Collections & File I/O

Lab 6

Announcement

- There will be a Lab Test next week!
- Please check the “Notes on Midterm Exam” post on ETL.

Objectives

- Get accustomed to use Java collections.
- Get accustomed to read/write a file.
- Experience designing your own programming.

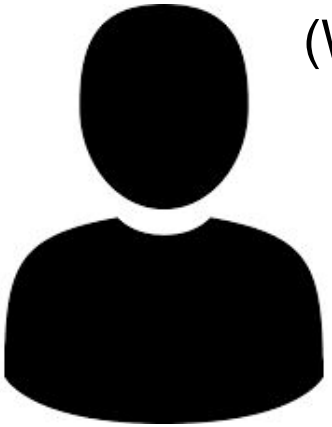
Java Collections Framework

- Let’s try out some interfaces and classes in Java collections framework. (Example codes will be uploaded.)

File I/O (Input/Output)

- Until the last lab session, the data of the programs were gone when we exit the program.
- One way to solve this problem is to save data as a file.
  - (Alternative: Database)
- What functions do we need?
  - Write something to a file. ⇒ **FileWriter**
  - Read something from a file. ⇒ **Scanner** & **File**
  - ...

Today's Work! - Diary Application



(Write/List/Read/Delete)

Command/Input

DiaryUI

Output

Other classes if you need

Diary

Write/List/Read/Delete Entries

DiaryEntry

id, creation time, title, content

Diary Application

- Let’s make a diary application with collections.
- A user should be able to write/list/read/delete diary entries.
- DiaryUI class is already implemented to get user command/input, or print messages. DiaryUI handle some wrong inputs. Don’t care other exceptions.
- Implement createEntry, listEntries, readEntry, and deleteEntry methods of Diary class.

Output - Create Entries

```
Type a command
   create: Create a diary entry
   list: List diary entries
   read <id>: Read a diary entry with <id>
   delete <id>: Delete a diary entry with <id>
   search <keyword>: List diary entries whose contents
contain <keyword>
Command: create
   title: First Entry
   content: Dear Diary, Life is beautiful.
   The entry is saved.
```

Output - List Entries

```
Type a command
   create: Create a diary entry
   list: List diary entries
   read <id>: Read a diary entry with <id>
   delete <id>: Delete a diary entry with <id>
   search <keyword>: List diary entries whose contents contain
<keyword>
Command: list
   id: 3, created at: 2019/10/22 11:55:30, title: Third Entry
   id: 2, created at: 2019/10/22 11:48:30, title: Self-reflection
   id: 1, created at: 2019/10/22 11:47:28, title: First Entry
```

Output - Read Entries

```
Type a command
   create: Create a diary entry
   list: List diary entries
   read <id>: Read a diary entry with <id>
   delete <id>: Delete a diary entry with <id>
   search <keyword>: List diary entries whose contents
contain <keyword>
Command: read 5
   id: 5
   created at: 2019/10/22 11:47:28
   title: First Entry
   content: Dear Diary, Life is beautiful.
```

Output - Delete Entries

```
Type a command
   (...)
Command: delete 1
   Entry 1 is removed.

Type a command
   (...)
Command: list
   id: 3, created at: 2019/10/22 11:55:30, title: Third Entry
   id: 2, created at: 2019/10/22 11:48:30, title: Self-reflection
```

Upgrade Diary - Search Engine

- Let's make a search engine for your diary application.
- The user should be able to search entries which contain a given keyword in their contents.
  - ex) content: I will be a great engineer.
    keyword: great ⇒ I will be a great engineer (O)
    keyword: engine ⇒ I will be a great engineer (X)
  - HINT: use split(" ") method on entry contents.
- Which collection should you use?
  - List vs Set vs Map ?

Output - Search Entries

```
Type a command
   ...
   search <keyword>: List diary entries whose contents
contain <keyword>
Command: search I
   id: 3
   created at: Third Entry
   title: 2019/10/22 11:55:30
   content: I want to become a great engineer!
   id: 2
   created at: Self-reflection
   title: 2019/10/22 11:48:30
   content: Dear diary, Why am I studying computer science?
```

Upgrade Diary - Store Entries

- If you turn off your diary application, the entries will be erased.
- ⇒ Save entries with file I/O.
  - Create a “data” directory.
  - Save entries as files in “data” directory.
  - Name the files as “01.txt”, “02.txt”, …
  - Naming the files as “1.txt”, “2.txt”, ... may result in “10.txt” coming before “2.txt”.
- Also you need to load the entries from the storage when you turn on the diary program.

Discuss with Your Partner

- Discuss the overall design and issues with your partner before you write down the codes.
  - Which class should do which job?
  - Which variables should be defined in classes?
  - Which helper methods should be implemented?
  - How to make the searching engine work well even after the user shutdown and restart the diary program?
- If you and your partner finished, you can leave.
- TA will start implementation at 7:30.
- Ask any questions to TAs!
- The source code will be uploaded on ETL after the class.