CSC 360 Programming Assignment 8
Due Date: Monday, November 10

In this assignment, you will implement a Generic class and then implement a user class to use the Generic class. In your user class, you will specify the types to be used in your Generic class. The Generic class will also throw an Exception that you will have to define, to be caught in your user class.

The basic idea of this assignment is to implement a key-value pair generic class which might be used to populate a database. But for security reasons, we want to enhance this concept by requiring that aside from a matching key, you also need the proper security code. The security code will be called a lock. So our items will be key-lock-value triples. Because a key, a lock and a value can be of any type of object, we need to employ Generics. We would expect the key and lock to be limited to Comparable types since we will have to compare the value and lock to see if we have the right triple. We would actually expect the key and lock to be a <String> or <Integer> but possibly <Long> or other similar numeric wrapper types. The value can be any type of object and will not be Comparable.

The UML for the key-lock-value class is given below. Notice that we will not have to implement Comparable for the key or lock since those types are forced to be Comparable (that is, we restrict those types to already be Comparable). The user program will receive a syntax error if it tries to create a key-lock-value triple in which the key and/or lock is not of a Comparable type. You will explore this in the assignment.

```
SecureKeyValue<A extends Comparable<A>, B extends Comparable<B>, C>
-key: A
-lock: B
-value: C
-------------------------------------------------------------------------------------------------------------------
+SecureKeyValue( )                                  // no-arg constructor, worthless for us
+SecureKeyValue(key: A, lock: B)                    // sets a key with a security lock, but no value (yet)
+SecureKeyValue(key: A, lock: B, value: C)          // typical constructor
+getKey( ): A                                       // accessor for key
                        // note:  there are no direct accessors for lock or value because we do not
                        // want someone to access the value without the key and lock, nor do we
                        // want to just return the lock
+getValue(lock: B): C throws ErroneousLockException                  // returns value if lock is correct
+setValue(lock: B, value: C): void  throws ErroneousLockException   // change or set value
+lockCorrect(lock: B): boolean                                       // used to test the lock
+changeKey(key: A, lock B): void throws ErroneousLockException      // change key if lock is correct
+changeLock(lock: B, newLock: B): void throws ErroneousLockException // change lock
                                                    // if lock is correct
                        // note:  there is no toString since we want to keep the lock a secret, we could
                        // implement toString(lock: B): String but we will not
```

Aside from the above class, implement the ErroneousLockException exception which will only need a no-arg and 1-arg constructor.

Next, implement a user class. The user class will create a series of KeyLockValue objects and pass them messages. You will obtain the specific code from the website. For each KeyLockValue object, you will have to declare and instantiate it and pass it messages. These should all be done within a try-catch block. Each one (there are 10 in all) will have its own try-catch block so that the program does not terminate

after the first exception is thrown and caught. Some of the 10 that you will implement will cause syntax errors. Upon discovering the error, fix the code and in comments explain why the original version caused an error and what you did to correct it. Aside from the ErroneousLockException, some of these 10 will cause other Exceptions. Make sure you implement as many catch blocks as are necessary to handle the possible Exceptions. Every one of the try-catch blocks should implement ErroneousLockException. You only need to implement other catch blocks as needed.

NOTES:

- You will need to import java.math.*; to handle BigDecimal.
- You will need to also use your Fraction class from earlier in the semester.
- You will also need to download and compile the Person class (available on the website).
- You may need to add @SuppressWarnings("unchecked") to your User class (before main).

Capture the output when running the program and submit your three classes (SecureKeyValue, ErroneousLockException, User) with the output.