**IT1013: Programming in Java**

**Program manual**

**UNITIV**

**Practice Programs to illustrate the concepts**

**Program1: Nested Try**

package exceptionHandling;

class NestedTry {

public static void main(String args[]) {

try {

int a = args.length;

/\* If no command-line args are present,

the following statement will generate

a divide-by-zero exception. \*/

int b = 42 / a;

System.out.println("a = " + a);

try { // nested try block

/\* If one command-line arg is used,

then a divide-by-zero exception

will be generated by the following code. \*/

if(a==1) a = a/(a-a); // division by zero

/\* If two command-line args are used,

then generate an out-of-bounds exception. \*/

if(a==2) {

int c[] = {1};

c[42] = 99; // generate an out-of-bounds exception

}

} catch(ArrayIndexOutOfBoundsException e) {

System.out.println("Array index out-of-bounds: " + e);

}

} catch(ArithmeticException e) {

System.out.println("Divide by 0: " + e);

} } }

**Program2: ThrowDemo**

package exceptionHandling;

class ThrowDemo {

static void demoproc() {

try {

throw new NullPointerException("demo");

} catch(NullPointerException e) {

System.out.println("Caught inside demoproc.");

throw e; // rethrow the exception

}

}

public static void main(String args[]) {

try {

demoproc();

} catch(NullPointerException e) {

System.out.println("Recaught: " + e);

}}}

**Program3: ThrowsDemo**

package exceptionHandling;

class ThrowsDemo {

static void throwOne() throws IllegalAccessException {

System.out.println("Inside throwOne.");

throw new IllegalAccessException();

}

public static void main(String args[]) {

try {

throwOne();

} catch (IllegalAccessException e) {

System.out.println("Caught " + e);

} }}

**Program4: Finally\_Demo**

package exceptionHandling;

public class FinallyDemo {

// Through an exception out of the method.

static void procA() {

try {

System.out.println("inside procA");

throw new RuntimeException("demo");

} finally {

System.out.println("procA's finally");

}

}

// Return from within a try block.

static void procB() {

try {

System.out.println("inside procB");

return;

} finally {

System.out.println("procB's finally");

}

}

// Execute a try block normally.

static void procC() {

try {

System.out.println("inside procC");

} finally {

System.out.println("procC's finally");

}}

public static void main(String args[]) {

try {

procA();

}

catch (Exception e) {

System.out.println("Exception caught");

}

procB();

procC();

} }

**Program5: Custom Exception**

package exceptionHandling;

class MyException extends Exception {

private int detail;

MyException(int a) {

detail = a;

}

public String toString() {

return "MyException[" + detail + "]";

}

}

class ExceptionDemo {

static void compute(int a) throws MyException {

System.out.println("Called compute(" + a + ")");

if(a > 10)

throw new MyException(a);

System.out.println("Normal exit");

}

public static void main(String args[]) {

try {

compute(1);

compute(20);

} catch (MyException e) {

System.out.println("Caught " + e);

} } }