Midterm 3 (March 31)

THIS IS A REAL MIDTERM! This is the same structure as the practice midterm, but this midterm will have 3 questions.

The Midterm Exams are 3 exams consisting of 3 questions of 15 minutes each including upload time. You have a 5 minute grace period to upload answers to the question, and you must answer them in order.

The midterm exams are closed book solo effort. No collaboration is allowed. You will be expected to complete a question in 15 minutes, before moving on to the next question. You have a 5 minute grace period after the 15 minutes to upload that question.

In taking this exam, you agree to abide by the university policies on academic integrity. The exam is technically 50 minutes because each question will have a 5 minute grace period for submission.

You may have 2 sheets US Letter size or A4 paper with hand written notes on it. This is a synchronized midterm and it is **closed book** , but you may use 2 page paper cheatsheet that you made yourself.

Identify yourself by your CCID in each answer (a note on a diagram, a comment on a text/java file).

There are 3 questions (only 1 question for practice midterm). Each question has its own submission page:

* Question1 10:00am MT March 31 10:00 to 10:15am MT (10:15 to 10:20 grace period for submission)

https://eclass.srv.ualberta.ca/mod/assign/view.php?id=5099204

** This link will not open until the question is open

* Question2 10:15am MT March 31 to 10:30am MT (10:30 to 10:35 grace period for submission)

https://eclass.srv.ualberta.ca/mod/assign/view.php?id=5099205

** This link will not open until the question is open

* Question3 10:30am MT March 31 to 10:45am MT (10:45 to 10:50 grace period for submission)

https://eclass.srv.ualberta.ca/mod/assign/view.php?id=5099206

** This link will not open until the question is open

*** If your submission is more than 5 minutes late you (outside of grace period) you must submit an explanation using this assignment. Use the textbox.**

* If you have accommodations they will already be added for you, you will have more than your time modifier.

Technical Issues: Submitting to this page/assignment is to indicate technical difficulties you experienced, if we have to discuss them later. If you experience an internet outage or something similar that prevents submission you submit a response to this page.

If you miss the deadline for a question **submit anyway** and use this page to explain why your submission is late and why I should mark it. Late submissions will be accepted by eclass but reviewed at instructor's discretion.

You do not need to submit this page unless you have technical difficulties.

Each question on the midterm is marked as follows, examples are in []:

Excellent: 5 marks: Student's answer is correct, without flaw. Answer demonstrates ability to synthesize material at various levels of abstraction. Breadth and Depth of material is demonstrated and good judgement is used in providing the answer. [Perfect UML w/ 1 missing mulitiplicity] [Java code with 1 missing argument type]

Good: 4 marks: Like excellent but something is minor is impacted or missing. Not as consistent as excellent. [Perfect UML w/ 1 missing minor relationship] [Java code missed an exception]

Satisfactory: 3 marks Demonstrates they UNDERSTAND the core material but not subtleties. Can apply what's learned in class to simple examples or parts of the exam. There are issues with the answer but understanding of the question and answer are clear. [UML exists but is off] [Java code gets to the point but has problems]

Unsatisfactory: 2 marks. Some understanding is demonstrated but the answer is not satisfactory and is lacking. [UML exists but is not really correct] [Java code shows evidence of knowing what the problem and what the solution is but isn't getting there]

Fail: 0 marks. The student didn't understand the question and gave an irrelevant answer. The student gives no answer. The answer is not sufficient to demonstrate understanding. [UML exists but does do what is asked] [Wrong UML diagram] [Java code is not java at all]

Midterm 3 Question 1

```
/*
*Submit PDF, PNG, or JPEG only.*

*5 marks.*

*Sequence Diagrams and Composite Pattern*

* YOUR CCID MUST BE ON YOUR SEQUENCE DIAGRAM *

Restructure this code with the composite pattern and provide a UML
sequence diagram depicting the the sequence of operations one would
observe running the the `testDriver` method in the `TestDriver` class
once you have *refactored* this code using the composite pattern.

This code is just a simple expression tree with interger literals
and a multiplication operator.

*** Provide UML Sequence diagram for Composite Pattern'd TestDriver().testDrive()
***
*** Provide the code for the new TestDriver.testDrive method ***

*** Do not diagram TestDriver.getTestExpression() ***

*/
abstract class Expression {
 boolean isLiteral() { return false; } // is it a literal?
 boolean isMul() { return false; }     // is it a Multiply?
 int getValue() { throw new RuntimeException("Not a literal"); }
}
/* represent multiply operation */
class MulOp extends Expression {
 public Expression left;
 public Expression right;
 MulOp(Expression left, Expression right) {
  this.left = left;
  this.right = right;
 }
 boolean isMul() { return true; }
}
/* represent an integer value */
```

```
class Literal extends Expression {
 public int value;
 Literal(int value) {
  this.value = value;
 }
 boolean isLiteral() {
  return true;
 }
 int getValue() { return value; }
}
/* recursively evaluate an expression */
class RecursiveEvaluator {
 Expression evaluate(Expression e) {
  if (e.isLiteral()) {
   return e;
  } else if (e.isMul()) {
   return new Literal(evaluate(((MulOp)e).left).getValue() *
      evaluate(((MulOp)e).right).getValue());
  }
  throw new RuntimeException("Unknown Expression");
 }
}
class TestDriver {
 // This is a setup method that you don't have to diagram
 Expression getTestExpression() {
  // do not diagram the internals of this method
  Expression exp = new MulOp(
   new MulOp(
    new Literal(5),
    new Literal(6)
   ),
   new Literal(3)
  );
  return exp;
 }
 // refactor this and provide the code you refactored it to.
 void testDriver() {
  Expression exp = getTestExpression();
  // these 2 lines can change when you refactor
  Expression result = new RecursiveEvaluator().evaluate(exp); // 1
  assert( result.getValue()==90 );                            // 2
 }
}
```

Midterm 3 Question 2

```
/*
 *Submit PDF, PNG, or JPEG only.*


 *5 marks.*

* Mock Objects and Testing and UML Class Diagram *
```

* YOUR CCID **MUST** BE ON YOUR UML CLASS DIAGRAM *

Draw a **UML class diagram** of this code, and the **TestWaterMeter** class and a **testFlowShutdown** method, and mock objects meant to test the code on lines **47** and **48** (**47:** and **48:** in the code). This Java code is meant to be a water treatment plant. **Draw a well-designed UML class diagram** to represent this information. Provide the basic abstractions, attributes, methods, relationships, multiplicities, and navigabilities as appropriate. `...` or `// ...` means much code is omitted. ***Exceptions are classes, Exception is a class.***

```
*/
class WaterException extends Exception {
 WaterException(String s) {
  super(s);
  // ...
 }
 // ...
}
class TooMuchFlowException extends WaterException {
 TooMuchFlowException(String s) {
  super(s);
  // ...
 }
 // ...
}
interface WaterMeasurable {
 double litresPerSecond( WaterMeter pm ) throws WaterException;
}
interface Valve {
 void shutdown();
}
class WaterMeter {
 Valve[] valves;
 // ...
 boolean shutdown = false;
 double measureFlow( WaterMeasurable wm ) throws WaterException {
  try {
   double lps = wm.litresPerSecond( this );
   return lps;
  } catch (TooMuchFlowException e) {
/*47:*/     shutDownFlow();
/*48:*/     throw new WaterException("Too Much Flow triggers shutdown");
  } catch (WaterException e) {
   throw e;
  }
 }
 void shutDownFlow() {
  shutdown = true;
  // shutdown valves
  // ...
 }
```

```
 // ...
}
```

Midterm 3 Question 3

```
/*
 *Submit 1 Plain Text, 1 .java, or 1 .zip of .java only.*

 I prefer 1 text file over a .zip! Use an editor!

 *5 marks.*

*Template Method Pattern*

* YOUR CCID MUST BE ON AT THE TOP OF YOUR JAVA CODE *

Refactor this Java code using the *template method pattern*.
Conditionals must be replaced by polymorphism as per the template
method pattern. Provide the Java code for the refactored
PreferenceReader, EncryptedFilePreferenceReader, and
UnencryptedHttpUrlPreferenceReader. You can use the magic '...' to
indicate missing code, but you should name your methods well enough
for anyone to understand what is missing.

*/

import java.io.*;
/* DecryptInputStream can decrypt an encrypted inputstream */
class DecryptInputStream extends BufferedInputStream {
 public DecryptInputStream(InputStream in, byte[] header) {
  super(in);
  // ...
 }
 // ...
}
class HttpInputStream extends InputStream {
 public HttpInputStream(String httpURL) {
  // ...
 }
 public int read() throws IOException {
  // ...
 }
 // ...
}
class Preference {
 Preference(String strUserPrefs) {
  // ...
 }
 // ...
}
class PreferenceReader { // Refactor me
 // ...
```

```
/* This reads userpreference data from a file or a HTTP URL
* and then constructs the user's preference from that string
*/
Preference readPref(String fileOrURL, boolean isURL, boolean isEncrypted)
 throws IOException {
 InputStream in;
 if (isURL) {
  in = new HttpInputStream( fileOrURL );
 } else {
  in = new FileInputStream( fileOrURL );
 }
 if (isEncrypted) {
  byte[] buffer={0,0,0,0};
  // We need the first 4 bytes for the encryption header
  // We read them and then push them back onto the stream
  // allow us to return to this part in the stream
  in.mark(0);
  // read 4 bytes to check for encrypted
  in.read(buffer,0,4);
  in.reset(); // unread those 4 bytes, push them back onto the stream
  in = new DecryptInputStream( in, buffer );
 }
 // OK now we can read the string that represents the Preference
 BufferedInputStream bin = new BufferedInputStream(in);
 ByteArrayOutputStream buf = new ByteArrayOutputStream();
 int result = bin.read();
 while (result != -1) {
  result = bin.read();
  buf.write((byte) result);
 }
 // convert the read input to a UTF-8 String and return a new user pref.
 return new Preference( buf.toString("UTF-8") );
}
}
```