Exception 2

Resource: *http://blue.smu.edu.sg/exception2-resource.zip*

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
```

4. Compile the `PageDownloader` class (given). Note the compile errors. Then include in exception handling codes so that the program will keep prompting the user for a valid URL and prints the content of web page of the given URL.

```
Enter the URL> http://
Invalid URL!

Enter the URL> abc
Invalid URL!

Enter the URL> http://violet.smu.edu.sg
<!DOCTYPE html>
<html>
<body style="background-color:#A020F0">


</body>
</html>
```

5. You are given `BabyFeedMain.java` below:

```
public class BabyFeedMain {
  public static void main (String[] args) {

    System.out.println("Start feeding");
    try {
      Baby b = new Baby();
      b.setIsHungry(true);
      b.eat("baby food");

      b.setIsHungry(false);
      b.eat("hamburger");
    } catch (NotHungryException e) {
      System.out.println("There's a problem! " + e.getMessage());
    }
    System.out.println("End feeding");
  }
}
```

(a) Write a class called `NotHungryException.java` that extends `java.lang.Exception`. Provide a constructor that takes in the exception message. Make sure it compiles.

(b) Write a class called `Baby.java`. Baby has got one private attribute called `isHungry` (type `boolean`), and two other methods:
   a. a public setter for `isHungry`
   b. a public method called `eat` that takes in a `String` (the food description), and returns nothing. When `eat` is called, it will print "**eating <food description>**" if the baby is hungry. Otherwise, this method throws a `NotHungryException` to the caller with the exception message "**rejects <food description>**".

(c) Ensure that `BabyFeedMain` works with your classes.

6. A company accepts user orders by part numbers interactively. Users might make the following errors when entering data:
   - The part number is too low (less than 0)
   - The part number is too high (more than 999)
   - The quantity ordered is too low (less than 1)
   - The quantity ordered is too high (more than 5000)

(a) Write a class called `PartAndQuantityEntry.java` with a method called `enterPartAndQuantity` that takes in 2 `int`s (the part number and the quantity ordered). This method may either print a message "successfully inserted order", or throw back a `java.lang.IllegalArgumentException` object (with an appropriate exception message) to the caller.

(b) Write a class called `TestPartAndQuantityEntry.java` with a `main` method. The `main` method prompts the user for the part and quantity, and calls the `enterPartAndQuantity` method with these 2 numbers. If an exception occurs, `main` catches the exception and prints the error message embedded in the exception object caught.

(c) Modify `main` so that if an exception occurs, it repeatedly prompts for the part and quantity until the order is successfully ordered. Note there is no need to actually create an order (Order object or anything like that).

A sample screen is shown below:

```
Enter part number> 100

Enter quantity> 5005
Invalid order, part number should be between 0 and 999, quantity ordered between 1 and 5000

Enter part number> 100

Enter quantity> 2005
Successfully inserted order
```

7. Given in the resource is the DVD Rental Application source files. `PatronDAO` & `DVDDAO` have been converted to read from a file (instead of hardcoding the values in the constructor).
   a. Create your own user-defined exception class called DataException. DataException is an unchecked exception.
   b. Modify PatronDAO & DVDDAO so that the method throws a DataException whenever there is an exception generated by the file reading/writing code.
   c. The RentalMenu should catch this exception and print "Internal Error." and quit the program.
   d. Modify `RentalDAO` so that rental information is saved to a text file called rental.csv in the data directory. This information should be made available when the program runs the next time.
      i. You may decide on the data format of `rental.csv`.
      ii. RentalDAO should make use of `DVDDAO` and `PatronDAO` where required.
   e. Complete `compile.bat` & `run.bat`
      - Compile.bat must compile all the Java files in the `src` directory and place the class files in the `classes` folder
      - `run.bat` must run `RentalApplication` from the `classes` folder.

8. Compile the DatabaseDemo class (given). Note the compile errors. Then include in exception handling code and run the code. The code given sets the parameter for username and returns the full name of the user.

   Add logic so that the program will keep prompting the user for a valid username and password, and prints either "Welcome, <fullname>!" or "Access Denied" until the user enters "quit" for username and password.

   Write `compile.bat` & `run.bat`
   - compile.bat will compile all the Java files in the `source` directory and place the class files in the `output` folder
   - `run.bat` will run `RentalApplication` from the `classes` folder.

   Note:
   1. Do not modify the directory structure.
   2. The class files should be placed in the output folder.
   3. The code requires the use of the jar file in the lib folder.

```
Enter username >apple
Enter password >apple123
Welcome Apple Ang

Enter username >orange
Enter password >orange1234
Access Denied

Enter username >quit
Enter password >quit
Bye bye!
```

- END -