Faculty of Arts and Science
University of Toronto

Midterm Test

Department:	Computer Science
Instructor:	Matt Medland
Date & Time:	10:10 a.m. – Thurs., Feb. 27, 2014

Conditions: Closed Book
Duration: 50 minutes

This test counts for 15% of your final grade

Name: SOLUTIONS.
(Please underline last name)

Student Number: ____________________

Question Marks

1	______	/60
2	______	/15
3	______	/10
4	______	/10

Total ______ /95 = ______ %

1. [Modeling & UML Questions, 60 marks Total]
(a) [Composition vs. Aggregation – 5 marks] UML class diagrams show relationships between classes. Two similar relationships between classes are **aggregation** and **composition**. The diagram below, taken from the lecture notes, illustrates these relationships:

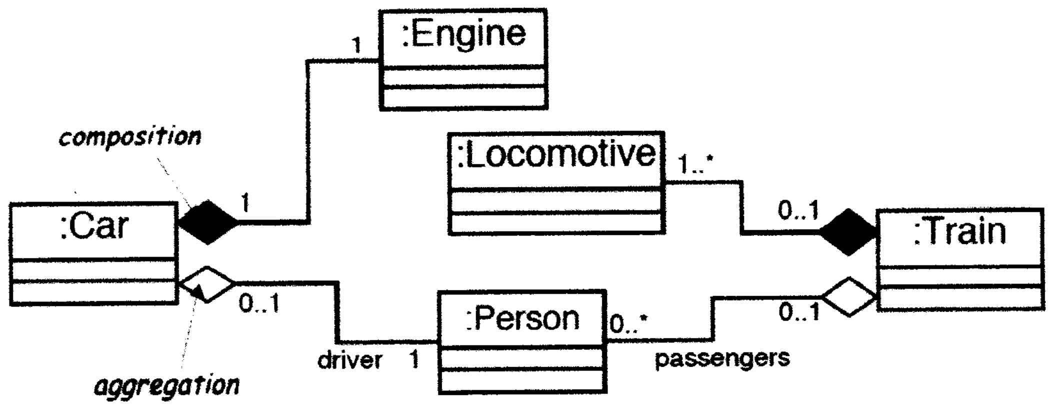


Figure 1: aggregation vs. composition

Describe the difference between **aggregation** and **composition**. You may refer to the diagram if you like.

AGGREGATION IS A HAS-A RELATIONSHIP, BUT COMPOSITION IS EVEN STRONGER. COMPOSITION IMPLIES OWNERSHIP, OR A WHOLE-PART RELATIONSHIP. FURTHER, IF THE WHOLE IS REMOVED, SO ARE ALL THE PARTS. IN THE EXAMPLE, IF THE CAR IS REMOVED, SO IS THE ENGINE, BUT NOT THE PERSON.

(b) [Implementing Aggregation & Composition – 5 marks] Are **aggregation** and **composition** relationships realized (implemented) the same, or different, in typical object-oriented programming languages? If different, how?

IN GENERAL, THEY ARE IMPLEMENTED THE SAME. HOWEVER, IN LANGUAGES THAT SUPPORT INNER-CLASSES (EX. JAVA) THE "OWNED" CLASS COULD BE AN INNER-CLASS. (EX. LOCOMOTIVE INNER-CLASS IN TRAIN) OTHER EXAMPLES MAY INCLUDE CREATING & DESTROYING PRIVATE MEMBERS IN CONSTRUCTORS/DESTRUCTORS, BUT THIS IS OBJECT, NOT CLASS, LEVEL.

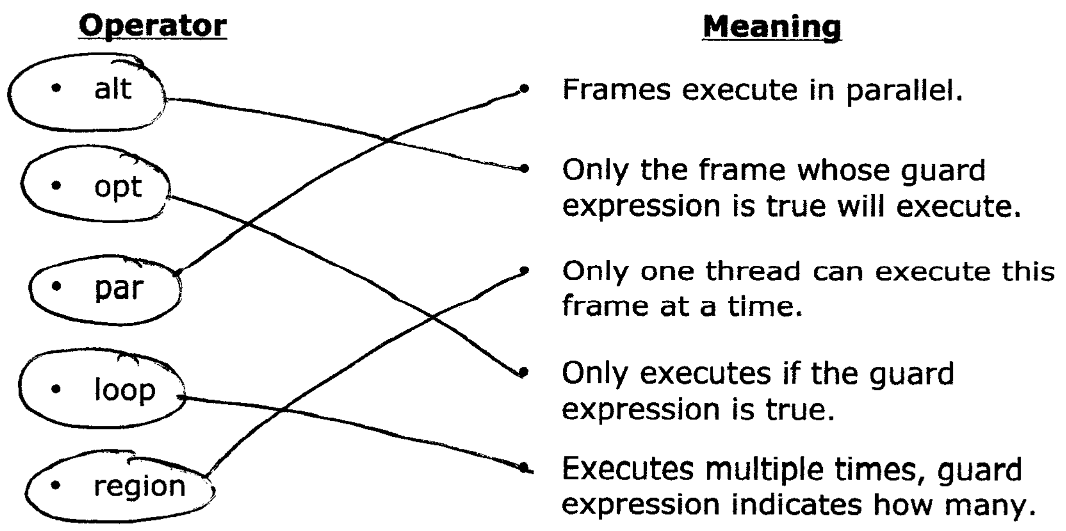
(c) [Static vs. Dynamic modeling – 10 marks] Describe, briefly, the differences between **dynamic** and **static** modeling, as it applies to software systems. List some diagram types, UML or otherwise, used in static and dynamic modeling.

STATIC MODELING IS USED TO MODEL THE STRUCTURE, OR DECOMPOSITION OF THE STATIC PARTS OF A SYSTEM (EX. CLASSES, CODE, MODULES, PACKAGES, ETC.) DYNAMIC MODELING DEALS WITH THE CHANGING & RUNTIME ATTRIBUTES OF THE SYSTEM.

DIAG. FOR STATIC: CLASS, PACKAGE, COMPONENT, DEPLOYMENT.

DIAGS FOR DYNAMIC: USE CASE, SEQUENCE, COLLABORATION, STATE CHART, ROBUSTNESS, ACTIVITY.

(d) [Name the Frame – 5 marks] Interaction frames, or **combined fragments**, show groups of related messages in UML sequence diagrams. The related messages are grouped to show conditional, repeated, or otherwise different, flow in the sequence diagram. Execution of the frame is determined by its (optional) guard expression. The frame operators discussed in lecture, along with their meanings, are listed below. Draw lines to connect the operator to its meaning.


Operator
Meaning
alt
opt
par
loop
region
Frames execute in parallel.
Only the frame whose guard expression is true will execute.
Only one thread can execute this frame at a time.
Only executes if the guard expression is true.
Executes multiple times, guard expression indicates how many.

(e) [Sequence Diagram – 15 marks] The Transmission Control Protocol, or TCP, uses a "handshake" process to establish a connection between client and server. The process (slightly modified) is described on the TCP Wikipedia page as below:

TCP Connection Establishment

To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections. Once the server is listening, a client may initiate a connection. To establish a connection, the three-way (or 3-step) handshake occurs:

1. SYN: The connection is initiated by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.

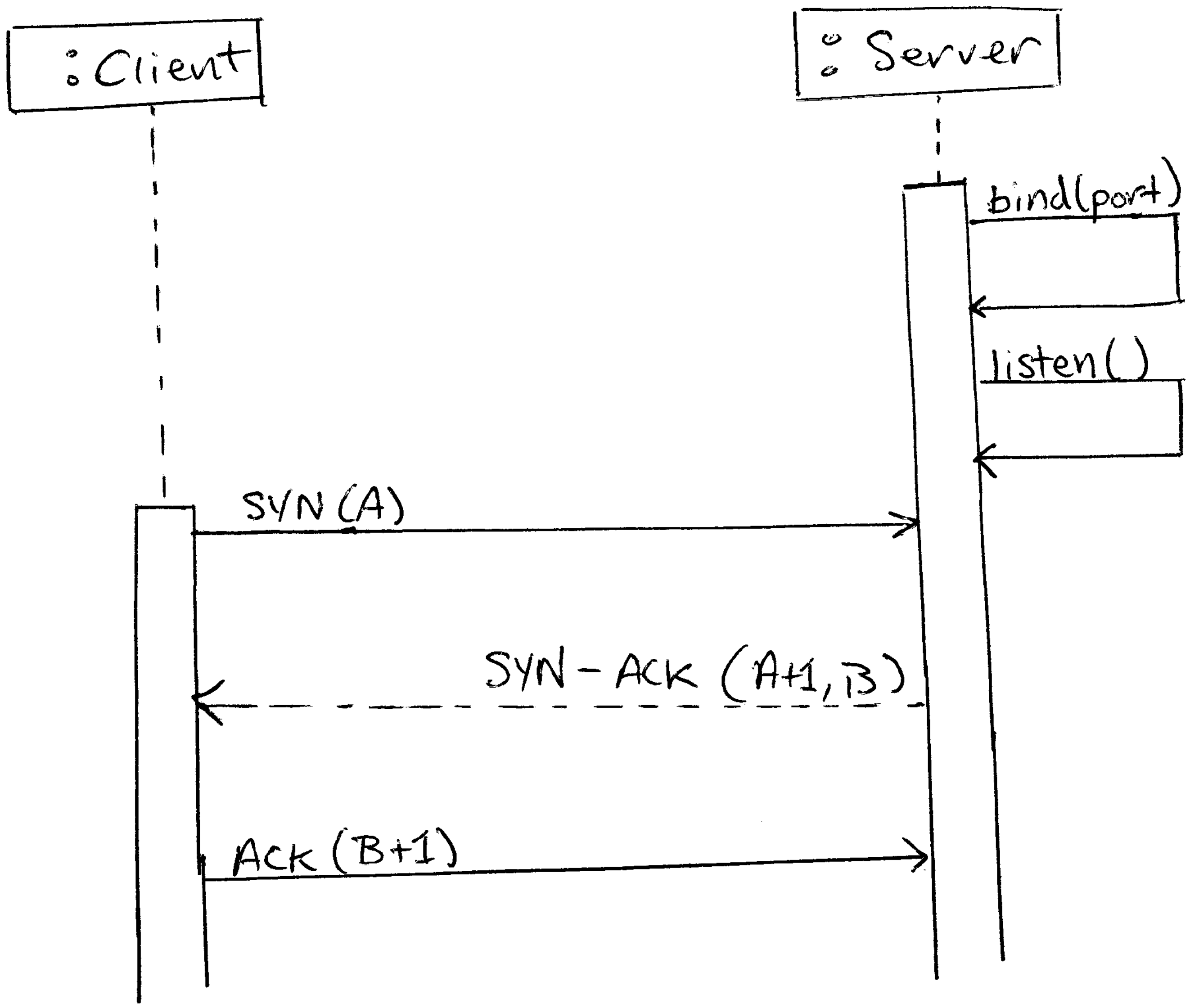
2. SYN-ACK: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B.

3. ACK: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e. B+1.

At this point, both the client and server have received an acknowledgment of the connection, and the connection is established.

On the next page, draw a UML sequence diagram representing the TCP connection establishment process outlined above.

[answer question 1(e) here]



(f) [Use Case Diagram – 20 marks] An insurance brokerage has contracted you to build an application allowing clients to apply for life insurance online. The brokerage sells policies on behalf of the life insurance company, ACME Life. A detailed use case for the brokerage software is given below:

Apply for Life Insurance

Main Success Scenario:

1. customer selects desired insurance plan
2. system determines if client qualifies for selected plan
3. customer fills in the online application form
4. customer submits the application
5. validator checks that all fields are filled & data is valid
6. payment processor authorizes transaction with bank
7. payment processor performs transaction with bank
8. system saves the approved application
9. system notifies client & ACME Life of the new coverage

Extensions:

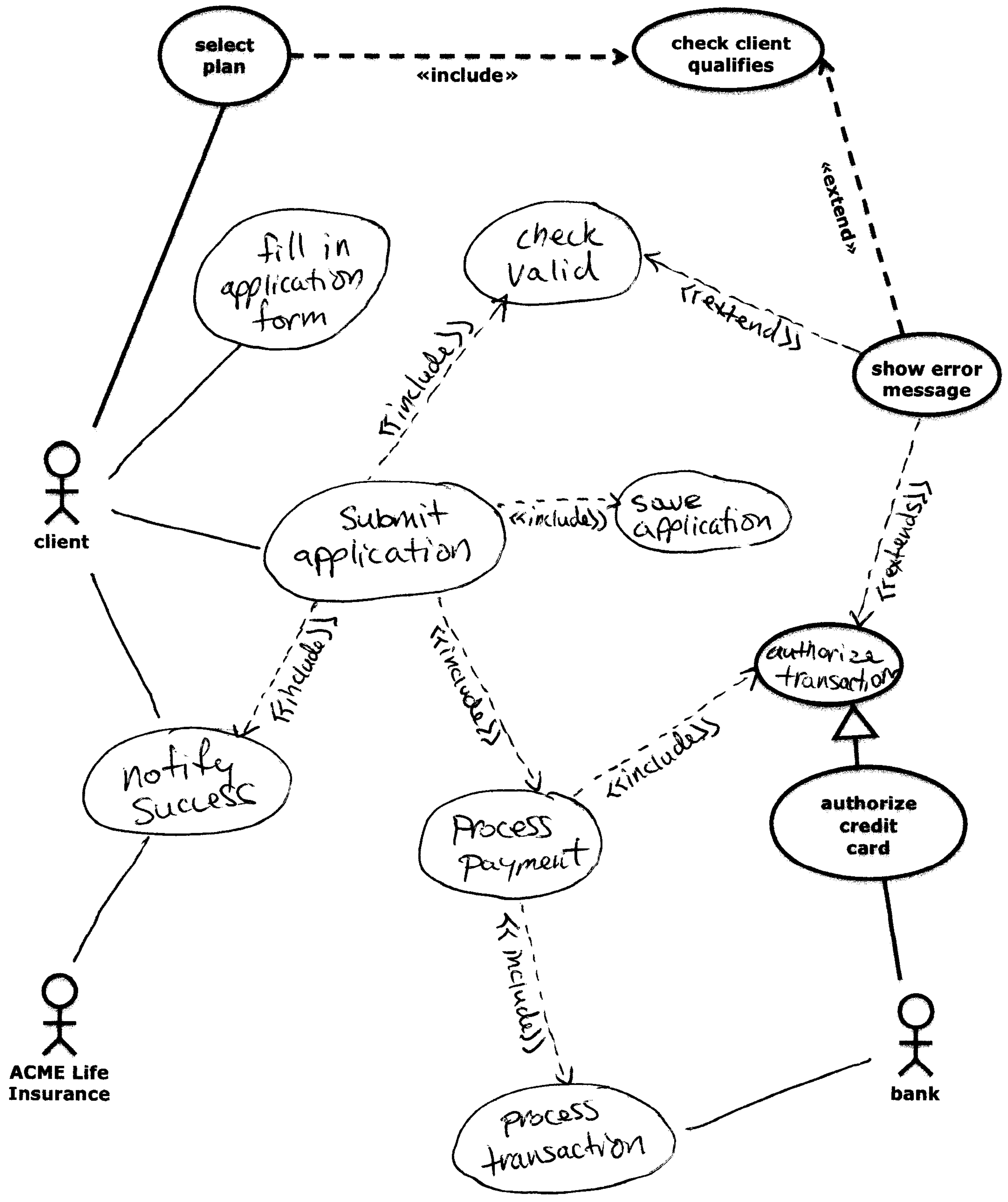
2a. client does not qualify, show error message
5a. data missing or invalid, show error message
6a. payment authorization failed, show error message

Complete the use case diagram for this detailed use case, which is already started on the next page.

Some hints and suggestions:

- use the scratch paper at the back of this package to practice laying out your diagram so you don't make a mess
- you shouldn't have more than a dozen (12) use cases in the diagram in total – write them down before drawing
- assume this brokerage only allows credit cards for payment
- the "submit" use case will have the most **«include»** relationships, and in general is the most interesting

[answer question 1(f) here]



2. [Software Architecture, 15 marks Total]
(a) [Coupling & Cohesion – 10 marks] Recall from the lectures that a good software architecture minimizes **coupling** between modules, and maximizes **cohesion** within modules. Explain what each means and why they are important.

COUPLING: MINIMIZING COUPLING MEANS MODULES DON'T KNOW MUCH ABOUT EACH OTHER. THIS IS IMPORTANT BECAUSE IT MAKES CHANGING THE IMPLEMENTATION EASIER. ALSO, FORCES GOOD API DESIGN.

COHESION: MAXIMIZING COHESION IN MODULES MEANS THAT THEIR CONTENTS (CLASSES, INTERFACES, FUNCTIONS, SOURCE FILES, ETC.) ARE STRONGLY RELATED AND BELONG TOGETHER. THIS IS IMPORTANT TO DETERMINE & MAKE CLEAR WHAT PARTS OF ~~THE~~ THE CODE SERVE WHAT PURPOSE.

(b) [2 & 3-Layer Architectures – 5 marks] A 2-layer software architecture is usually referred to as client-server. A 3-layer architecture is similar to a 2-layer architecture, but the application (or business) logic is moved to a separate layer. What are some benefits of doing this? (Hint: think of your answer to the question above)

MAKES PRESENTATION-LAYER & SERVER-LAYER BOTH UNI-PURPOSE & MORE MAINTAINABLE. PUTS THE BUSINESS LOGIC ALL IN ONE COHESIVE LAYER. CAN CHANGE VIEWS (CLIENT) OR PERSISTENCE (SERVER) WITHOUT AFFECTING BUSINESS LOGIC. MINIMIZES COUPLING BY NOT SPREADING BUSINESS LOGIC ACROSS BOTH CLIENT & SERVER.

3. [Risk Analysis, 10 marks Total]

(a) [Calculating RRL – 5 marks] Risk Management is about assessment and control of risks. In lecture, we discussed assessing risk by calculating Risk Exposure (RE) and calculating the Risk Reduction Leverage (RRL) of mitigating actions to help with control. The formulas for RE and RRL are given below:

RE = probability of unsatisfactory outcome × associated **cost**

RRL = (RE$_{before}$ – RE$_{after}$) ÷ cost of mitigating action

Suppose you have a software system that that costs $100,000 to patch a bug found after release, and also suppose that there is a 10% chance you will have to patch it. We calculate **RE$_{before}$** as:

$$RE_{before} = 10\% \times \$100{,}000 = \$10{,}000$$

You also know that 60% of the bugs are race conditions and 40% are memory leaks. The software to detect (and allow you to fix) memory leaks costs $2,000, and the software to detect possible race conditions costs $3,000. We calculate the RRL for buying the memory leak software as follows:

$$RE_{after} = 10\% \times 60\% \times \$100{,}000 = \$6{,}000$$

$$RRL_{memory\_leak\_detector} = (\$10{,}000 - \$6{,}000) \div \$2{,}000 = 2$$

Perform the same calculation for buying the software that detects race conditions:

RACE CONDITION DETECTOR FINDS ALL RACE CONDITIONS WHICH ACCOUNT FOR 60% OF DEFECTS, SO 40% REMAIN AFTER.

$$RE_{after} = 10\% \times 40\% \times \$100{,}000 = \$4{,}000$$

$$RRL_{race\_cond\_detector} = \frac{(\$10{,}000 - \$4{,}000)}{\$3{,}000}$$

$$= 2$$

SAME AS MEMORY-LEAK DETECTOR!

(b) [Using RRL to make decisions – 2 marks] Using the results of the previous question, and assuming you have $4,000 to spend, which software should you buy?

WE CAN ONLY AFFORD ONE. COULD BUY CHEAPER ONE TO SAVE MONEY, OR THE MORE EXPENSIVE ONE WHICH DETECTS MORE BUGS. EITHER ARGUMENT IS GOOD. BUT MOST IMPORTANTLY, WE CAN'T USE RRL TO DECIDE.

What if your boss increased your budget to $5,000? Then which should you buy?

BUY BOTH FOR SURE!
THEY BOTH HAVE GOOD RRL.

(c) [Interpreting different RRL values – 3 marks] RRL can be seen as a return on investment (ROI). RRL tells you how effective your dollars spent on a particular mitigating action are at lowering your potential exposure cost. Given the ROI view, how would you interpret the following?

- **RRL > 1**

  GOOD ROI. DO IT IF YOU HAVE THE MONEY.

- **RRL = 1**

The reduction in risk exposure equals the cost of the mitigating action. Maybe not bother with this mitigating action, just pay the cost to fix if it happens. If injury or loss of life is possible, we should probably do it.

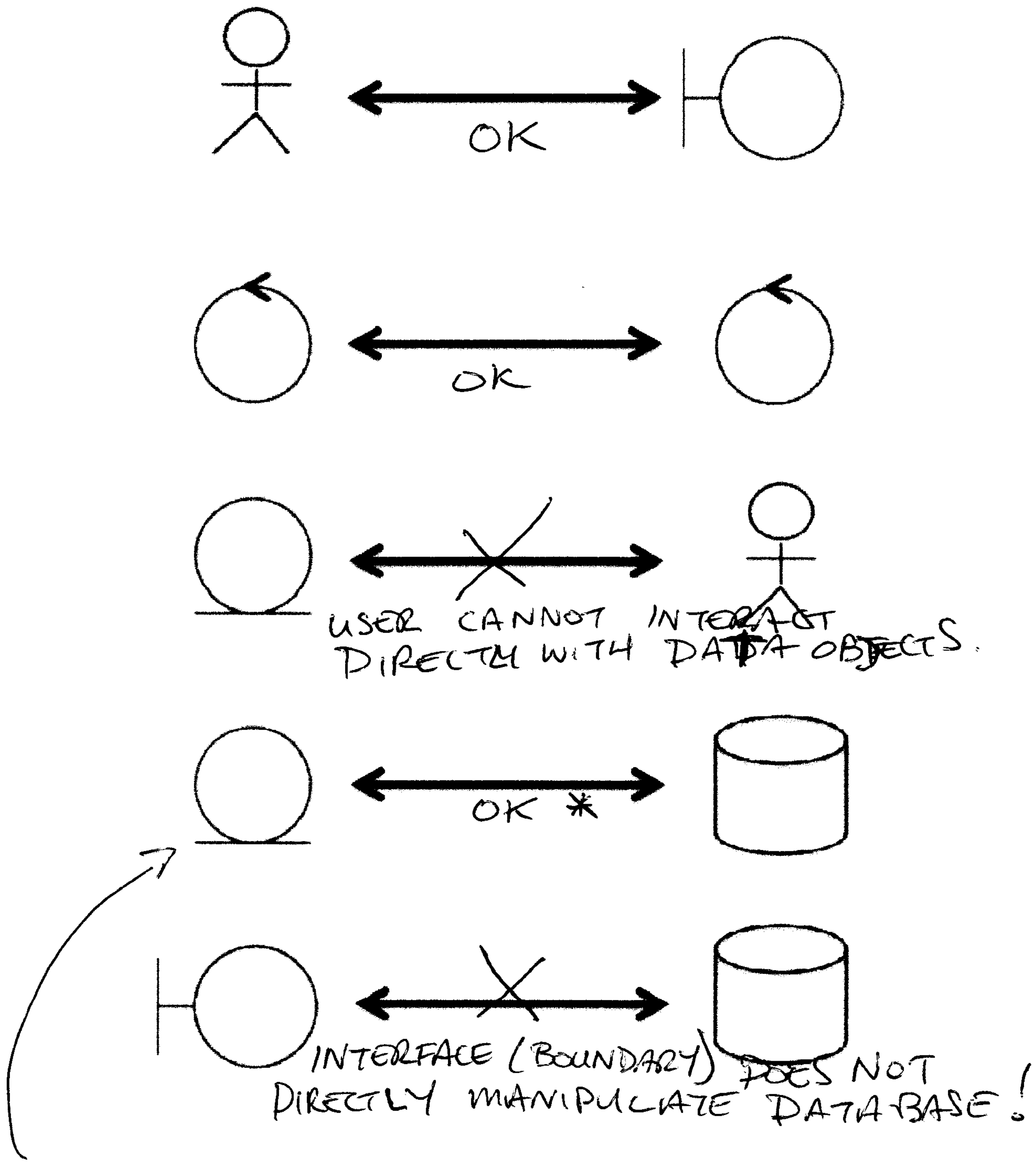
- **RRL between 0 and 1**

  COSTS MORE THAN YOU SAVE. STILL IMPROVES SITUATION, BUT AT HIGH COST. MAYBE ONLY DO IT IF INJURY OR LOSS OF LIFE IS POSSIBLE.

- **RRL < 0**

  MITIGATING ACTION ACTUALLY INCREASED OUR RISK! NEVER DO THIS ONE.

4. [Robustness Diagrams, 10 marks Total]
Indicate which interactions are **NOT** allowed in robustness diagrams, and state why not.



USER CANNOT INTERACT DIRECTLY WITH DATA OBJECTS.

INTERFACE (BOUNDARY) DOES NOT DIRECTLY MANIPULATE DATABASE!

* IF YOU SAID THAT ENTITIES MUST GO THROUGH CONTROL OBJECTS TO PERSIST I'D BE OK WITH GIVING THAT FULL MARKS.

[scratch paper]

[scratch paper]

[scratch paper]