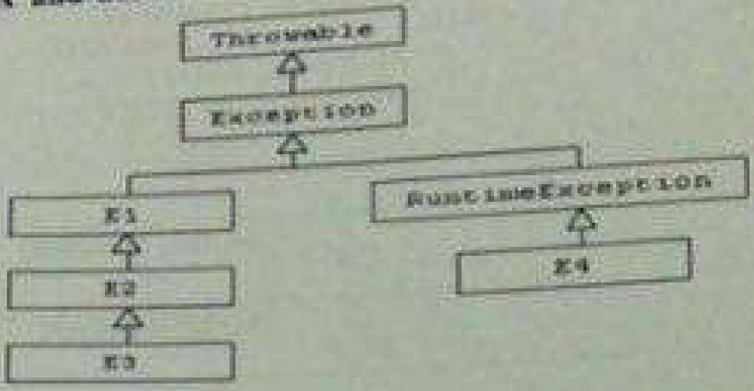
```
catch (MyException e)
{
   System.out.println("catch2:"+e.getMessage());
}
catch(InputMismatchException e)
{
   System.out.println("catch3: invalid entry");
}
catch(Exception e)
{
   System.out.println("catch4:"+e.getMessage());
}
finally {System.out.println("Finally main");}

}
}
```

Assume the user's input for the three declared variables a , b and c will be as shown in the following table, show the program output for each user input such that each run will output no more than 5 lines: **(10 marks)**

a	b	c	output
-3	2	4	1. method1 catch: x is less than 0
			2. finally method1
			3. Finally main
			4.
			5.
30	20	75	1. Finally method2
			2. method1 catch: x is greater than 50
			3. Finally method1
			4. Finally main
			5.
3	0	200	1. finally method1
			2. catch1 Arithmetic Exception
			3. Finally main
			4.
			5.
200	8	one	1. catch method2
			2. Finally method2
			3. finally method1
			4. cath 2: NumberFormatException
			5. Finally main

4.75/5

Question 1: [5 Marks] Given the following class hierarchy, and the class descriptions ClassX and ClassY:



ClassX

public void ***m1()*** → can throw exceptions of type **E1**, **E2**, **E3** or **E4**
public void ***m2()*** → does not throw any exception.

ClassY

public void ***Method1(ClassX x)*** → calls the methods ***m1()*** of the object passed as a parameter ***x***. In case the call would cause an exception, the following processes are performed:

- If the exception is of type **E2** or **E3**, the message contained in the exception object is displayed on the console.
- If the exception is of type **E1** or **E4**, it is transmitted to the calling block.
- In all cases (exceptions are thrown or not), ***m2()*** should be called before the end of the method ***Method1(ClassX x)***.

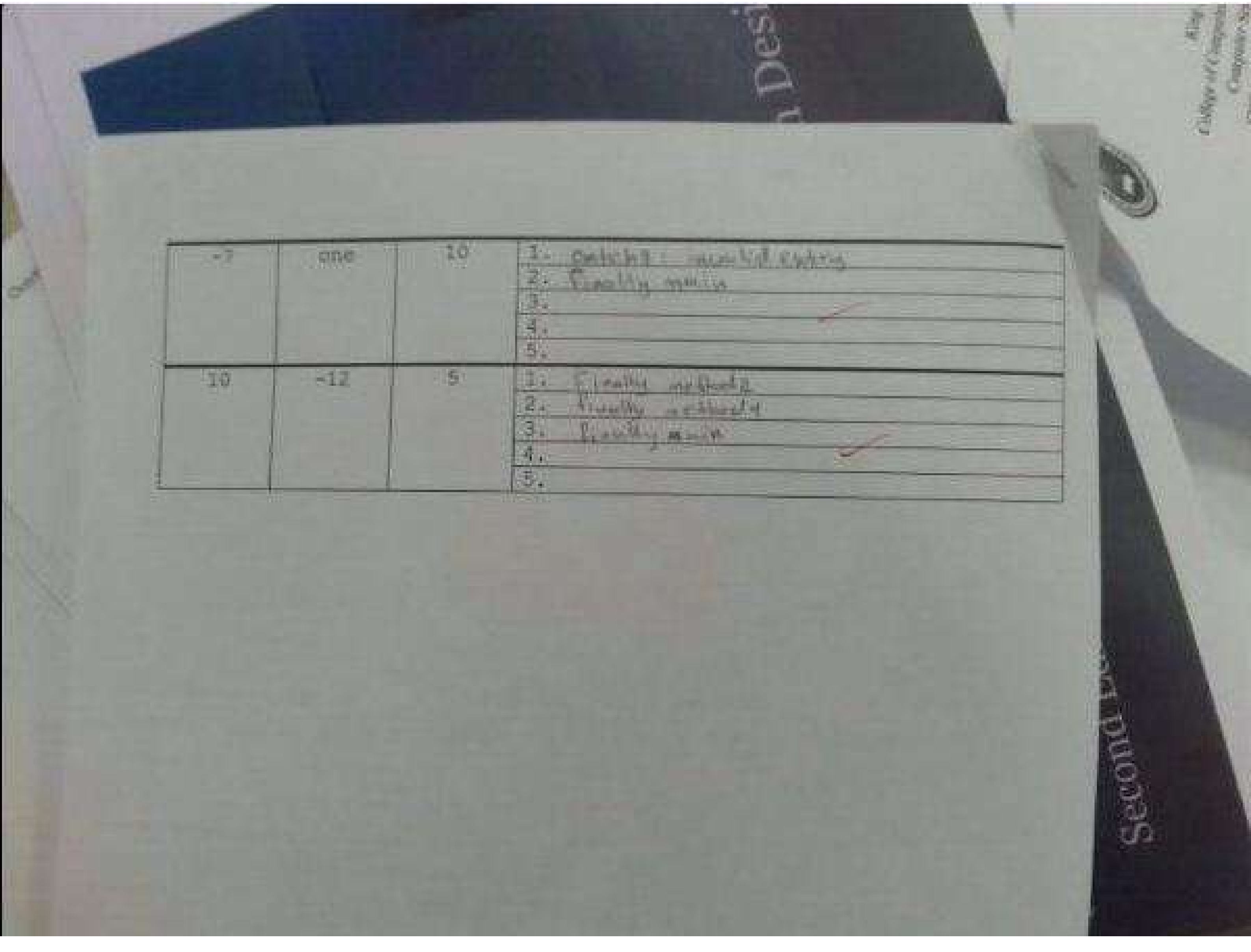
Complete the method *Method1(ClassX x)* of ClassY.

-1/4

```
public class ClassY{
public void method1(ClassX x) throws E1, E4 ✓{

try{
    x.m1(); ✓
}

catch( E3          e3 ✓ ){ ✓
System.out.println( e3.getMessage() );
}

catch( E2          e2 ✓ ){
System.out.println( e2.getMessage() ); ✓
}
catch( E1          e1 ✓ ){
    throw e1; ✓
}

catch( E4     e4 ✓ ){
    throw e4; ✓
}

finally{
    x.m2() ✓
}}}
```

Question 1: **Given the java code below:**

```
import java.util.*;

class MyException extends Exception
{

   public MyException(String message)
   {
      super(message);
   }
}//end class MyException

public class Test
{                                    10    -2         5
   public static void method1(int x, int y, String s) throws Exception
   {
      try
      {
         if(x<0)
            throw new IllegalArgumentException("x is less than 0");
         double result=x/y;  1/2
         method2(s);  75
      }
```



-7	one	10	1. catching: [illegible] 2. finally main 3. ✓ 4. 5.
10	-12	5	1. finally method2 2. finally method1 3. finally main ✓ 4. 5.

Question 1: [5 Marks]

1. Trace the following program assuming the input is: -6 5a 56

```
import java.util.*;
import java.io.*;
public class excp {
static Scanner in=new Scanner(System.in);
public static void main(String[] args){
try{
   System.out.println(m2());
   m1();}
catch(Exception e){
  if (e instanceof IOException)
      System.out.println("excpmain");
  else System.out.println("unknown
exception");}
finally { System.out.println("finally");  }
System.out.println("the end"); }

public static int m1() throws  IOException
{
  try  { throw new IOException();}
  catch ( IOException imeRef){
          System.out.println("m1  ");
          throw imeRef; }
}
public static int m2() {
 boolean t=true;  int x=0;
 while(t){
      try{
         x = in.nextInt();
         if (x<0) throw new Exception("negative");
         System.out.println("Input OK");
         t = !t;  }
catch(InputMismatchException e){
System.out.println("Please enter integers only");
 in.next();
}
catch(Exception e){
  System.out.println(e.getMessage());}  }
return x;}}
```

2.5

Output

Input = -6

negative
-6
m1
excpmain
finally
the end

Input = 5a

Please enter integers only
0
m1
excpmain
finally
the end

Input = 56

Input ok ✓
56 ✓
m1 ✓
excpmain ✓
finally ✓
the end ✓

56 < 0 → F

```
    catch( IllegalArgumentException e)
    {
       System.out.println("method1 catch: "+e.getMessage());
    }
    finally
    {System.out.println("finally method1"); }

}//end method1
public static void method2( String s) throws MyException
                              24
{
    try
    {       25
       int x= Integer.parseInt(s);

       if(x>50)
         throw new IllegalArgumentException("x is greater than 50");
    }
   catch(NumberFormatException e)
    {
       System.out.println("catch methd2");
       throw new MyException("NumberFormatException") ;
           }
    finally
    { System.out.println("Finally method2");}
}

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

    try
    {
       int a=scan.nextInt();
       int b=scan.nextInt();
       String c=scan.next();

       method1(a,b,c);
    }//end try
    catch(ArithmeticException e)
    {
    System.out.println("catch1:Arithmetic Exception");
    }
```

Q1:Trace the following Java program (4.5)

```
public class testExceptions{
static int number=3;       3
public static int  m1(int a){
try{
if(a%2==0) throw new ExceptionA  ("even !");
if(a++>0) throw new ExceptionA("m1 throws an ExceptionA");
number++;
}catch(ExceptionB  e)
{System.out.println(e.getMessage()+" ** catch m1");}
finally
{ System.out.println("finally m1");
}
return 50;}

public static String m2(){

try{          3
number = m1(number);
if(number >0) throw new ExceptionB("more than zero");
return "try";
}
catch(ExceptionB  exp)
{ System.out.println(exp.getMessage()+" ** catch1 m2");
return "catch1";}

catch(ExceptionA  exp)
{ System.out.println(exp.getMessage()+" ** catch2 m2");
return "catch2";}

finally{ System.out.println("finally m2 "+ number); }
}
```

m1(n)
m2
m3
main

```
// method that will test if year entered is 4 digit number
public void test_year(int age) throws E1
{
If( x < 1000)
throw new E1("num should be >1000");
}
```
1

```
// method that will read year from user then call method test_year to check if it is 4 digit number
public int get_year() throws E1        0.5
{
Scanner read=new Scanner (System.in);
int x = read.nextInt(); ※
test_year(x);
}
```

```
public static void main(String [] args) {
try {
int age = get_year ();
}
                                        0.5
Catch ( InputMismatchException E3 )    ← from get_year (int x = read.nextInt()) ※
{ System.out.println("enter a correct value");}

Catch ( E1 e1      0.5 ){
   age=age+1000;
  try{
     if (age < 1900)
                                        0.5
     throw new E2("num should be >=1900")
  }
Catch ( E2 e2      0.5 )
{ System.out.println(e2.getMessage()); 0.25
                                        0.75
} finally{ System.out.println("Year is :"+age );}

}
```

```
Class E1 extends Exception{
public E1(String description){
super(description);
}}
```
1

```
Class E2 extends Exception{
public E2(String description){
super(description);
}}
```

```
public static void m3(){

try{
int num=Integer.parseInt(m2());     catch1
}catch (NumberFormatException exp)
{System.out.println("non-numeric String  ");}
}
```

```
public static void main(String args[]){
try{m3();}
catch(Exception e){ System.out.println("last catch");}}}

class ExceptionA  extends RuntimeException  {
   public ExceptionA (String m){
       super(m);} }
 class ExceptionB  extends ExceptionA {
  public ExceptionB (String m){
       super(m);} }
```

Output:

finally m1 ①

m1 throws an ExceptionA ** catch2m2 ①

finally m2 3 ①

non-numeric string ①

Q2: Complete code (5.5)

Complete the following program that will read from user a 4 digit integer number as his/her year of birth. The program include 2 user defined exception classes

E1: is a checked exception, thrown if year entered is less than 4 digits Y < 1000

E2: is a checked exception, thrown if year entered is less than 1900 Y < 1900

These user defined exceptions are to be handled in main in addition to any other type of exception

- **If E1 is caught** : fix the year to make it 4 digits then test it if the fixed number in range (**>= 1900**).
- **If E2 is caught** : the exception message is displayed.
- **In any case**, display the year entered.
  finally