Exception exercises

A. Consider the following method

```
public int dummy() throws IOException {
        int value = 0;
        try {
                value = 1;
                contractor.doIt();
                value = 2;
        } catch (NullPointerException e) {
                value += 10;
        } catch (RuntimeException e) {
                value += 20;
        }
        return value;
}
```

What value does the method return if contractor.doIt:

1. throws a NullPointerException
2. throws an ArithmeticException
3. throws an IOException
4. throws an AssertionError
5. completes successfully

B. Consider the following method

```
/**
 * The number of lines in the specified file
 */
```

```
public int lineCount(File file) {
        BufferedReader reader = new BufferedReader(new
        FileReader(file));
        int count = 0;
        String line;
        while ((line = reader.readLine())!= null) {
                count ++;
        }
        reader.close();
        return count;
}
```

Complete the code above so that in case of an exception thrown by an IO method,

1. the method ignores the exception and lets it propagate to the client.
2. the method catches the exception and returns -1
3. the method catches the exception, writes out an error message, and terminates the program.
4. the method catches the exception, writes out an error message, and returns a value of -1 to the client.
5. the method catches the exception, and throws another exception to the client.

C. Consider the following method for posting a withdrawal from an account

```
        public void withdraw(int amount) {
                this.balance -= amount;
                notifyObservers();
```

```
            log.logWithdrawal(this.accountNumber,
amount);
        }
```

A withdrawal is not complete unless it is successfully logged. But the method logWithdrawal might return a LogFailureException if the entry cannot be logged.

Modify the method so that it catches the exception and resets the balance to its original value before throwing an UnsuccessfulWithdrawalException to its client. Assume that notifyObservers must be done after updating the balance and before logging the withdrawal.

Answers for part A:

1. throws a NullPointerException
   returns 11
2. throws an ArithmeticException
   returns 21 (ArithmeticException is derived from RuntimeException)
3. throws an IOException
   doesn't return any value since IOException is not caught in dummy
4. throws an AssertionError
   doesn't return any value since AssertionError is not caught in dummy (an Error shouldn't actually be caught anywhere)
5. completes successfully
   returns 2