1). Write a program containing a possible exception. Use a try block to throw it and a catch block to handle it promptly.

Solution =>

```
import java.util.Scanner;
class InvalidInput extends Exception
{
public InvalidInput (String msg)
{
super(msg);
}
}
class CheckChar
{
void check(char c) throws InvalidInput
{
int code= (int)c;
if (!((code>=65 && code<=90) || (code>=97 && code<=122)))
{
throw new InvalidInput ("Not an Alphabet.");
}
else
{
System.out.println("Input is Valid.");
}
}
}
public class Demo
{
public static void main(String[] args)
{
```

```
Scanner sc = new Scanner(System.in);
CheckChar obj = new CheckChar();
System.out.print("Enter any Character: ");
try
{
obj.check(sc.next().charAt(0));
}
catch(Exception e)
{
System.out.print(e);
}
}
}
```

2). Write a program that illustrates the application of multiple catch statements.

Solution =>

```
public class MultipleCatch
{
public static void main(String[] args)
{
try
{
int a[] = new int[5];
a[5] = 3/0;
}
catch(ArithmeticException e)
{
System.out.println("Arithmetic Exception occurs ");
}
catch(ArrayIndexOutOfBoundsException e)
```

```
{
System.out.println("ArrayIndexOutOfBounds Exception occurs ");
}
catch(Exception e)
{
System.out.println("Parent exception occurs ");
}
System.out.println("Rest of the code ");
}
}
```

3). Write a program that demonstrates how certain exception types are not allowed to be thrown.

Solution =>

```
public class NotThrow
{
public static void checkDivide(int dividend,int divisor)
{
if(divisor == 0)
{
throw new Exception("Not Possible.");
}
}
public static void main(String[] args)
{
int a = 10;
int b = 0;
checkDivide(a,b);
}
}
```

4). Write a program to demonstrate the concept of re-throwing an exception.

Solution =>

```
public class ExceptionHandling
{
public static void main(String[] args)
{
try
{
methodWithThrow();
}
catch(NullPointerException ex)
{
System.out.println("NullPointerException Re-thrown in methodWithThrow() method will be handled here");
}
}

static void methodWithThrow()
{
try
{
String s = null;
System.out.println(s.length());
}
catch(NullPointerException ex)
{
System.out.println("NullPointerException is caught here");
throw ex;
}
```

```
}
}
```

5). You will be given two integers and as input, you have to compute x/y. If x and y are not 32 bit signed integers or if y is zero, exception will occur and you have to report it. Read sample Input/Output to know what to report in case of exceptions.

Sample Input 0:

10

3

Sample Output 0:

3

Sample Input 1:

10

Hello

Sample Output 1:

Java.util.InputMismatchException

Solution =>

```
import java.io.*;
import java.util.*;
public class Solution
{
public static void main(String[] args)
{
try
{
Scanner in = new Scanner(System.in);
int x = in .nextInt();
int y = in .nextInt();
System.out.println(x/y);
}
```

```
catch (ArithmeticException e)
{
System.out.println(e);
}
catch (InputMismatchException e)
{
System.out.println(e);
}
}
}
```