

Lab Assignment

Program 1

```
import java.util.Scanner;

public class Exception1 {

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

        int num = 0;

        do {
            System.out.println("Enter a number between 1 and 10");
            num = scan.nextInt();

            if (num < 1 || num > 10)

                System.out.println("\nIllegal value, " + num + " entered.
Please try again.");
        } while (num < 1 || num > 10);

        System.out.println("\nValue correctly entered! Thank you.");
    }
}
```

Lab Questions:

1. Type* the program above and compile. Run and enter an integer between 1 and 10.

```
> run Exception1
Enter a number between 1 and 10
2

Value correctly entered! Thank you.
> |
```

2. The program is requesting a number between 1 and 10. Run the program again and enter 5.5. Although this number is between 1 and 10, the program will abort. Examine the error message. You should see the word Exception, the method where the exception occurred (main), the class name of the exception (InputMismatchException), as well as the call stack listing the method calls.

```
> run Exception1
Enter a number between 1 and 10
5.5

java.util.InputMismatchException
    at java.util.Scanner.throwFor(Unknown Source)
    at java.util.Scanner.next(Unknown Source)
    at java.util.Scanner.nextInt(Unknown Source)
    at java.util.Scanner.nextInt(Unknown Source)
    at Exception1.main(Exception1.java:12)
```

3. Add a try/catch block to catch and handle the InputMismatchException exception. Identify the statements that cause the error as well as the portions of the program that depend upon these statements. Enclose these statements within the try block. Follow the try block with

the catch block given below. Note, the `InputMismatchException` class is defined in `java.util` and must be imported. Also, when the `Scanner` throws an `InputMismatchException`, the input token will remain in the buffer so that it can be examined by the program. In our case, we will not be examining the token, but will simply clear out of the buffer to start over.

```

1  import java.util.Scanner;
2  import java.util.*;
3  public class Exception1 {
4
5      public static void main(String[] args) {
6          Scanner scan = new Scanner(System.in);
7          int num = 0;
8          try{
9              do {
10                 System.out.println("Enter a number between 1 and 10");
11                 num = scan.nextInt();
12
13                 if (num < 1 || num > 10)
14
15                     System.out.println("\nIllegal value, " + num + " entered. Please try again.");
16                 } while (num < 1 || num > 10);
17
18                 System.out.println("\nValue correctly entered! Thank you.");
19             }
20             catch (InputMismatchException ime) {
21                 System.out.println("Enter whole numbers only, with no spaces or other characters");
22                 scan.next();          // clear the scanner buffer
23             }
24         }
25     }

```

4. Compile and run the program again, testing with a variety of input (integers, floats, characters) The program should not abort when floats or character data is given.

- integer

```

> run Exception1
Enter a number between 1 and 10
1
Value correctly entered! Thank you.

```

- floats

```

> run Exception1
Enter a number between 1 and 10
2.3
Enter whole numbers only, with no spaces or other characters

```

- characters

```

> run Exception1
Enter a number between 1 and 10
c
Enter whole numbers only, with no spaces or other characters

```

5. Complete the security checklist for Program 1.

```

1  import java.util.Scanner;
2  import java.util.*;
3  public class Exception1 {
4      public static void main(String[] args) {
5          Scanner scan = new Scanner(System.in);
6          int num = 0;
7          try{
8              do {
9                  System.out.println("Enter a number between 1 and 10");
10                 num = scan.nextInt();
11                 if (num < 1 || num > 10)
12                     System.out.println("\nIllegal value, " + num + " entered. Please try again.");
13             } while (num < 1 || num > 10);
14             System.out.println("\nValue correctly entered! Thank you.");
15         }
16         catch (InputMismatchException ime) {
17             System.out.println("Enter whole numbers only, with no spaces or other characters");
18             scan.next();          // clear the scanner buffer
19         }
20     }
21 }

```

Security Checklist	
Vulnerability: Exception	
Check each line of code	Line number
1. line number of statement that may cause an exception to occur.	10
2. line number of any statement in the exception handler that may expose information about the program.	16-19
3. line number of any statement in the exception handler that may expose information about the file system.	-
Highlighted areas indicate vulnerabilities!	

Program 2

```
import java.util.Scanner;
import java.io.File;

public class Exception2 {

    public static void main(String[] args) {

        int total = 0;
        int num = 0;

        File myFile = null;
        Scanner inputFile = null;

        myFile = new File("inFile.txt");
        inputFile = new Scanner(myFile);

        while (inputFile.hasNext()) {
            num = inputFile.nextInt();
            total += num;
        }

        System.out.println("The total value is " + total);
    }
}
```

Lab Questions:

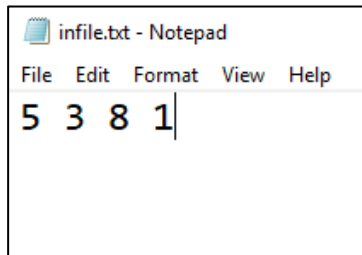
1. As we saw in Lab Program 1, the Scanner may throw an *InputMismatchException*. This is an unchecked exception and we are not required to handle unchecked exceptions. The Java runtime system will handle the exception by aborting the program and displaying the exception information. Other methods may throw checked exceptions. A checked exception must be handled. Type* the program above and compile. You should receive a compilation error stating that the *FileNotFoundException* must be caught or declared to be thrown.
2. We will first test the program knowing that the input file does not exist. Add a try/catch block such that if the input file is not found, report the problem and display a total of 0. Use the `getMessage()` method of the exception object to report the problem.

```
catch (FileNotFoundException fnf)
{
    //display total of 0
    System.out.println(fnf.getMessage())
}
```

```
1 import java.util.Scanner;
2 import java.io.File;
3 import java.io.FileNotFoundException;
4 public class Exception2 {
5     public static void main(String[] args) {
6         int total = 0;
7         int num = 0;
8         File myFile = null;
9         Scanner inputFile = null;
10        try{
11            myFile = new File("inFile.txt");
12            inputFile = new Scanner(myFile);
13            while (inputFile.hasNext()) {
14                num = inputFile.nextInt();
15                total += num;
16            }
17            System.out.println("The total value is " + total);
18        }
19        catch (FileNotFoundException fnf){
20            System.out.println("The total value is " + total);
21            System.out.println(fnf.getMessage());
22        }
23    }
24 }
```

```
> run Exception2
The total value is 0
inFile.txt (The system cannot find the file specified)
> |
```

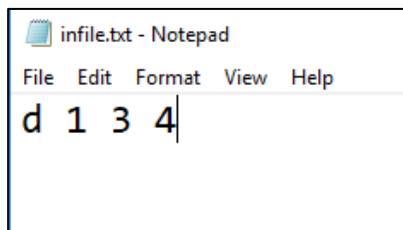
3. Create a file named inFile.txt using a plain text editor. Enter only integer values into the file and save the file in the same folder as the program (if you are using a IDE, then put it in the main project folder). Run the program to verify that the program functions correctly with a valid input file.



```
inFile.txt - Notepad
File Edit Format View Help
5 3 8 1
```

```
> run Exception2
The total value is 17
>
```

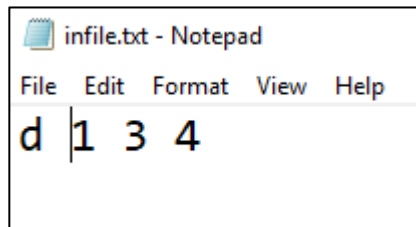
4. Any number of catch clauses may be coded for a single try block. Add another catch clause to catch *InputMismatchException* immediately before or after the *FileNotFoundException* clause. Remember to import the exception class and to clear the scanner buffer. Modify the inFile.txt file to include an error. Run and test the program. As soon as an exception occurs, the catch clause is executed. Since the *InputMismatchException* catch clause is outside of the read loop, the first invalid piece of data will cause the program to end without reading the remainder of the file.



```
inFile.txt - Notepad
File Edit Format View Help
d 1 3 4
```

```
> run Exception2
java.util.InputMismatchException
    at java.util.Scanner.throwFor(Unknown Source)
    at java.util.Scanner.next(Unknown Source)
    at java.util.Scanner.nextInt(Unknown Source)
    at java.util.Scanner.nextInt(Unknown Source)
    at Exception2.main(Exception2.java:15)
```

5. Any number of try blocks may be coded with each having a least one catch clause. Rather than catching the *InputMismatchException* outside the loop, let's move it inside the loop. This will allow our program to report the invalid data, resume processing, and produce correct results for the valid data given. Add a new try block within the while loop and move the *InputMismatchException* catch block such that the while loop is as shown below.



```
inFile.txt - Notepad
File Edit Format View Help
d 1 3 4
```

```
> run Exception2
Illegal value found
The total value is 8
> |
```

6. Complete the security checklist for Program 2.

```
1 import java.util.Scanner;
2 import java.io.File;
3 import java.util.InputMismatchException;
4 import java.io.FileNotFoundException;
5 public class Exception2 {
6     public static void main(String[] args) {
7         int total = 0;
8         int num = 0;
9         File myFile = null;
10        Scanner inputFile = null;
11        try{
12            myFile = new File("inFile.txt");
13            inputFile = new Scanner(myFile);
14            while (inputFile.hasNext()) {
15                try {
16                    num = inputFile.nextInt();
17                    total += num;
18                }
19                catch (InputMismatchException ime) {
20                    System.out.print("Illegal value found\n");
21                    inputFile.next();
22                }
23            }
24        }
25    }
26 }
```

```

22     }
23 }
24     System.out.println("The total value is " + total);
25 }
26 catch (FileNotFoundException fnf){
27     System.out.println("The total value is " + total);
28     System.out.println(fnf.getMessage());
29 }
30 }
31 }

```

Security Checklist	
Vulnerability: <i>Exception</i>	
Check each line of code	Line number
1. line number of statement that may cause an exception to occur.	12,13,16
2. line number of any statement in the exception handler that may expose information about the program.	19-22
3. line number of any statement in the exception handler that may expose information about the file system.	26-29
Highlighted areas indicate vulnerabilities!	

Discussion Questions

1. With keyboard input, what types of errors may occur?

Answer InputMismatchException

2. An *InputMismatchException* is an unchecked exception. Since exception handling of an unchecked exception is not required, why should the programmer add this checking?

Answer ควรใส่ Try/Catch เพื่อจัดการกับ Exception เพื่อป้องกันไม่ให้เกิด Error ที่ตัว Input ที่รับเข้ามา

3. What types of exceptions may occur when working with a file? Hint, visit the Java API and select the java.io library. Review the list of exceptions included in this library.

Answer FileNotFoundException

4. Describe other types of exceptions (non-IO related) that may occur at runtime. Hint: view the Exceptions category of various libraries such as java.lang and java.util in the Java API.

Answer ArithmeticException, NumberFormatException, InputMismatchException

5. If a method throws a checked exception, how can the calling method avoid coding a try/catch block to handle the exception? In what cases might the calling method use this option?

Answer throws จะเรียกใช้ก็ต่อเมื่อ method นั้นยังไม่ต้องจัดการกับ Exception แต่จะให้ method อื่นที่เรียก method นี้เป็นตัวจัดการแทน

6. Exception classes are derived from other exception classes. For example, the *FileNotFoundException* class is derived from *IOException* class. And, the *IOException* class is derived from the *Exception* class. Rather than check for the exact exception class, the catch clause could specify a super class. What are the advantages and disadvantages of specifying a super class in a catch clause?

Answer