally {
            System.out.println("No error caught, finally block is executed");
        }

        System.out.println("Now, let's catch the error");

        try{
            int data = 30 / 0;
            System.out.println("This results in error" + data);
        } catch (ArithmeticException f) {
            System.out.println("Manually caught Arithmetic exception");
        }

        finally {
            System.out.println("Finally runs regardless");
        }

    }
}
```
ISHU KUMAR 2006270

OUTPUT:



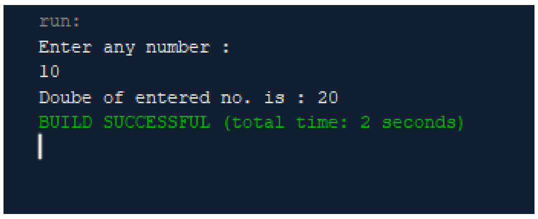
```
run:
5
No error caught, finally block is executed
Now, let's catch the error
Manually caught Arithmetic exception
Finally runs regardless
BUILD SUCCESSFUL (total time: 0 seconds)
```

Q-4 Write a Java class which has a method called ProcessInput(). This method checks the number entered by the user. If the entered number is negative then throw
an user defined exception called NegativeNumberException, otherwise it displays the
double value of the entered number.

Code :-

```
package lab.pkg9;

import java.util.*;

class NegativeNumberException extends Exception {

    @Override
    public String toString() {
        return "Entered number cannot be Negative";
    }
}

public class Q4 {

    static void ProcessInput() throws NegativeNumberException {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter any number : ");
        int a = sc.nextInt();
        if (a < 0) {
            throw new NegativeNumberException();
        } else {
            System.out.println("Doube of entered no. is : " + 2 * a);
        }
    }

    public static void main(String[] args) {
        try {
            ProcessInput();
        } catch (NegativeNumberException e) {
            System.out.println(e);
        }
    }

}
```

OUTPUT:



```
run:
Enter any number :
10
Doube of entered no. is : 20
BUILD SUCCESSFUL (total time: 2 seconds)
```

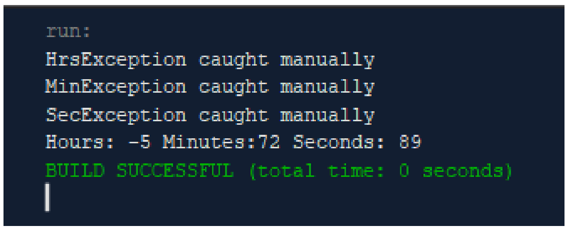
ISHU KUMAR 2006270

Q-5 Write a program to create user defined exceptions called HrsException, MinException and SecException. Create a class Time which contains data members hours, minutes, seconds and throw the user defined exceptions if hours (>24 & <0),minutes(>60 & <0),seconds(>60 & <0).

CODE:

```
package lab.pkg9;
class HrsException extends Exception {
}

class MinException extends Exception {
}

class SecException extends Exception {
}

class time {
    int hrs, min, sec;
    time(int a, int b, int c) {
        hrs = a;
        min = b;
        sec = c;
    }

    public void check() {
        try {
            if (hrs < 0 || hrs > 24) {
                throw new HrsException();
            }
        } catch (HrsException h) {
            System.out.println("HrsException caught manually");
        }
        try {
            if (min < 0 || min > 60) {
                throw new MinException();
            }
        } catch (MinException m) {
            System.out.println("MinException caught manually");
        }
        try {
            if (sec < 0 || sec > 60) {
                throw new SecException();
            }
        } catch (SecException s) {
            System.out.println("SecException caught manually");
        }
    }

    public void show() {
        System.out.println("Hours: " + hrs + " Minutes:" + min + " Seconds: " + sec);
    }
}


public class Q5 {

    public static void main(String args[])
    {
        time t = new time(-5, 72, 89);
        t.check();
        t.show();
    }
}
```

OUTPUT:



```
run:
HrsException caught manually
MinException caught manually
SecException caught manually
Hours: -5 Minutes:72 Seconds: 89
BUILD SUCCESSFUL (total time: 0 seconds)
```

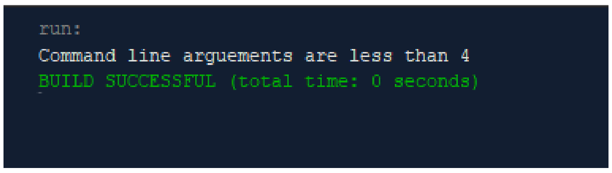
ISHU KUMAR 2006270

Q-6) Create an user defined exception named Check Argument to check the number of arguments passed through command line. If the number of arguments is less than four, throw the Check Argument exception, else print the addition of squares of all the four elements.

CODE:

```
class MyError extends Exception {
}

class q6 {
   public static void main(String args[]) {
      try {
         if (args.length < 4) {
            throw new MyError();
         } else {
            int sum = 0;
            System.out.println("Normal execution");
            int[] arr = new int[args.length];
            for (int i = 0; i < args.length; i++) {
               arr[i] = Integer.parseInt(args[i]);
               System.out.println("arr[" + i + "] : " + arr[i] * arr[i]);
               sum = sum + arr[i] * arr[i];
            }
            System.out.println("Sum of square of command line arguements : " + sum);
         }
      } catch (MyError e) {
         System.out.println("Command line arguements are less than 4");
      } }}
```

OUTPUT:



```
run:
Command line arguements are less than 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

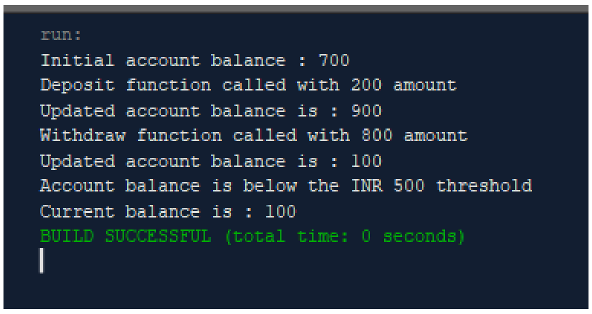
Q-7 Write a java program to create Account with 500 rupee minimum balance, deposit amount, withdraw amount and also throws LessBalanceException which returns the statement that says withdraw amount is not valid.

CODE:

```
package lab.pkg9;
class AccountError extends Exception {
}

class account {
   int balance;

   account(int x) {
      balance = x;
      System.out.println("Initial account balance : " + balance);
   }

   public void check() {
      try{
         if(balance < 500) {
            throw new AccountError();
         }
      } catch (AccountError e) {
         System.out.println("Account balance is below the INR 500 threshold");
         System.out.println("Current balance is : " + balance);
      }
   }

   public void withdraw(int w) {
      System.out.println("Withdraw function called with " + w + " amount");
      balance = balance - w;
      System.out.println("Updated account balance is : " + balance);
      check();
   }

   public void deposit(int d) {
      System.out.println("Deposit function called with " + d + " amount");
      balance = balance + d;
      System.out.println("Updated account balance is : " + balance);
      check();
   }

}
```

```
class Q7 {
    public static void main(String args[]) {
        account a = new account(700);
        a.deposit(200);
        a.withdraw(800);
    }
}
```

OUTPUT:



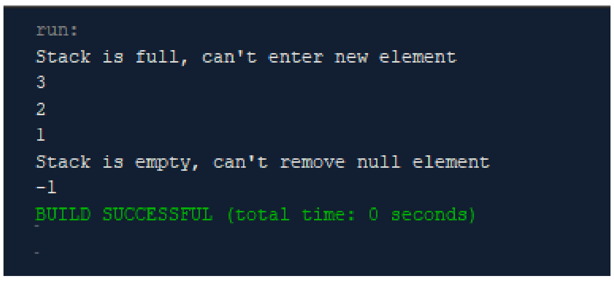
```
run:
Initial account balance : 700
Deposit function called with 200 amount
Updated account balance is : 900
Withdraw function called with 800 amount
Updated account balance is : 100
Account balance is below the INR 500 threshold
Current balance is : 100
BUILD SUCCESSFUL (total time: 0 seconds)
```

Q-8Write a java program to implement a stack class having methods push () and pop().
These methods must be designed to throw user defined exception when the stack is
empty or full.

CODE:

```
package lab.pkg9;
class EmptyError extends Exception {
}

class FullError extends Exception {
}


class stack {
    int n;
    int i = 0;
    int[] arr;
    stack(int n) {
        this.n = n;
        arr = new int[n];
    }

    public void push(int x) {
        try{
            if(i >= n){
                throw new FullError();
            } else {
                arr[i] = x;
                i++;
            }
        } catch (FullError f) {
            System.out.println("Stack is full, can't enter new element");
        }
    }

    public int pop() {
        try{
            if(i > 0){
                int y = arr[i-1];
                i--;
                return y;
            }
            else{
                throw new EmptyError();
            }
        } catch (EmptyError e) {
            System.out.println("Stack is empty, can't remove null element");
            return -1;
        }
    }

}


class Q8 {
    public static void main(String args[]) {
        stack s = new stack(3);
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);
        System.out.println(s.pop());
        System.out.println(s.pop());
        System.out.println(s.pop());
        System.out.println(s.pop());
    }
}
```

OUTPUT:



```
run:
Stack is full, can't enter new element
3
2
1
Stack is empty, can't remove null element
-1
BUILD SUCCESSFUL (total time: 0 seconds)
```