EECS3311 Fall 2017
Lab Exercise 3
Implementing an Iterable Dictionary ADT

Chen-Wei Wang

Due Date: 12:00 Noon, Monday, October 30

Check the Amendments section of this document regularly for changes, fixes, and clarifications.

1 Policies

- **Your (submitted or unsubmitted) solution to this lab exercise (which is not revealed to the public) remains the property of the EECS department. Do not distribute or share your code in any public media (e.g., github) in any way, shape, or form. The department reserves the right to take necessary actions upon found violations of this policy.**
- You are required to **work on your own** for this lab. **No** group partners are allowed.
- When you submit your lab, you claim that it is **solely** your work. Therefore, it is considered as **an violation of academic integrity** if you copy or share **any** parts of your Java code during **any** stages of your development.
- When assessing your submission, the instructor and TA will examine your code, and suspicious submissions will be reported to the department if necessary. **We do not tolerate academic dishonesty**, so please obey this policy strictly.
- You are entirely responsible for making your submission in time. Back up your work **periodically**, so as to minimize the damage should any sort of computer failures occur.
- The deadline is **strict** with no excuses: you receive **0** for not making your submission in time.
- You are free to work on this lab anywhere, but you are advised to test your code via your EECS account before the submission.

2 Learning Outcomes of this Lab Exercise

1. Apply the Iterator Pattern.
2. Define a Generic Class.
3. Write unit tests to verify the correctness of your software.
4. Draw professional architectural diagram using the draw.io tool.

3 Required Readings

- Iterator Pattern:
  - Architecture
  - Lecture notes (Watch the corresponding lecture recording if necessary)
  - Sample codes
  - Tutorial Videos on "Generic Parameters and the Iterator Pattern" and " Information Hiding and the Iterator Pattern".
- Generic Parameter:
  - Lecture notes (Watch the corresponding lecture recording if necessary)
  - Tutorial videos (see the above reference)
- You can also find an abundance of resources on DbC, TDD, ESpec tests, and Eiffel code examples from these two sites:
  - `http://eiffel.eecs.yorku.ca`
  - `https://wiki.eecs.yorku.ca/project/eiffel/`

Contents

4 Problem

The dictionary ADT (Abstract Data Type) supports the storage and access of a collection of entries. Each entry is characterized by a search key and its associated value. It is necessary that keys are not duplicated, so that each key can uniquely identify an associated value. On the other hand, different keys may map to different values. You can think of a dictionary as a mathematical function, not a relation.

4.1 Programming

4.1.1 Getting Started

- Download the starter project archive `Lab_3.zip` from the course moodle, unzip it, and launch EStudio.
- Choose `Add Project`, browse to the project configuration file `Lab_3/dictionary/dictionary.ecf`, then select `Open`.
- It is expected that the starter project does not compile. This is because there are missing classes, inheritance declarations, and feature definitions.
- Study carefully the `INSTRUCTOR_DICTIONARY_TESTS` class, where test cases are provided to illustrate how the various features are expected for you to implement.
- This is the suggested workflow for you:
  - You should start by adding the necessary classes and feature declarations in order to at least make the entire project compile. You receive 0 marks if your submitted project does not compile.

    Any implementation classes that you add must be created under the `model` cluster.
  - Then, implement those features so that the provided tests pass.
  - Then, add your own tests until you are satisfied before submission.

    Any test classes that you add must be created under the `tests/student` cluster.

4.1.2 Tasks

- A generic class `DICTIONARY[V, K]` is already declared for you, where `V` and `K` denotes types of, respectively, the values and their search keys.
- Notice that in Eiffel collection classes such as `ARRAY`, `LINKED_LIST`, and `TUPLE`:
  - There is a Boolean attribute `object_comparison`, which indicates if items in the collections are to be compared using reference equality (i.e., via `=` when `object_comparison` is `false`) or object equality (i.e., via the user-redefined `is_equal` feature when `object_comparison` is `true`).
  - By default, the `object_comparison` attribute is `false`. If you mean it to be `true`, you can call the command `compare_objects`.
  - See the `test_array_comarison` feature in class `INSTRUCTOR_DICTIONARY_TESTS` for an example.
- For this lab exercise, you are forced to implement the dictionary ADT via a naive solution using two linear data structures: an array of values and a linked list of keys. A value (from the values array) and a key (from the keys list), given that both are at the same index, make an entry in the dictionary.
- You are forbidden to use any other library class such as `HASH_TABLE` to implement the dictionary. Such a constraint is imposed to make you practice the "unlucky" case of implementing the Iterator Pattern.
- You are required to make the `DICTIONARY` class *iterable*.
- Then, identify all `-- Your Task` comments which indicate what you should complete.
- Comments within the `DICTIONARY` class, together with the test queries in `INSTRUCTOR_DICTIONARY_TESTS`, should provide you with information to complete all the programming tasks.

4.2 BON Architecture Diagram

– Use draw.io to draw a BON Architecture Diagram that details **all** classes in the `model` cluster (i.e., supplier classes) and the `INSTRUCTOR_DICTIONARY_TESTS` class (i.e., the client class).

You do **not** need to show all features. We leave it to your judgement to summarize the critical classes/features/contracts for others to understand the design of the project in a single page.

– When completing your drawing, also export the diagram into a PDF.

– Move the source of your draw.io drawing (name it `dictionary.xml`) and its exported PDF (name it `dictionary.pdf`) to the `docs` directory of your project.

4.3 Report

– Compile (into a single PDF file named `Report.pdf`) including:

- A cover page that clearly indicates: 1) course (`EECS3311`); 2) semester (`Fall 2017`); 3) name; and 4) CSE login;
- **Section "Contract View"**:

  In this section, present the **contract view** of the *DICTIONARY* classes. You may just copy and paste from the `contract view` in EStudio, but make sure that the presentation in this section is well-formatted.
- **Section "Architectural Diagram"**:
  - ◇ Insert the BON diagram `dictionary.pdf` (exported from `draw.io`) for your design here.
  - ◇ Explain how the Iterator Pattern is implemented in the `model` cluster.
  - ◇ Explain how you implement the feature `another_cursor` in the `DICTIONARY` class.

5 Submission

5.1 Checklist before Submission

1. Complete all contracts and implementations indicated with "`-- Your task`" in the starter code.
   - Do **not** modify any tags of the contracts. Do **not** modify signatures (names and types) of existing features. You **may** add auxiliary features to existing classes if necessary.
   - You should exercise the test-driven development method that was taught.
   - Write and pass as many tests as possible on your software, as only passing the given tests will **not** ensure good quality of your development.
2. Use draw.io to draw a BON Architecture Diagram.
   - When completing your drawing, also export the diagram into a PDF.
   - Move the source of your draw.io drawing (name it `dictionary.xml`) and its exported PDF (name it `dictionary.pdf`) to the `docs` directory of your project.
3. Move the source of your report (`Report.pdf`) into the `docs` directory. Do **not** put any other format of the report (e.g., word).

5.2 Submitting Your Work

To get ready to submit:

- Close EStudio
- Type the following command (only available via your lab account) to clean up the `EIFGENs` directory:

```
cd ~/Desktop/EECS3311_Labs/Lab_3
eclean dictionary
```

- Make sure the directory structure of your project is identical to Fig. **??** (with **no** `EIFGENs`).

  By the due date, submit via the following command:

```
cd ~/Desktop/EECS3311_Labs/Lab_3
submit 3311 lab3 dictionary
```

  A check program will be run on your submission to make sure that you pass the basic checks (e.g., the code compiles, passes the given tests, *etc*). After the check is completed, feedback will be printed on the terminal, or you can type the following command to see your feedback (and later on your marks):

```
feedback 3311 lab3
```

  In case the check feedback tells you that your submitted project has errors, you must fix them and re-submit. Therefore, you may submit for as many times as you want before the submission deadline, to at least make sure that you pass all basic checks.

  Note. You will receive zero for submitting a project that cannot be compiled.

6 Amendments