Concordia University

Department of Computer Science and Software Engineering

SOEN 331 - S and U

Formal Methods for Software Engineering

Assignment 2

The Object-Z specification language

Dr. Constantinos Constantinides, P.Eng.

constantinos.constantinides@concordia.ca

February 18, 2021

1 General information

Date posted: Thursday 18 February, 2021.

Date due: Thursday, 11 March, 2021, by 23:59.

Weight: 25% of the overall mark.

2 Introduction

This is a team assignment. Each team should designate a leader who will submit the assignment electronically. There are **3** parts in this assignment, with a total of **50** points. You must prepare all your solutions in LaTeX and produce a single `pdf` file. Please make sure you include all names and id's of all contributing team members as the authors. Name the file after your team, e.g. `team1.pdf`.

2.1 Notation

Proper notation is crucial. For example, a different symbol should be used for a total function, than that for a partial function. Also, different symbols exists for injective, surjective or bijective functions. Once you decide on the type of the function, you then must look up and locate the appropriate symbol. A template is provided on the course website to help you prepare your specification in Object-Z. The template makes use of a LaTeXpackage called `oz`. Together with package `oz`, you will also need to refer to "The Comprehensive LaTeXSymbol List." Both a reference to `oz` and "The Comprehensive.." document are available from the course website.

3 Ground rules

This is an assessment exercise. You may not seek any assistance while expecting to receive credit. **You must work strictly within your team and seek no assistance for this project (from the instructor, the teaching assistants, fellow classmates and other**

teams or external help). Please note that you should **not** discuss the assignment during tutorials. Failure to do so will result in penalties or no credit.

All team members are expected to work relatively equally on each problem. The team leader has the responsibility to ensure that the team does not violate this rule. Failure to do so will result in penalties. In your submission, you must include only the names of those people who contributed to the assignment. Accommodating someone who did not contribute will result in penalties.

If there is any problem in the team (such as lack of contribution, etc.), the team leader must contact the instructor as soon as the problem appears.

4 Type system

We introduce the basic types [*PhoneNumberType*, *NameType*]. We also introduce an enumerated type *Message* which will assume values that correspond to success and error messages.

Consider a system where you maintain your contacts. Let us consider an unbounded contacts list. Each person in your contacts list has a unique first- and last name combination (for simplicity, let us call it 'name'), and is marked with one or possibly more phone numbers. People may not share phone numbers.

5 Your assignment

5.1 State (7 pts)

The declaration of the state of the system is defined by

- The set of phone numbers (call it *numbers*) that are recorded in contacts.
- A record of association between names and phone numbers, given by a correspondence (call it *recorded*).

1. Provide a diagram to visualize the state of the system.
2. Provide a formal definition for *numbers*.
3. Does *recorded* have to be captured by a function? What requirements would a function enforce? Explain in detail.
4. What is the domain and the codomain of *recorded*?
5. What type of function should *recorded* be (full or partial)? Explain in detail.
6. Will *recorded* be an injective, surjective, or bijective? Explain in detail.
7. Provide a formal definition for *recorded*.

5.2 Class `Contacts` (35 pts)

Define a formal specification in Object-Z for class *Contacts* whose interface contains the following *robust specifications*:

- `MakeNewContact`: Adds a new person to Contacts with a single phone number.
- `AddNumber`: Adds an additional phone number for an existing contact.
- `SearchForNumber`: Returns a collection of phone numbers for a given person.
- `DeleteNumber`: Deletes an existing number.

5.3 Class `Contacts2` (8 pts)

Subclassify `Contacts` to introduce class `Contacts2` that behaves exactly like `Contacts`, while introducing a robust operation to search for a person, given a phone number through operation `SearchForPerson`.

6 Guidelines

There are a few things to consider and the following guidelines will help you stay in the right track. They will also help you validate your formal specification.

- Start by creating a Venn diagram of the system. What types do you have?
- What mappings do you have, if any? If there is a mapping of elements from one set to another, does it constitute a function? Why? What type of function is it? Full, or partial? Why?
- Is it possible to represent this function as a set? What elements does it expect to hold?
- Let us investigate the function further: Is it injective, surjective, or bijective? Why?
- What is the invariant property of the system? This may not necessarily be a single expression, but it may be composed by a number of expressions.
- What is the state of the system? What variables do you have and what are their corresponding types?
- What is the system initial state? Have you considered all variables that must be present in the initial state? Does the invariant hold upon instantiation of the system?
- What operations do you expect to have in the system? Which operations form part of the system's exposed interface? Which ones are utility/auxiliary operations (i.e. not part of the interface)?
- For each operation, start by describing the associated assertions (preconditions and postconditions) in plain English. In order to allow this operation to proceed, what

expectations are imposed on the client? What expectations are then imposed on the provider in order to consider the operation successful? Once you have high-level descriptions, translate them in formal expressions.

- Provide some sample data in order to simulate the behavior of your system while going through various scenarios. Does the invariant hold once the state has been initialized? Does the invariant hold before an operation proceeds and also upon successful termination? Does the postcondition hold upon successful termination? If a precondition does not hold, what does the system do? Does it ignore a precondition failure, or does it provide robustness (i.e. handle abnormal situations)? Is there any scenario under which your formal specification can fail?

- Overall, does your formal system faithfully model its requirements and is it reliable? (reliability = correctness + robustness)

7 What to submit

Please submit your `pdf` file at the Electronic Assignment Submission portal

(`https://fis.encs.concordia.ca/eas`)

under **Theory Assignment 2**.

END OF ASSIGNMENT
