Lab 7

**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**.

**Due Date:**

24 hours after your lab ends

**Submission:**

**cs\_submit CS3330\_<section> LAB7 <pawprint>.cs3330.lab<#>.zip**

ex: cs\_submit CS3330\_01B LAB7 billybob.cs3330.lab7.zip

**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 (like what was done in the pre-lab).

**Description:**

This week your lab will be a smaller than usual lab, but don’t let that deceive you. We are exploring a new topic and you should not put this lab off until tomorrow.

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 structured almost exactly the same as the pre-lab. The names for these classes should be InvalidAgeException, InvalidNameException, and InvalidBalanceException.

Person class:

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

Driver Class:

main:

For main you should create an ArrayList of type Person and, after error checking the values inputted from the user, instantiate a Person object and add it to the ArrayList. After the user enters q to quit inputting Person objects you should call toString on each Person in ArrayList.

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.

***AT NO POINT SHOULD YOUR PROGRAM CRASH!!!!!***

This means you will need to use a try-catch for each variable the user gives you. If you completed the prelab, this lab should just be a matter of looking up the necessary methods to error check the variables.

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.

public static void 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.

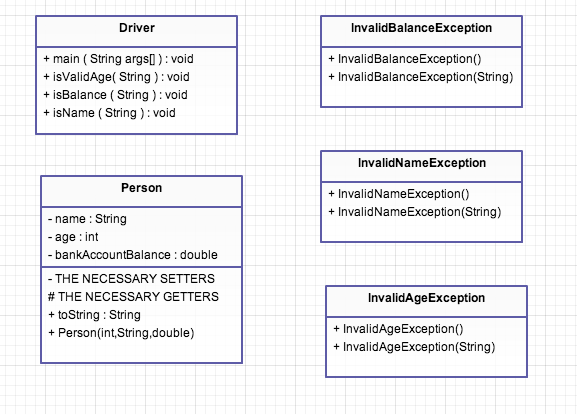
public static void isValidAge(String input):

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.

public static void isName(String input):

This method will additionally check to make sure that the string inputted 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.

UML Class Diagrams:



**Begin Sample Output:**

“Happy case”:

Enter person info or q to quit.

Please enter the name of this person:Tyler Nivin

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:Matt England

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: Tyler Nivin

Bank Balance: 0.0

Age: 20

Name: Ankil Patel

Bank Balance: 2.5E8

Age: 22

Name: Matt England

Bank Balance: 2.0

Age: 75

“Not-So-Happy case”:

Enter person info or q to quit.

Please enter the name of this person:ty!er

lab7.twn346.cs3330.InvalidNameException: You have entered an invalid name.

…

Please enter an age for this person: @#$@#

lab7.twn346.cs3330.InvalidAgeException: You did not enter an Integer.

…

Please enter the name of this person:

lab7.twn346.cs3330.InvalidNameException: You did not enter a name.

**…**

Please enter an age for this person: 350

lab7.twn346.cs3330.InvalidAgeException: Age can not be equal to or more than 150.

**…**

Please enter an age for this person: -24

lab7.twn346.cs3330.InvalidAgeException: Age can not be 0 or negative.

…

Please enter an age for this person:

lab7.twn346.cs3330.InvalidAgeException: You did not enter an age.

…

Please enter a bank account balance for this person: %$#%#$%

lab7.twn346.cs3330.InvalidBalanceException: You did not enter a double.

**Grading:**

**General: 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 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 AUTMOTACIALLY!!!!**

**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 is as follows**

**3 points (no partial) -** Header comments and submission code in every class file. You also must have good comments.

**Three Exception Classes**

**3 points (each) –** For correctly implementing the class.

**Driver.java**

**3 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 instructions in 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.

**BONUS 5 points:**

Create an interface class for isValidData which contains abstract methods listed in 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.