October 31, 2014.

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México
Escuela de Diseño, Ingeniería y Arquitectura
Departamento de Tecnologías de Información y Computación

Software Design and Architecture Second Exam

Instructor: Ariel Ortiz Course Number and Section: Tc3049.1

Name: ______________________________ Student ID: ____________

Apegándome al Código de Ética de los Estudiantes del Tecnológico de Monterrey, me comprometo a que mi actuación en este examen esté regida por la honestidad académica. En congruencia con el compromiso adquirido con dicho código, realizaré este examen de forma honesta y personal, para reflejar, a través de él, mi conocimiento y aceptar, posteriormente, la evaluación obtenida.

Firma: ____________________

VERY IMPORTANT: During the exam you may not be connected to the network, use cellphone, or use anyone else's material. Any evidence of cheating or fraud will be punished with a DA (Academic Dishonesty) grade. This punishment is for both the person who copies and for the person who allows to be copied.

General Instructions

Rename the `exam2.rb` file to `A0MMMMMMM.rb`, where `A0MMMMMMM` is your student ID. **Type your name and student ID in a comment at the top of this Ruby source file.**

The source file contains a series of unit tests that you can use to test your solutions. **Do not modify the code for these tests.**

Once you have finished the exam, copy the source file to the removable USB memory drive provided by the instructor.

Problems

The provided class `LinkedList` represents a sequential collection of elements implemented as a circular doubly-linked list. You are not allowed to modify this class in any way. The following problems require you to extend the functionality of this class by using mechanisms such as delegation.

1. **(40%) Adapter Pattern.** Write a class called `StackAdapter` that allows adapting a `LinkedList` so that it can be used as LIFO (*Last-In, First-Out*) stack. The adapter should contain these methods:

   - `initialize(a)`: Initializes a stack, where `a` is the adaptee list.
   - `push(e)`: Pushes `e` into the top of this stack (start of the adaptee list).
   - `pop`: Pops an element from the top of this stack (start of the adaptee list). Raises a `RuntimeError` if the stack is empty.
   - `empty?`: Returns `true` if this stack is empty, otherwise returns false.

   See the corresponding unit test for more details and examples.

2. **(30%) Iterator Pattern.** Add an instance method called `iterator()` to the `StackAdapter` class from the previous problem. This method should return an instance of the `Enumerator` class in order to enable internal and external iteration over all the elements of the corresponding stack object. More specific details and examples are contained in the given unit test.

3. **(30%) Decorator Pattern.** Write a class called `SizeListDecorator` that allows decorating a `LinkedList`. This class should define a method called `size()` that returns the number of elements contained in the decorated object. `SizeListDecorator` should not be a subclass of `LinkedList`. The `SizeListDecorator` class should delegate the implementation of the methods `empty?()`, `add_first()`, `add_last()`, `remove_first()`, `remove_last()`, `each()`, and `to_s()` to the decorated instance of `LinkedList`. See the corresponding unit test for more details and examples.