UCID		Name	
Measure	**Weight**	**Evaluation**	**Score**
Design	50%	Design is evaluated by the UML submission.	0.00%
Testing	50%	Testing is evaluated by running the tests, reviewing the tests, and calculating test coverage.	0.00%
		Total	0.00%
		The grade will be entered in D2L as the following value out of 100 (rounded to nearest 10th):	0.0

		Category	Rating	Options
Design (30%)	The UML diagram details the program design, showing class relationships, methods, class and instance variables, parameter types, return value types, access modifiers (public, private, etc), other modifiers (i.e., static, final and abstract) and cardinalities. The class containing main() does not need to be shown if it contains no other methods. The database can be depicted as a single rectangle.	Readability		Yes/No
	The diagram is consistent (e.g., relationships are supported by data members) and appears to be an accurate representation of the code.	Readability		Yes/No
	Diagram is legible - lines are tidy, text can be read, and fits on the page(s) neatly. The file is a PDF.	Readability		Yes/No
	Each class has a clear, distinct purpose (encapsulation)	OOP		90-100% (8), 70-89% (6), 50-69% (4), <50% (0)
	Inheritance and/or interfaces are used in the design, and used appropriately	OOP		
	Aggregation and/or composition is used in the design, and is used appropriately	OOP		
	Pattern (Singleton, Builder, Observer, Strategy, or M-V-C) and/or generics is used in design, and is used appropriately	OOP		
	Method overloading is used (polymorphism), and is used appropriately	OOP		Yes (2) /No (0)
	Single Responsibility Principle is followed throughout (if Singleton pattern is used appropriately, violation of SRP is permitted in this instance)	OOP		90-100% (4), 70-89% (3), 50-69% (2), <50% (0)
	Open-Closed Principle is followed throughout	OOP		
	Liskov Substitution Principle is followed throughout	OOP		
	Interface Substitution Principle is followed throughout	OOP		
	Dependency Inversion Principle is followed throughout	OOP		
	Public methods are not assumed to receive correct imput and throw exceptions accordingly	Java		90-100% (4), 70-89% (3), 50-69% (2), <50% (0)
	Class names are appropriate (singular, indicate purpose)	Java		
	Method names indicate expected behavior and follow naming conventions	Java		
	Variable types (including data structures) are appropriate for purpose	Java		
Grader Remarks				Total marks
				Percentage

		Category	Rating	Options	Points	Out of
Testing (25%)	The edu.ucalgary.oop package is used	Compatibility		Yes/No	0	1
	The tests appear to be correct Java	Compatibility		Yes/No	0	4
	There are no tests which involve connecting to the database or user interface.	Compatibility		Yes/No	0	1
	All test files include 'Test' in the name. Only test files include 'Test' in the name.	Compatibility		Yes/No	0	1
	One class is examined manually at random to evaluate the next test coverage assessment. Name of evaluated class:					
	Edge cases and boundary conditions have been considered.	Coverage		All (15), 1-3 missing (10), 4-7 missing (5), > 7 missing (0)	0	15
	Three tests in different test files will be randomly spot-checked for the following assessments. Names of evaluated tests:					
	Tests use AAA structure.	Design		All 3 tests meet criteria (3), two tests meet criteria (2), one test meets criteria (1), no tests meet criteria (0)	0	3
	Tests have meaningful names which convey information about testing scenario.	Design			0	3
	Tests follow Java naming convention (begin with 'test').	Design			0	3
	Tests have useful error messages (except in cases where exception is expected, this point is automatically given).	Design			0	3
	Tests only test a single scenario (usually only one assert statement).	Design			0	3
	Tests are self-contained (setup information may be in @Before).	Design			0	3
Grader Remarks				**Total marks**	0	40
				Percentage	0.00%	