University of Windsor

SCHOOL OF COMPUTER SCIENCE
03-60-212 – OOP USING JAVA
SUMMER 2014

ASSIGNMENT– 3

*[Submit your complete work **within the due date and time** as indicated in the CLEW]*

Objectiv: Working with Exception Handling.

Part 1:

Write a java application program that does the followings [Your are encouraged to define a class, say, ***ArrayAccess*** and an application, say, ***TestArrayAccess***:

1. The program should allow a user to input integer values into an integer array and search the array. The size of the array should be decided either through user input or while creating an object of class ***ArrayAccess***.
2. The program should also allow the users to retrieve values from the array by index or by specifying a particular value to locate.
3. The program should handle any exceptions that might arise when inputting values or accessing array elements.
1. The program should throw an ***InputMismatchException,*** if an attempt is made to enter a non-integer value into the array. (You might have to import ***java.util.InputMismatchException.***
2. If an attempt is made to access an element outside the array bounds, catch the ***ArrayIndexOutOfBoundsException*** and display an appropriate error message.
3. If an attempt is made to access an element for which the user has not yet input any value, the program should throw an ***ArrayIndexOutOfBoundsException***.
4. The program should throw a ***NumberNotFoundException*** if a particular value cannot be found in the array during a search. You have to define the class ***NumberNotFoundException*** to use it.

A sample output from test run of TestArrayAccess.java:

```
Created an integer array of size 5

    ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
2
The array is empty. Cannot search an empty array!

    ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
```

```
(3)-Search for a value.
Enter your choice (0 to quit):
3
The array is empty. Cannot search an empty array!

   ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
1
Enter an integer to store:
23

   ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
1
Enter an integer to store:
12

   ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
1
Enter an integer to store:
9

   ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
2
Enter the index:
5
Currently the index must be within 0 - 2

   ****** MENU ******
(1)-Enter a value into the array.
```

```
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
2
Enter the index:
1
Value at position 1 is 12

   ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
3
Enter the value to search:
44
Value to search could be found.

   ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
3
Enter the value to search:
23
Found 23 at position 0

   ****** MENU ******
(1)-Enter a value into the array.
(2)-Retrive a value using index.
(3)-Search for a value.
Enter your choice (0 to quit):
0
Bye...
```

Part 2:

Consider the class ***Date2*** that is provided for you as a part of this assignment. In this class the validity of the date strings have not been checked as in the Date class you defined before. Modify this ***Date2*** class and write a new ***Date*** class, so that it can now handle exceptions if an invalid date string is passed to the constructor. You have to define three exception classes as follows:

1. ***InvalidMonthException***: If the user enters anything other than a legal month number (integers from 1 to 12), your program will throw and catch a ***InvalidMonthException*** and ask the user to ***re-enter a valid month***.
2. ***InvalidYearException***: If the user enters a year that is not in the range 1000 to 3000 (inclusive), then your program will throw and catch a ***InvalidYearException*** and ask the user to ***re-enter a valid year***. (There is nothing very special about the numbers 1000 and 3000 other than giving a good range of likely dates.) See Self-Test Exercise 19 in Chapter 4 for details on leap years.
3. ***InvalidDayException***: If the user enters anything other than a valid day number (integers from 1 to either 28 , 29 , 30 , or 31 , depending on the month and year), then your program will throw and catch a ***InvalidDayException*** and ask the user to ***re-enter a valid day***.
4. You can assume that the user always enters the valid data in the second attempt.

Sample output from TestDate:

```
Invalid Day Exception
Invalid day for July of 2008. Enter a valid day:
7
Invalid Month Exception
Invalid month. Enter a valid month (1 - 12):
8
Invalid Year Exception
Invalid year. Enter a valid year (1000 - 3000):
2014
date1 = July 7, 2008
date2 = August 31, 2010
date3 = February 22, 2014
date1 is earlier than date2
date1 is earlier than date3
date2 is later than date1
date2 is earlier than date3
date3 is later than date1
date3 is later than date2
```