CMPUT301 Winter 2021 Midterm 1

Midterm 1 (Feb 1)

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 10 minutes each. You have 10 minutes per question, 5 minutes to upload, 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 10 minutes and then upload within 5 minutes before moving on to the next 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 it is **closed book** , but you may use 2 page paper cheatsheet that you made yourself.

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

* Question1 10:00am MDT February 1 10:00 to 10:15am MDT February 1 (10:15 to 10:20 grace period for submission)

** This link will not open until the question is open https://eclass.srv.ualberta.ca/mod/assign/view.php?id=4986852

* Question2 10:15am MDT February 1 to 10:30am MDT February 1 (10:30 to 10:35 grace period for submission)

** This link will not open until the question is open https://eclass.srv.ualberta.ca/mod/assign/view.php?id=4993268

* Question3 10:30am MDT February 1 to 10:45am MDT February 1 (10:45 to 10:50 grace period for submission)

** This link will not open until the question is open https://eclass.srv.ualberta.ca/mod/assign/view.php?id=4993269

* 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 extension]

Satisfactory: 3 marks Demonstrates they UNDERSTAND the core material but not subleties. 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 1 Question 1

Submit PDF, PNG, or JPEG only.

5 marks.

Java to UML Class Diagram

Convert this Java code to a UML class diagram. This Java code is meant to represent trials and experiments. Draw a well-designed UML class diagram to represent this information. Provide the basic abstractions, attributes, methods, relationships, multiplicities, and navigabilities as appropriate. "...” means much code is omitted.

```
public abstract class Result {
    public abstract boolean getBinary();
    public abstract int getCount();
    public boolean isBinary() { return false; }
}
public class Trial {
    protected Result result;
    public Trial(Result r) {
        result = r;
    }
    public Result getResult() {
        return result;
    }
}
public class BinaryResult extends Result {
    boolean success;
    public BinaryResult(boolean success) {
        this.success = success;
    }
    public boolean getBinary() {
        return success;
    }
    public int getCount() {
        return 0;
    }
    public boolean isBinary() { return true; }
}
public class MultiBinary extends Result {
    private BinaryResult[] results;
    public MultiBinary(BinaryResult[] results) {
        this.results = results;
    }
    public boolean getBinary() {
        return false;
    }
    public int getCount() {
        int count = 0;
        for (BinaryResult b: results) {
            if (b.getBinary()) {
                count = count + 1;
            }
        }
        return count;
    }
}
```

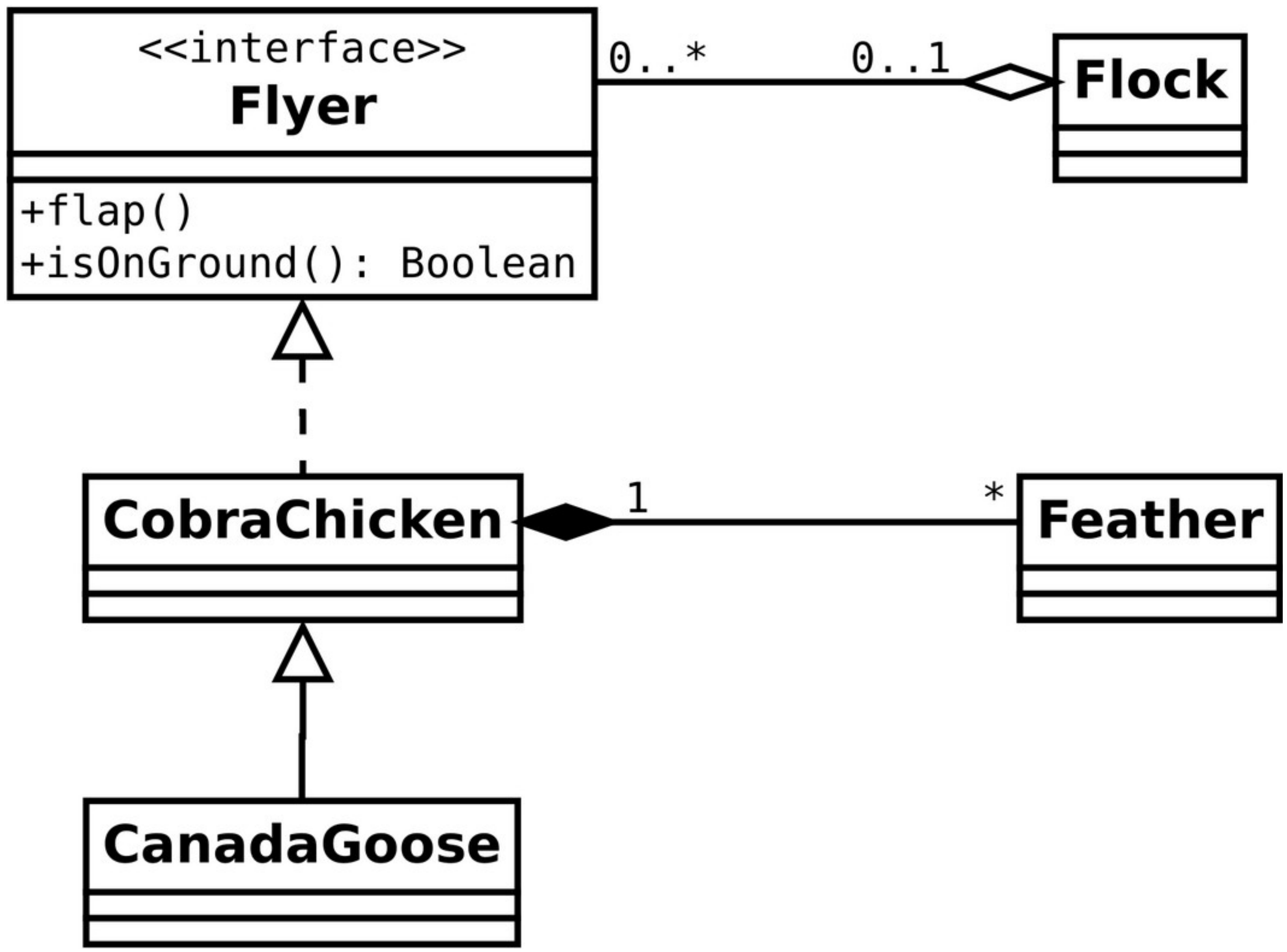
Midterm 1 Question 2

Plain text preferred: .txt or .java
Submit .txt, .java, a zip of .java, PDF, PNG, or JPEG only.

5 marks.

Convert this UML class diagram of an a goose based videogame to skeletal Java Code. Include all attributes and obviously public methods. Make sure all required methods are implemented. Include all generalizations and necessary associations. If you need space feel free to use the back of the page. You may use ellipses (...) to indicate there is some code in the method but it's a midterm and who has time to write that code!

ACCOMMODATIONS: If you need a textual version of this UML Q2.pdf and Q2.svg are available for you.



Midterm 1 Question 3

Submit PDF, PNG, or JPEG only.

5 marks.

Liskov Substitution Principle

This Java code is poor! It violates Liskov Substitution principle!
Restructure this code as UML and **draw a UML Class Diagram** of **restructured code** that *doesn't violate the Liskov Substitution principle.*
Draw a well-designed UML class diagram to represent

this information. Provide the basic abstractions, attributes, methods, relationships, multiplicities, and navigabilities as appropriate.
"...” means much code is omitted.

This Java code is about the punishment of Sisyphus who cheated death twice only to get caught by Zeus. Zeus forced Sisyphus to move a boulder up a hill everyday only to have it roll back down the hill so he'd have to move it up the hill the next day.

Be sure to leave a note on the diagram highlighting how you address the violation

```
public interface Location {
    // ...
}
public class Boulder {
    protected Location location;
    public Location getLocation() { return location; }
    public void moveTo(Location l) { location = l; }
}
public class PermanentBoulder extends Boulder {
    // This boulder doesn't move!
    public PermanentBoulder(Location l) {
        this.location = location;
    }
    public void moveTo(Location l) {
        System.err.println("Don't move me");
    }
}
public class Sisyphus {
    // A boulder mover
    protected Location topOfTheHill;
    protected Location bottomOfTheHill;
    public Sisyphus() {
        // ...
    }
    public void push(Boulder b, Location l) {
        chalkHands(); // chalk up your hands, get ready to move a boulder
        b.moveTo(l);
    }
    public void punishment(Boulder b) {
        push(b, bottomOfTheHill);
        push(b, topOfTheHill);
        push(b, bottomOfTheHill);
        // enforce constraint: boulder is now at the bottomOfTheHill
        assert(b.getLocation() == bottomOfTheHill);
    }
    protected chalkHands() {
        //...
    }
}
public class MiniSisyphus extends Sisyphus {
    protected chalkHands() {
        // ... uses a little less chalk (mini hands!)
    }
}
```