CS 212 - Lab 3

Exception Handling

GOALS:

Explore the use of System.in and validating input
Explore exception handling
Create and use a programmer-defined exception class

For additional information on exceptions:
http://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

Exercise:

Create a new project named **lab3a**

Within this project create a new class named **Date**. The code for this class can be found at http://venus.cs.qc.cuny.edu/~aabreu/cs212/lab3/Date.java

Within this project create a new class named **Lab3App**

1. Modify the comments at the beginning of both classes to include a title, a description, and your names as the authors.

2. Include the import statement necessary to make the Scanner class available. `import java.util.Scanner;`

3. Within the Lab3aApp class include the following code:

```
public static void main(String[] args) {
      // Create a Scanner object and connect it to
      System.in Scanner scan = new Scanner(System.in);
}
```

4. Include the code required to:
   - Create a reference variable of type Date called **aDate**
   - Create three integer variables: **month, day, year**

5. The *nextInt* method, defined in the Scanner class, permits the user of the program to input an integer value. For example:

```
System.out.print("Enter the month as an integer:");
month = scan.nextInt();
```

6. Enter the code above **and** the code required to get the day and year from the user.

7. Run your program to ensure that your code works properly. **Note: when you run your program the prompts will appear in the message pane at the bottom of the screen.**

8. Instantiate a new Date object using the parameterized constructor passing the values you entered. The reference to this object should be stored in **aDate**.

9. The nextInt method may/can throw an **InputMismatchException**. An **InputMismatchException** is an "unchecked" exception. It is thrown by the nextInt method when the input does not match an integer or the value is outside the range for that type. This will happen if the user enters non-numeric data for month, day, or year. Add a throws InputMismatchException clause to the header of the main method and run the program.
   Answer question #1. Note: **Remove the throws clause before continuing.**

   Try running your program again and enter "w" for the month.
   Answer question #2.

10. Modify your code by replacing the input statements with the following statements.

```
try {
      System.out.print("Enter the month as an integer: ");
      month = scan.nextInt();
      System.out.print("Enter the day as an integer: ");
      day = scan.nextInt();
      System.out.print("Enter the year as an integer:
      "); year = scan.nextInt();
      aDate = new Date(month, day, year);
} catch (InputMismatchException ime) {
      System.out.println("Invalid input entered. Enter an integer");
}
```

   Add the import statement needed for InputMismatchException:

```
import java.util.InputMismatchException;
```

   Try running your program again and enter "w" for the month. **Answer question #3.**

11. The purpose of including a try…catch is to handle these types of exceptions in a more elegant manner. For example, prompt the user to enter the data again rather than having the program terminate prematurely. Include the following code (the code in bold) to your program.
    Note: *done* should be declared as a *boolean* type and initialized to false.

```
while(!done) {
      try {
            System.out.print("Enter the month as an integer:
            "); month = scan.nextInt();
            System.out.print("Enter the day as an integer: ");
            day = scan.nextInt();
            System.out.print("Enter the year as an integer:
            "); year = scan.nextInt();
            aDate = new Date(month, day, year);
            done = true;
      } catch (InputMismatchException ime) {
            System.out.println("Invalid input entered. Enter an integer");
      }
}
```

Execute your program again and enter "w" for month or day or year. When entering "w" for day you will enter a valid integer for month; when entering "w" for year you will enter valid integers for month and day. You are entering invalid data to test your code; you will have to enter valid data at some point in order for your program to continue executing.

When the scanner throws an InputMismatchException, the scanner will not pass the token that caused the exception, so that it may be retrieved or skipped via some other method. The nextLine method can be used to bypass the erroneous input. **Add the nextLine statement to the catch block, and execute the program again. Answer question #4.**

Note that this is not the most elegant solution for example, if you enter a valid month and a valid day, but enter a non-numeric character for year, you are forced to enter all values again.

12. Execute your program and enter a "13" for month. Since "13" is a valid integer your program does not detect an error. Include the following code (the code in bold) in the setMonth method defined in the Date class.

```
if (month >= 1 && month <= 12)
        dMonth = month;
else
        throw new RuntimeException("Invalid Month: month out of range");
```

13. Execute your program; enter 13, 12, 2016 for month, day, and year respectively.

    Answer question #5.

14. Currently your program does not contain a handler for RuntimeExceptions. Include an appropriate catch block to handle a RuntimeException. Place this catch block **above** the existing catch blocks for InputMismatchException.

    Answer question #6.

15. Execute your program; enter 13, 12, 2016 for month, day, and year respectively.

    Answer question #7.

16. Since many types of exception are considered RuntimeExceptions, it is sometimes desirable to create your own exception class. Create a new class called **DateException** and include the following code:

```
public class DateException extends Exception{
     public DateException() {
          super("Invalid value for Date");
     }
     public DateException(String message) {
          super(message);
     }
}
```

17. Add a comment at the beginning to include a title, a description, and your names as the authors. You should include appropriate JavaDoc for both constructors.

18. Modify the setMonth method in the Date class by including the following code (the code in bold).

```
public void setMonth(int month) throws DateException{
     if (month >= 1 && month <= 12)
         dMonth = month;
     else
         throw new DateException("Invalid Month: month out of range");
 }
```

    Answer questions #8 and 9.

19. Modify the code in the RuntimeException handler so that it appears as it does below.

```
    catch (DateException ex) {
      System.out.println("DateException: " + ex.getMessage());
  }
```

20. Execute your program; enter 13, 12, 2016 for month, day, and year respectively.
    Answer question #10.

    Enter valid data for month, day, and year so that your program will terminate.
    Answer question #11.

21. Include the code required to loop until the user enters a valid month. Keep in mind that the user might enter "13" or "q3" for month, both are invalid.
    Then loop until the user enters a valid day without making them re-enter the month. Then loop until the user enters a valid year.
    If the date is invalid, throw a DateException with an appropriate message and repeat the above steps until the date is valid.
    Display the valid date, using the Date's toString method.

22. Modify the setDay and the setYear methods to throw DateExeption as follows:

    setDay -
    A valid day depends on the month. Additionally, February has an additional day when the year is a leap year. Since the year is not yet known, accept integers from 1 to 29 for that month, and throw a DateException with an appropriate message, otherwise.

    setYear -
    For our purposes, a valid year is one that falls between 1752 and this year, inclusive. Throw a DateException with an appropriate message, if the year is invalid. Add a private method to the Date class that determines if a year parameter represents a leap year. Use the method to determine if day 29 is valid for that year, and throw a DateException with an appropriate message, if it is not.

    Note: 2/29/2016 is a valid date, but 2/29/2017 is not.
    A year is a leap year if it is divisible by 4 but not 100, or if it is divisible by 400.

23. Test the program with several dates (valid and invalid) and write the valid dates to a text file called *validates.txt*.

The following questions and the answers must be typed using MS Word, WordPad, or Notepad. Please include blank lines between each question so that I can provide any necessary feedback.

1. Did the ***throws InputMismatchException*** clause cause any errors? Why, or why not?

2. Describe what happened? Include your interpretation of the output displayed. Where is the exception object created?

3. Describe the output displayed. How is it different than the output displayed previously? Describe what happened this time vs. what happened previously. Please be specific. What was the last statement to execute within the try block?

4. Explain why the statement **done = true;** is placed at the end of the try block. Was there a difference in output when you added the nextline statement to the catch block?

5. Why must you enter all three values (month, day, and year) before your program determines that month is invalid?

6. What error do you get? Move the catch block/handler so that it is below the others. You shouldn't get the error now. Why do you think the error was produced when the handler appeared first?

7. Describe what happens and the output displayed. How is it different than the output displayed in step #13?

8. Unreported exception errors are produced. Why? Please be specific. Include the code required to fix these errors. Please ask your instructor if you do not know what code is required.

9. Recall that the setMonth method previously threw a RuntimeException and did not need to have a throws clause appended to the end of the method header. Why does it need a throws clause now?

10. Describe the code that is executed after the user enters 13, 12, and 2016. Be specific. What type of exception object is created? Where is the exception object created? Which statement in the try block is the last to be executed?

11. What is the difference between the keyword "throw" and the keyword "throws"?

Submit the following to your lab instructor:

- Date.java
- DateException.java
- Lab3aApp.java
- Your answers to questions #1 - 11.