EECS2030 Fall 2016
Preparation Exercise for Lab Test 2:
A Birthday Book

Chen-Wei Wang

Contents

Objectives of this Exercise

- Given requirements (descriptions written in English) and expected uses of software (i.e., tester classes), declare and define classes and methods accordingly.
- Write an application (some kind of tester) which interacts with user inputs and manipulate objects.
- Understanding the behaviours of nested loops, by means of the complete tracing of a given input value.
- Using the combination of iterations (for-loops or while-loops), selections (if-statements), and the data structure of one-dimensional arrays to solve problems.
- Using breakpoints and debugger in the Eclipse IDE to help find errors in programs with sophisticated control structures (e.g., nested and/or series of loop and/or if-statements) and data structures (e.g., several arrays).
- Exercising a common pattern of interaction: reading inputs (numbers or strings) from users, performing calculations, and displaying results on the console.

Required Readings

- Slides on classes and objects
- Watch the tutorial video here and study the slides here on *Encapsulation in Java*.

1 Before Getting Started

- You are **only** allowed to do so using Java arrays. The use of any Java library classes such as `ArrayList`, `LinkedList`, `HashMap`, *etc.*, is **forbidden** in this exercise.
- For each problem, start by sketching your solution (not necessarily in valid Java syntax) on a piece of paper, as if it were a paper test.
- When your program does not behave as you expected (which is very likely!) on certain input values, set breakpoints at the beginning of the `main` method, or at critical points of your program, and then run the debugger to observe closely the changes on *Variables* and *Expressions* line by line.
- For each of the tasks below, study the expected runs (where user inputs are marked in red) that we give to you carefully, and make sure that your program outputs in the same manner.
- However, the input values that we give in these expected runs are just examples. You should test your program with various input values to convince yourself that your program is indeed correct.

2 Task: Implementing Classes for Birthdays, Entries, and Books

A birthday book stores a collection of entries, where each entry is a pair of a person's name and their birthday. No two entries stored in the book are allowed to have the same name. Each birthday is characterized by a month and a day. A birthday book is first created to contain an empty collection of entires, and may store *up to 10* entries.

Given a birthday book, we may inquire about the number of entries currently stored in the book, add a new entry by supplying its name and the associated birthday, remove the entry associated with a particular person, find the birthday of a particular person, or get a reminder list of names of people who share a given birthday.

You are required to create and define Java classes, attributes, and methods to implement the above (informal) system requirements. Follow these steps to complete this task:

- Study the following `BirthdayBookTester` class (Section 2.3) carefully. It indicates the minimum set of classes and signatures of methods that you need to define in order to make it compile. In addition to what is suggested from this tester class, you are free to declare new classes, attributes, or methods as you find necessary, as long as you satisfy all the requirements as outlined in Section 2.1, and your final developed project remains compilable.
- Create and **type verbatim** the tester class `BirthdayBookTester` (Section 2.3) in your lab test project. You may do this `incrementally`: as soon as you encounter a line that does not compile (because of missing classes or methods), you make the necessary class or method definitions accordingly. To make this tester class **compile**, you must create the indicated classes and methods with the indicated signatures (i.e., names, parameters, and return types).
- Implement the methods that are suggested by the tester class, according to its expected outputs (Section 2.4).

2.1 Requirements for Task

Here are requirements that you must follow **stringently**:

1. **Nowhere** in all methods (e.g., the `toString()` method, which **returns** a string, rather than printing a string to the console) that you define can contain **any** print statements. That is, the only `System.out.println(...)` statements that can be found in your project are in the tester class `BirthdayBookTester` that you are given.
2. Watch the tutorial video and study the slides on *Encapsulation in Java*. **All** attributes declared in your classes must be **private**, meaning that **no** outside classes can access these attributes. Instead, **all** outside classes can only call accessor methods (to gain information) or mutator methods (to change attribute values). For example, the following class

```
1  class A {
2    int i;
3    String s;
4  }
```

is **unacceptable** for this exercise as the two attributes `i` and `s` are not declared as **private**. Here is the expected version:

```
1  class A {
2    /* Only within class A can we access these two attributes directly. */
3    private int i;
4    private String s;
5    /* These accessors and mutators are for outside classes to
6     * access and mutate the two attribute values.
7     */
8    public int getI() { return i; }
9    public void setI(int i) { this.i = i; }
10   public String getS() { return s; }
11   public void setS(String s) { this.s = s; }
12 }
```

3. To implement the collection of entries of a birthday book, you are **only** allowed to do so using Java arrays. The use of any Java library classes such as `ArrayList`, `LinkedList`, `HashMap`, *etc.*, is **forbidden** in this exercise.

2.2 Tips for Task

Each birthday book contains a (possibly empty) array of entries. When removing an entry from the array, you have two options of implementing it:

1. Assign to *null* the corresponding position in the array (i.e., where the birthday book entry to be removed is stored). For example, say there are three entries in the book:

$$\{ \quad (\texttt{"Alan"},\ 1954-6-13), \quad (\texttt{"Mark"},\ 1964-7-14), \quad (\texttt{"Tom"},\ 1944-3-16) \quad \}$$

Then after removing the entry associated with `"Mark"` from the book, we have:

$$\{ \quad (\texttt{"Alan"},\ 1954-6-13), \quad \mathit{null}, \quad (\texttt{"Tom"},\ 1944-3-16) \quad \}$$

The consequence of this first approach is that when later printing the book or scanning through the book for removal, you have to avoid[1] all *null* slots in the middle of the array(s). Similarly, when later adding a new entry to the book, you will need to scan through the array and find the first available *null* slot.

2. Find where the entry to be removed is in the array, then: **1)** shift all its right neighbours to the left by one position; and **2)** assign the *old* right-most right neighbour to *null*. For example, say there are three entries in the book:

$$\{ \quad (\texttt{"Alan"},\ 1954-6-13), \quad (\texttt{"Mark"},\ 1964-7-14), \quad (\texttt{"Tom"},\ 1944-3-16) \quad \}$$

Then after removing Mark from the book, we have:

$$\{ \quad (\texttt{"Alan"},\ 1954-6-13), \quad (\texttt{"Tom"},\ 1944-3-16), \quad \mathit{null} \quad \}$$

Notice that before the shift, the right-most right neighbour in the above example (i.e., Tom) is located at index 2. After the shift, index 2 points to *null*, meaning that particular slot is available for a new entry to be stored. Observe that this second approach, unlike the first approach, does not result in any *null* slots in the middle of the array, but only at the end.

[1] Otherwise, you will have `NullPointerException` at runtime.

2.3 BirthdayBookTester.java

```
1  public class BirthdayBookTester {
2    public static void main(String[] args) {
3      System.out.println("(01)----------------------------------------");
4      /* Create a birthday instance with month and day. */
5      Birthday bd01 = new Birthday(1, 11);
6      System.out.println("(" + bd01.getMonth() + ", " + bd01.getDay() + ")");
7
8      System.out.println("(02)----------------------------------------");
9      Birthday bd02 = new Birthday(2, 12);
10     Birthday bd03 = new Birthday(3, 13);
11     Birthday bd04 = new Birthday(4, 14);
12     Birthday bd05 = new Birthday(5, 15);
13     Birthday bd06 = new Birthday(6, 16);
14     Birthday bd07 = new Birthday(7, 17);
15     Birthday bd08 = new Birthday(8, 18);
16     Birthday bd09 = new Birthday(9, 19);
17     Birthday bd10 = new Birthday(10, 20);
18     Birthday bd11 = new Birthday(11, 21);
19     Birthday bd12 = new Birthday(12, 22);
20     System.out.println(bd01.toString());
21     System.out.println(bd02.toString());
22     System.out.println(bd03.toString());
23     System.out.println(bd04.toString());
24     System.out.println(bd05.toString());
25     System.out.println(bd06.toString());
26     System.out.println(bd07.toString());
27     System.out.println(bd08.toString());
28     System.out.println(bd09.toString());
29     System.out.println(bd10.toString());
30     System.out.println(bd11.toString());
31     System.out.println(bd12.toString());
32
33     System.out.println("(03)----------------------------------------");
34     Birthday bd13 = new Birthday(6, 16);
35     System.out.println("bd06: " + bd06.toString());
36     System.out.println("bd07: " + bd07.toString());
37     System.out.println("bd13: " + bd13.toString());
38     System.out.println("Contents of bd06 and bd13 are equal: " + bd06.equals(bd13));
39     System.out.println("Contents of bd07 and bd13 are equal: " + bd07.equals(bd13));
40
41     System.out.println("(04)----------------------------------------");
42     /* Create a new entry using a name and a birthday. */
43     Entry e1 = new Entry("A", bd01);
44     System.out.println("Entry e1's name: " + e1.getName());
45     System.out.println("Entry e1's birthday: " + e1.getBirthday().toString());
46     System.out.println("Entry e1's string value: " + e1.toString());
47
48     System.out.println("(05)----------------------------------------");
49     /* Create a new entry using a name, a birth month, and a birth day. */
50     Entry e2 = new Entry("B", 2, 12);
51     System.out.println("Entry e2's name: " + e2.getName());
52     System.out.println("Entry e2's birthday: " + e2.getBirthday().toString());
53     System.out.println("Entry e2's string value: " + e2.toString());
54
```

```
55      System.out.println("(06)----------------------------------------");
56      Entry e3 = new Entry("A", 1, 11);
57      System.out.println("e1: " + e1.toString());
58      System.out.println("e2: " + e2.toString());
59      System.out.println("e3: " + e3.toString());
60      System.out.println("Entries e1 and e2 are equal: " + e1.equals(e2));
61      System.out.println("Entries e1 and e3 are equal: " + e1.equals(e3));
62
63      System.out.println("(07)----------------------------------------");
64      e2.setName("A");
65      /* Change e2's birthday to January 11. */
66      e2.setBirthday(1, 11);
67      System.out.println("e1: " + e1.toString());
68      System.out.println("e2: " + e2.toString());
69      System.out.println("e3: " + e3.toString());
70      System.out.println("Entries e1 and e2 are equal: " + e1.equals(e2));
71      System.out.println("Entries e1 and e3 are equal: " + e1.equals(e3));
72
73      System.out.println("(08)----------------------------------------");
74      /* Change e3's birthday to the same as bd03. */
75      e3.setBirthday(bd03);
76      System.out.println("e1: " + e1.toString());
77      System.out.println("e2: " + e2.toString());
78      System.out.println("e3: " + e3.toString());
79      System.out.println("Entries e1 and e2 are equal: " + e1.equals(e2));
80      System.out.println("Entries e1 and e3 are equal: " + e1.equals(e3));
81
82      System.out.println("(09)----------------------------------------");
83      BirthdayBook bb = new BirthdayBook();
84      System.out.println("Number of entries: " + bb.getNumberOfEntries());
85      System.out.println("Returned number of entries: " + bb.getEntries().length);
86
87      System.out.println("(10)----------------------------------------");
88      System.out.println(bb.toString());
89
90      System.out.println("(11)----------------------------------------");
91      System.out.println("Name A exists in book: " + bb.nameExists("A"));
92      System.out.println("Name B exists in book: " + bb.nameExists("B"));
93      System.out.println("Name C exists in book: " + bb.nameExists("C"));
94
95      System.out.println("(12)----------------------------------------");
96      Birthday bdOfA = bb.getBirthday("A");
97      Birthday bdOfB = bb.getBirthday("B");
98      Birthday bdOfC = bb.getBirthday("C");
99      /* Return birthdays on names.
100      * When names are non-existing, their associated birthdays are nulls.
101      */
102     System.out.println("Birthday of A: " + bdOfA);
103     System.out.println("Birthday of B: " + bdOfB);
104     System.out.println("Birthday of C: " + bdOfC);
105
106     System.out.println("(13)----------------------------------------");
107     /* Get reminders on birthdays.
108      * When birthdays are non-existing, no persons will be reminded. */
109     String[] toRemind = bb.getReminders(bd13);
110     System.out.println("Number of reminders of bd13 (from empty book): " + toRemind.length);
```

```
    toRemind = bb.getReminders(6, 16);
    System.out.println("Number of reminders of June 16 (from empty book): " + toRemind.length);

    System.out.println("(14)----------------------------------------");
    /* Remove entries from the book.
     * When names are non-existing name, removing their associated entries have no effect.
     */
    bb.removeEntry("A");
    bb.removeEntry("B");
    bb.removeEntry("C");
    bb.removeEntry("D");
    System.out.println("Number of entries: " + bb.getNumberOfEntries());
    System.out.println("Returned number of entries: " + bb.getEntries().length);

    System.out.println("(15)----------------------------------------");
    /* Add new entries to the book.
     * When names are non-existing, new entries are added to the book.
     */
    bb.addEntry(e1.getName(), e1.getBirthday().getMonth(), e1.getBirthday().getDay());
    bb.addEntry("B", bd13);
    bb.addEntry("C", 6, 16);
    System.out.println("Number of entries: " + bb.getNumberOfEntries());
    System.out.println("Returned number of entries: " + bb.getEntries().length);
    System.out.println("First returned entry: " + bb.getEntries()[0].toString());
    System.out.println("Second returned entry: " + bb.getEntries()[1].toString());
    System.out.println("Third returned entry: " + bb.getEntries()[2].toString());

    System.out.println("(16)----------------------------------------");
    System.out.println(bb.toString());

    System.out.println("(17)----------------------------------------");
    System.out.println("Name A exists in book: " + bb.nameExists("A"));
    System.out.println("Name B exists in book: " + bb.nameExists("B"));
    System.out.println("Name C exists in book: " + bb.nameExists("C"));

    System.out.println("(18)----------------------------------------");
    bdOfA = bb.getBirthday("A");
    bdOfB = bb.getBirthday("B");
    bdOfC = bb.getBirthday("C");
    /* Birthdays of existing names are not nulls */
    System.out.println("Birthday of A: " + bdOfA.toString());
    System.out.println("Birthday of B: " + bdOfB.toString());
    System.out.println("Birthday of C: " + bdOfC.toString());

    System.out.println("(19)----------------------------------------");
    /* Return names of persons whose birthdays are January 16.
     * No entries added so far have this birthday.
     */
    toRemind = bb.getReminders(1, 16);
    System.out.println("Number of reminders of January 16: " + toRemind.length);

    System.out.println("(20)----------------------------------------");
    toRemind = bb.getReminders(1, 11);
    /* Return names of persons whose birthdays are January 11.
     * One entry added so far has this birthday.
     */
```

```
167     System.out.println("Number of reminders of January 11: " + toRemind.length);
168     System.out.println("First person to remind: " + toRemind[0]);
169
170     System.out.println("(21)----------------------------------------");
171     toRemind = bb.getReminders(6, 16);
172     /* Return names of persons whose birthdays are June 16.
173      * Two entries added so far have this birthday.
174      */
175     System.out.println("Number of reminders of June 16: " + toRemind.length);
176     System.out.println("First person to remind: " + toRemind[0]);
177     System.out.println("Secon person to remind: " + toRemind[1]);
178
179     System.out.println("(22)----------------------------------------");
180     /* removing non-existing name: no effect */
181     bb.removeEntry("D");
182     System.out.println(bb.toString());
183
184     System.out.println("(23)----------------------------------------");
185     /* removing existing name: remove the associated entry */
186     bb.removeEntry("A");
187     System.out.println(bb.toString());
188
189     System.out.println("(24)----------------------------------------");
190     toRemind = bb.getReminders(1, 11);
191     /* After the entry associated with "A" is deleted,
192      * no entries added so far have this birthday.
193      */
194     System.out.println("Number of reminders of January 11: " + toRemind.length);
195
196     System.out.println("(25)----------------------------------------");
197     /* removing existing name: remove the associated entry */
198     bb.removeEntry("C");
199     System.out.println(bb.toString());
200
201     System.out.println("(26)----------------------------------------");
202     System.out.println("B's birthday: " + bb.getBirthday("B").toString());
203     System.out.println("C's birthday: " + bb.getBirthday("C"));
204
205     System.out.println("(27)----------------------------------------");
206     Birthday bd = new Birthday(6, 16);
207     toRemind = bb.getReminders(bd);
208     /* After the entry associated with "C" is deleted,
209      * only the entry associated with "B" has this birthday.
210      */
211     System.out.println("Number of reminders of June 16: " + toRemind.length);
212     System.out.println("First person to remind: " + toRemind[0]);
213
214     System.out.println("(28)----------------------------------------");
215     bb.addEntry("D", bd04);
216     bb.addEntry("E", 9, 19);
217     System.out.println(bb.toString());
218
219     System.out.println("(29)----------------------------------------");
220     toRemind = bb.getReminders(bd04);
221     System.out.println("Number of reminders for bd04: " + toRemind.length);
222     System.out.println("First person to remind: " + toRemind[0]);
```

```
223
224     System.out.println("(30)----------------------------------------");
225     /* Adding an entry whose name already exists
226      * replaces the associated entry's birthday.
227      */
228     bb.addEntry("E", 04, 14);
229     System.out.println(bb.toString());
230
231     System.out.println("(31)----------------------------------------");
232     toRemind = bb.getReminders(4, 14);
233     System.out.println("Number of reminders for April 14: " + toRemind.length);
234     System.out.println("First person to remind: " + toRemind[0]);
235     System.out.println("Second person to remind: " + toRemind[1]);
236
237     System.out.println("(32)----------------------------------------");
238     /* Non-empty and empty books are not equal. */
239     BirthdayBook bb2 = new BirthdayBook();
240     System.out.println("bb and bb2 are equal: " + bb.equals(bb2));
241
242     System.out.println("(33)----------------------------------------");
243     /* Non-empty books of different sizes are not equal. */
244     bb2.addEntry("B", 6, 16);
245     bb2.addEntry("D", 4, 14);
246     System.out.println("bb and bb2 are equal: " + bb.equals(bb2));
247
248     System.out.println("(34)----------------------------------------");
249     /* Non-empty books,
250      * of same sizes and where entries at corresponding positions are equal,
251      * are equal.
252      */
253     bb2.addEntry("E", bd04);
254     System.out.println("bb and bb2 are equal: " + bb.equals(bb2));
255
256     System.out.println("(35)----------------------------------------");
257     /* Non-empty books of different sizes are not equal. */
258     bb2.addEntry("F", 10, 15);
259     System.out.println("bb and bb2 are equal: " + bb.equals(bb2));
260   }
261 }
```

2.4 Expected Output from Executing `BirthdayBookTester`

```
(01)----------------------------------------
(1, 11)
(02)----------------------------------------
January 11
February 12
March 13
April 14
May 15
June 16
July 17
August 18
September 19
October 20
November 21
December 22
(03)----------------------------------------
bd06: June 16
bd07: July 17
bd13: June 16
Contents of bd06 and bd13 are equal: true
Contents of bd07 and bd13 are equal: false
(04)----------------------------------------
Entry e1's name: A
Entry e1's birthday: January 11
Entry e1's string value: A was born on January 11
(05)----------------------------------------
Entry e2's name: B
Entry e2's birthday: February 12
Entry e2's string value: B was born on February 12
(06)----------------------------------------
e1: A was born on January 11
e2: B was born on February 12
e3: A was born on January 11
Entries e1 and e2 are equal: false
Entries e1 and e3 are equal: true
(07)----------------------------------------
e1: A was born on January 11
e2: A was born on January 11
e3: A was born on January 11
Entries e1 and e2 are equal: true
Entries e1 and e3 are equal: true
(08)----------------------------------------
e1: A was born on January 11
e2: A was born on January 11
e3: A was born on March 13
Entries e1 and e2 are equal: true
Entries e1 and e3 are equal: false
(09)----------------------------------------
Number of entries: 0
Returned number of entries: 0
(10)----------------------------------------
There are 0 entries in the book

(11)----------------------------------------
```

```
Name A exists in book: false
Name B exists in book: false
Name C exists in book: false
(12)----------------------------------------
Birthday of A: null
Birthday of B: null
Birthday of C: null
(13)----------------------------------------
Number of reminders of bd13 (from empty book): 0
Number of reminders of June 16 (from empty book): 0
(14)----------------------------------------
Number of entries: 0
Returned number of entries: 0
(15)----------------------------------------
Number of entries: 3
Returned number of entries: 3
First returned entry: A was born on January 11
Second returned entry: B was born on June 16
Third returned entry: C was born on June 16
(16)----------------------------------------
There are 3 entries in the book
A was born on January 11
B was born on June 16
C was born on June 16
(17)----------------------------------------
Name A exists in book: true
Name B exists in book: true
Name C exists in book: true
(18)----------------------------------------
Birthday of A: January 11
Birthday of B: June 16
Birthday of C: June 16
(19)----------------------------------------
Number of reminders of January 16: 0
(20)----------------------------------------
Number of reminders of January 11: 1
First person to remind: A
(21)----------------------------------------
Number of reminders of June 16: 2
First person to remind: B
Secon person to remind: C
(22)----------------------------------------
There are 3 entries in the book
A was born on January 11
B was born on June 16
C was born on June 16
(23)----------------------------------------
There are 2 entries in the book
B was born on June 16
C was born on June 16
(24)----------------------------------------
Number of reminders of January 11: 0
(25)----------------------------------------
There are 1 entries in the book
B was born on June 16
(26)----------------------------------------
```

```
B's birthday: June 16
C's birthday: null
(27)----------------------------------------
Number of reminders of June 16: 1
First person to remind: B
(28)----------------------------------------
There are 3 entries in the book
B was born on June 16
D was born on April 14
E was born on September 19
(29)----------------------------------------
Number of reminders for bd04: 1
First person to remind: D
(30)----------------------------------------
There are 3 entries in the book
B was born on June 16
D was born on April 14
E was born on April 14
(31)----------------------------------------
Number of reminders for April 14: 2
First person to remind: D
Second person to remind: E
(32)----------------------------------------
bb and bb2 are equal: false
(33)----------------------------------------
bb and bb2 are equal: false
(34)----------------------------------------
bb and bb2 are equal: true
(35)----------------------------------------
bb and bb2 are equal: false
```