CS3330: Lab 8

Due 48 hours after your lab ends

Purpose:

- Becoming familiar with the various exceptions thrown by frequently used functions.
- Learning how to handle various exceptions being thrown so that the user can fix their mistake.
- Learning how to write custom error handling procedures

Directions:

Familiarize yourself with the exceptions that various functions that you use frequently throw. You will be checking the java documentation frequently this lab. Don't ask questions like "what exception does this function throw?", that is for you to look up.

Complete each part of the lab assignment that follows using the description given in each section. Be sure to include comments as appropriate in your program. If your program does not compile or contains runtime errors, you will **receive no credit**.

Submission:

```
cs_submit CS3330_LAB-<section> LAB8 <pawprint>.cs3330.lab<#>.zip
```

Description:

You will be reading in some sample input of different types, instantiating and filling an ArrayList of these objects, and then calling toString on each of these objects. Please refer to previous labs for how to do the bulk of this lab. The focus of this week's lab is on the custom error handling.

Exception classes:

There will be three exception classes created and the names for these classes should be:

InvalidAgeException, InvalidNameException, and InvalidBalanceException.

Person class:

There will be one Person class that will have a name, age, and bank account balance variables. You will instantiate this class with variables that the user provides, but only after error checking the user's input. Make sure to override the toString method to match output.

Driver Class:

main:

In main, you should create an ArrayList of type Person. Prompt the user for the necessary variables and check their input for errors. Instantiate a Person object using the error checked input and add it to the ArrayList. When the user is done adding Person objects, they will type 'q' and information should be printed for each Person in the ArrayList using the toString method.

> **Error Checking in main:** The main point of this lab is the custom error checking. You will try to provide more helpful descriptions of what went wrong to the user and also prevent your program from crashing. ***AT NO POINT SHOULD YOUR PROGRAM CRASH!!!!!*** Also, main should only include Try and Catch statements, no throw or throws handling.

All three of the following error checking methods will make sure that the string passed to them is not empty. If so, throw the appropriate exception and tell the user as much.

isName(String input):

This method will additionally check to make sure that the input string by the user is a valid name. Names *CAN* contain spaces but should *NOT* contain anything else that is not a letter. For the purposes of this lab assume that we will only test English letters. This method will throw an InvalidNameException.

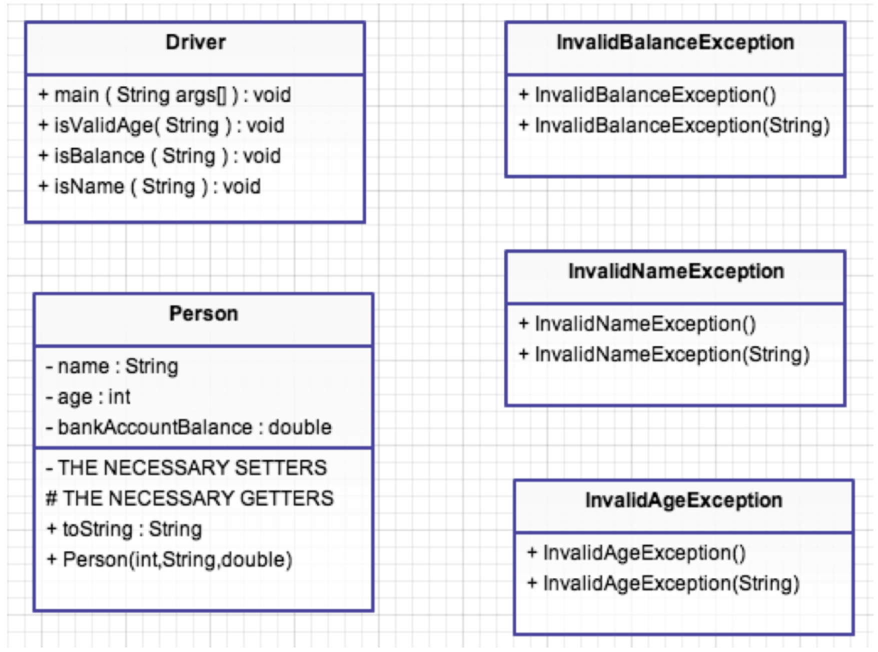
isValidAge(String input):

This method will additionally check to make sure that the age is not <= 0 OR >= 150. If so, you should throw the exception with two different messages alerting the user to which check they failed. Check the sample output to understand this. This method will throw an InvalidAgeException.

isBalance(String input):

This method will additionally check to make sure that the string passed to this function is indeed a double. This will necessitate a try-catch inside of this method. Remember that you can run any arbitrary code in a catch block. This method will throw an InvalidBalanceException.

UML Class Diagram:


Driver
+ main (String args[]) : void
+ isValidAge(String) : void
+ isBalance (String) : void
+ isName (String) : void
InvalidBalanceException
+ InvalidBalanceException()
+ InvalidBalanceException(String)
InvalidNameException
+ InvalidNameException()
+ InvalidNameException(String)
Person
- name : String
- age : int
- bankAccountBalance : double
- THE NECESSARY SETTERS
# THE NECESSARY GETTERS
+ toString : String
+ Person(int,String,double)
InvalidAgeException
+ InvalidAgeException()
+ InvalidAgeException(String)

Sample Output “Happy Case” (no exceptions):

```
Enter person info or q to quit.
Please enter the name of this person: Matthew England
Please enter an age for this person: 20
Please enter a bank account balance for this person: 0

Enter person info or q to quit.
Please enter the name of this person: Ankil Patel
Please enter an age for this person: 22
Please enter a bank account balance for this person: 250000000

Enter person info or q to quit.
Please enter the name of this person: Madison DeHart
Please enter an age for this person: 75
Please enter a bank account balance for this person: 2

Enter person info or q to quit.
Please enter the name of this person: q

Name: Matthew England
Bank Balance: 0.0
Age: 20

Name: Ankil Patel
Bank Balance: 2.5E8
Age: 22

Name: Madison DeHart
Bank Balance: 2.0
Age: 75
```

POSSIBLE EXCEPTIONS:

`Please enter the name of this person: ty!er`
`mremtf.cs3330.lab8.InvalidNameException: You have entered an invalid name.`

`Please enter an age for this person: @#$@#`
`mremtf.cs3330.lab8.InvalidAgeException: You did not enter an Integer.`

`Please enter the name of this person:`
`mremtf.cs3330.lab8.InvalidNameException: You did not enter a name.`

`Please enter an age for this person: 350`
`mremtf.cs3330.lab8.InvalidAgeException: Age can not be equal to or more than 150.`

`Please enter an age for this person: -24`
`mremtf.cs3330.lab8.InvalidAgeException: Age can not be 0 or negative.`

`Please enter an age for this person:`
`mremtf.cs3330.lab8.InvalidAgeException: You did not enter an age.`

`Please enter a bank account balance for this person: %$#%#$%`
`mremtf.cs3330.lab8.nvalidBalanceException: You did not enter a double.`

EXAMPLE OUTPUT WITH ERRORS

```
Enter person info or q to quit.
Please enter the name of this person: matt!
mremtf.cs3330.lab8.InvalidNameException: You have entered an invalid name.

Please enter the name of this person:
mremtf.cs3330.lab8.InvalidNameException: You did not enter a name.

Enter person info or q to quit.
Please enter the name of this person: matthew england
Please enter an age for this person: %$#%#$%
mremtf.cs3330.lab8.InvalidAgeException: You did not enter an Integer.

Enter person info or q to quit.
Please enter the name of this person: matthew england
Please enter an age for this person: -2
mremtf.cs3330.lab8.InvalidAgeException: Age can not be 0 or negative.

Enter person info or q to quit.
Please enter the name of this person: matthew england
Please enter an age for this person:
mremtf.cs3330.lab8.InvalidAgeException: You did not enter an age.

Enter person info or q to quit.
Please enter the name of this person: matthew england
Please enter an age for this person: 350
mremtf.cs3330.lab8.InvalidAgeException: Age can not be equal to or more than 150.

Enter person info or q to quit.
Please enter the name of this person: matthew england
Please enter an age for this person: 22
Please enter a bank account balance for this person: %$#%#$%
mremtf.cs3330.lab8.InvalidBalanceException: You did not enter a double.

Enter person info or q to quit.
Please enter the name of this person: matthew england
Please enter an age for this person: 22
Please enter a bank account balance for this person: 34.56

Enter person info or q to quit.
Please enter the name of this person: eric holder
Please enter an age for this person: 33
Please enter a bank account balance for this person: -20.67

Enter person info or q to quit.
Please enter the name of this person:q

Name: matthew england
Bank Balance: 34.56
Age: 22

Name: eric holder
Bank Balance: -20.67
Age: 33
```

Grading:

If your program does not compile, run all the way or produce any input/output (I/O) because most of the source code is commented out then your lab will receive a grade of zero. NO EXCEPTION!!!!!
Not Having Submission Code and other header comments in each of the class you create will also result in a grade of ZERO. NO EXCEPTION!!!!!

IF YOUR PROGRAM CRASHES AT ANY POINT YOU WILL LOSE 10 POINTS AUTOMATICALLY!!!!

OUTPUT: For this lab, the WHITE SPACE can be different. This means the blank lines between output and what-have-you. Otherwise, it should match very closely. Do not over-embellish. We should be able to tell if your output matches with a quick glance.

Rubric:

3 points (no partial) - Javadocs and other comments.

Three Exception Classes
3 points (each) – For correctly implementing the class.

Driver.java
2 points (each) – For correctly using a try-catch to see if the value is valid. Also, within this nine points, you can lose points for not following the instructions on how to implement the driver class.

3 points (each) – correctly implementing the methods that will check if the input is valid and throw a *helpful* exception message if they don't. You should NOT just throw the default exception.

3 points - Person.java

BONUS 5 points:
Create an interface class for isValidData which contains abstract methods listed in the driver class. Then have driver class use this interface. Also, fix the way double is printed out so that it looks like normal notation for money.