Advanced Object-Oriented Programming

CPT204 – Lab 13
Erick Purwanto

Xi'an Jiaotong-Liverpool University
西交利物浦大学

CPT204 Advanced Object-Oriented Programming

Lab 13

Priority Queue

Welcome !

- Welcome to Lab 13 !
- We are going to create a priority queue with **an resizing array based binary heap** that support logarithmic-time operations
  - double the size of the array if it is full

- You will find in this lab
  1. Lab Exercise 13.1 - 13.4, and their hints
  2. *No Exercises* this week, you can use your time to complete Lab 14 Part A, B : Explicit MinPQ and its application

- Download **lab13** zip files from Learning Mall
- Import the **lab13** files and the library to an IntelliJ project
  - Read **lab1** again for reference

Test Case for Lab Exercise 13.1 - 13.4

- Test case 1:

```
ARBinHeap<Integer> pq = new ARBinHeap<>();
pq.isEmpty();                               →       true
pq.size();                                  →       0

pq.add(6);  pq.add(3);  pq.add(9);
pq.getMin();                                →       3

pq.add(7);  pq.add(5);  pq.add(8);  pq.add(2);

Object[] arr = pq.toArray();                →       [ null 2 5 3 7 6 9 8 ]

pq.delMin();                                →       2

arr = pq.toArray();                         →       [ null 3 5 8 7 6 9 ]
```

Lab Exercise 13.1 ARBinHeap CONSTRUCTORS

- Complete **two constructors** of ARBinHeap that take zero or one argument.
- It initializes an empty binary heap with the given initial capacity,
  - or initial capacity 1 for the one with no arguments.

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 13.1 ARBinHeap CONSTRUCTORS Hints

- you can implement the constructor taking one arguments first,
  - and then calling it for the empty constructor passing the default value of 1
- initialize the array heap with array of Comparable
  - need to add 1 to the initial capacity because we don't use the first array element
  - and then cast to an array of type parameter T
- initialize size

Lab Exercise 13.2 ARBinHeap GETMIN

- Complete the method T **getMin**() of ARBinHeap.
- It returns a smallest item on this binary heap,
    - and this binary heap must not be empty.

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 13.2 ARBinHeap GETMIN Hints

- simply returns the root of the binary heap

Lab Exercise 13.3 ARBinHeap ADD

- Complete the method void **add**(T item) of ARBinHeap.
- It adds a new item to this binary heap.
- Double the size of the array if the array is full.

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 13.3 ARBinHeap ADD Hints

- double the size of the array if it is full
    - use a helper method
- increment size
- place the new item at the next element in the array
    - to keep the tree complete
- swim that new item
    - to maintain the heap property

Lab Exercise 13.4 ARBinHeap DELMIN

- Complete the method T **delMin**() of ARBinHeap.
- It removes and returns a smallest item on this binary heap,
    - and this binary heap must not be empty.
- We do not implement halving/resizing down this time.

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 13.4 ARBinHeap DELMIN Hints

- store the root value
  - to be returned at the end of the method
- swap the root with the last item
  - to keep the tree complete
- decrement size
- sink that new root
  - to maintain the heap property
- nullify the reference to the deleted item to avoid loitering

Test Case for Lab Exercise 13.1 - 13.4

- Test case 1:

```
ARBinHeap<Integer> pq = new ARBinHeap<>();
pq.isEmpty();                                →        true
pq.size();                                   →        0

pq.add(6);  pq.add(3);  pq.add(9);
pq.getMin();                                 →        3

pq.add(7);  pq.add(5);  pq.add(8);  pq.add(2);

Object[] arr = pq.toArray();                 →        [ null 2 5 3 7 6 9 8 ]

pq.delMin();                                 →        2

arr = pq.toArray();                          →        [ null 3 5 8 7 6 9 ]
```

Thank you for your attention !

- In this lab, you have learned:
  - To create a priority queue that supports logarithmic-time operations, with resizing array as the underlying data structure