# Exception 2

1. **Check if file exists by using the exists() method in java.io.File**

   The `java.io.File` class contains useful methods for dealing with files. The code fragment below can be used to check if a particular file (in this case `c:\students.txt`) exists on the hard disk.

```
String fileNameAndPath = "c:/students.txt";
File f = new File(fileNameAndPath);
if (f.exists()) {
  System.out.println("Yes, " + fileNameAndPath + " exists");
} else {
  System.out.println("No, " + fileNameAndPath + " does not exist");
}
```

   Write a class called `TextFileReader.java`, with a `main` method. Using the code fragment above, `main` should check whether `c:\students.txt` exists, and print out the corresponding message.

   (a) Make sure `TextFileReader` compiles and runs.

   (b) Create a dummy text file (using Notepad) called `students.txt` containing a few lines of names, and store it in `c:\`. Then run `TextFileReader` again to ensure that it works correctly.

   "`c:\students.txt`" is what we call an absolute path. An absolute path specifies exactly where on the specific drive and folder a particular file is found. In most circumstances it's better to use a relative path – relative to where the java command (java.exe) is executed.

   (c) Modify `TextFileReader` so that it uses a relative path:

```
String fileNameAndPath = "data/students.txt";
```

   In this case, `TextFileReader` will try to search for a `data` folder relative to where you invoke the java command, and search for `students.txt` in it. Check to see that your relative path name works by moving your `students.txt` to the appropriate folder and observing the output of your program.

2. **Check if file exists by catching exception**

   When using the `Scanner` class that takes in a `File` parameter, a `FileNotFoundException` is thrown when the file does not exists.

```
String fileNameAndPath = "data/students.txt";
try {
    Scanner sc = new Scanner(new File(fileNameAndPath));
    System.out.println("Yes, " + fileNameAndPath + " exists");
} catch (FileNotFoundException e) {
    System.out.println("No, " + fileNameAndPath + " does not exist");
}
```

To measure how fast a piece of code runs, we make use of the `System.currentTimeMillis()` method. This method returns us the number of milliseconds that has lapsed since 1 Jan 1970.

```
long start = System.currentTimeMillis();

// TODO
// place the method(s) that you wish to measure the execution time here

long end = System.currentTimeMillis();
// prints the number of milliseconds that has elapsed
System.out.println(end – start);
```

Measure the time taken to check if a file exists via exception (Q2 sample code) versus using the File's exists method (Q1 sample code).

3. The `Adder` class (given) is a program that prompts the user for two numbers and prints the sum.

```
Enter num 1> 1

Enter num 2> 2

The answer is 3
```

The program crashes when an invalid input(e.g. "abc") is entered.

```
Enter num 1> abc
Exception in thread "main" java.util.InputMismatchException
       at java.util.Scanner.throwFor(Scanner.java:909)
       at java.util.Scanner.next(Scanner.java:1530)
       at java.util.Scanner.nextInt(Scanner.java:2160)
       at java.util.Scanner.nextInt(Scanner.java:2119)
       at Adder.main(Adder.java:8)
```

Include exception handling into the program to make the program robust, i.e. inform the user of invalid inputs and prompt the user to try again.

```
Enter num 1> a
Please enter a number!

Enter num 1> a
Please enter a number!

Enter num 1> 1

Enter num 2> b
Please enter a number!

Enter num 2> 2

The answer is 3
```

- END -