CIS 2571: Introduction to Java
Lab Assignment

Name: ______________________________

Lab Assignment	#8 – Exception Handling and Text I/O
Due Date *(beginning of class)*	04/07/2014
Points	Short Answer Questions ___________/ 10 pts. Exception Handling Java Program source code files • AccountWithException.java (*modified Account.java file*) • InvalidBalanceException.java (*new*) • TestAccountWithException.java (*new*) output (console/GUI and file) • valid balance • invalid balance • receipt file uploaded .zip ___________/ 60 pts. **Total ___________/ 70 pts.**

<u>Lab Assignment #8 Activities</u>

1. Answer the questions in the spaces provided or attach a sheet with your answers (be sure to label questions for proper credit) *(2 points each, unless noted otherwise)*

Question	Your Answer
a. What classes (and their subclasses) are examples of **unchecked** exceptions?	
b. What are the **two different ways** that a programmer can deal with **checked** exceptions to avoid compiler errors? *(4 points)*	**Option #1:**
	Option #2:

c. Describe the steps that occur when an **exception** is **not caught** in the current method.	
d. How is a '**chained exception'** different from an exception that has been **rethrown**?	

2. You will modify a version of the **Account** class from previous labs to include exception handling and file output. The **Account.java** file will be given to you to modify.

   a) Create a custom exception class called **InvalidBalanceException** that is a subclass of the **Exception** class. It should be a public class in its own file and should contain a private data member (and public accessor method) to hold the invalid balance. The **InvalidBalanceException** constructor **takes an argument of an invalid account balance and invokes the superclass constructor with a descriptive message that includes the invalid balance amount**. *(10 points)*

   b) Modify a copy of the **Account.java** class file (given in class) to throw your newly created exception class when an illegal balance (< 0.0) is attempted to be set for an account. The public class, and file, should be called **AccountWithException**. *(14 points)*

   c) Create a public **TestAccountWithException** class, in its own java file, with a main method that accomplishes the following program requirements:

      - Prompt the user for an output filename. Create a file for text output (See **14.11.1 Writing Data Using PrintWriter**). *(10 points)*
        - If the file exists, prompt the user for a new filename.
        - Do not continue until a valid, non-existing, filename has been given and the file opened for output.
      - Prompt the user for a first name, last name, and balance (account id and annual interest rate can be constants). Create an **AccountWithException** object that includes the user specified data. *(4 points)*
      - When valid data exists in the **AccountWithException** object (i.e. **InvalidBalanceException** has **not** been thrown), output the object information to the display **as well as** the opened output file. Use the overloaded **toString()** method when outputting the data. *(6 points)*

- o Use the **finally** block to ensure the output file has been properly closed. *(2 points)*
- o Include exception handling to catch any thrown **InvalidBalanceExceptions** or **FileNotFound Exceptions** and display unique output identifying the specific exception. *(4 points)*
- o Use either the **GUI or console** for both input and display output of account information. *(2 points)*

d) Add a block comment at the top of each file to identify your name, file, date, class, assignment, and short description of the included class. **Use proper code alignment for full credit.** *(3 points total, 1 point each)*

e) Compile the source code until no errors are found.

- Common Errors: http://www.cs.armstrong.edu/liang/intro9e/debug.html

f) Run the Java bytecode and observe the results.

g) Attach a hardcopy printout of your source code files.

h) Attach a hardcopy printout of your sample output of console/GUI and file for the following test cases *(label each appropriately)*: *(3 points)*

- valid balance
- invalid balance
- generated output receipt

i) Create a **.zip** file containing only your **.java** source code files. Upload a copy of this **.zip** file to the appropriate assignment in Blackboard (this will be demonstrated during class, if necessary). *(2 points)*

- See the following link for a video on submitting assignment in Blackboard:
  http://ondemand.blackboard.com/r91/movies/bb91_student_submit_assignment.htm
- See the following link for a video on creating **.zip** files in Windows XP:
  http://www.youtube.com/watch?v=3xqF56OZo_k
- See the following link for instructions on creating **.zip** files:
  http://condor.depaul.edu/slytinen/instructions/zip.html