ITP20003 Java Programming

Lab 8. Try To Catch

Lab 8

- Missions 14 & 15

Team1	김소은	전혜원
Team2	김시온	김아론
Team3	김예군	김지민
Team4	김재윤	윤석규
Team5	박수현	양예진
Team6	백주열	박혜빈
Team7	유채우	Wongani
Team8	심충일	이종원
Team9	이지행	황보효정
Team10	이한빈	이혁재

M14. Calendar: Overview

- Construct calendar classes whose instances represent dates and offer date-related operations
  - Main class
  - Date class
- A Date object represents a date in the Gregorian calendar system
  - A year is a leap year when (1) it is divisible by 400, or (2) it is divisible by four while not being divisible by 100
- Main class
  - Receive a date from Standard Input
  - From the given date, find the dates of next 100, 200, 300, 400, 500 days, and print out them to Standard Outputs
  - Print out the monthly calendar of the given date to Standard Output
    - E.g., for "16 Nov 2018"

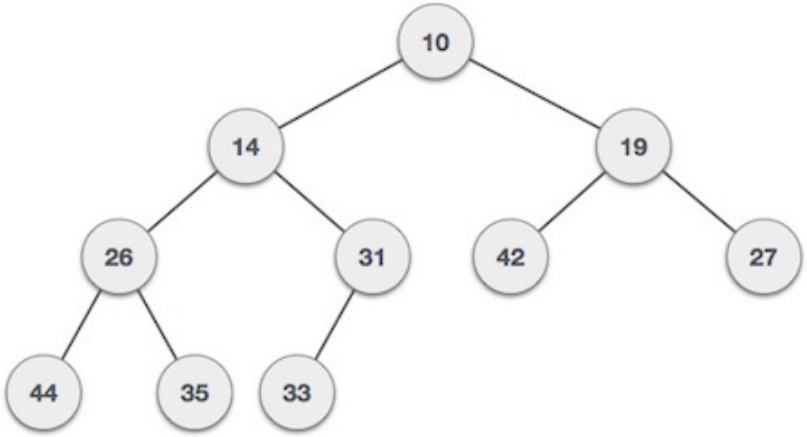
```
Nov 2018
Su Mo Tu We Th Fr Sa
            1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

M14. Calendar: Date class

- **Date(int year, int month, int date)**
  - throws a InvalidArgumentException unless 1≤month≤12 and 1≤date≤31
  - throws a InvalidDateException when there's no such date
- **Date(String str)**
  - parse a date string in a form such as "18-11-16", "2018-11-16", "16 Nov 2018", "16 November 2018", and "Nov 16, 2018"
    - c.f. use Integer.parseInt()
  - throws a InvalidArgumentException unless 1≤month ≤12 and 1≤date ≤31
  - throws a InvalidDateException when there is no such date
- **int diff(Date d)**
  - return how many dates exist between the current date and **d**
- **Date next(int days)**
  - return a new Calendar object of the next **days** date from what the **this** date object represents
- **String toString()**
  - return a string that shows the year, the month, the date and the day,
  - c.f. Jan 1, 1970 is Thursday

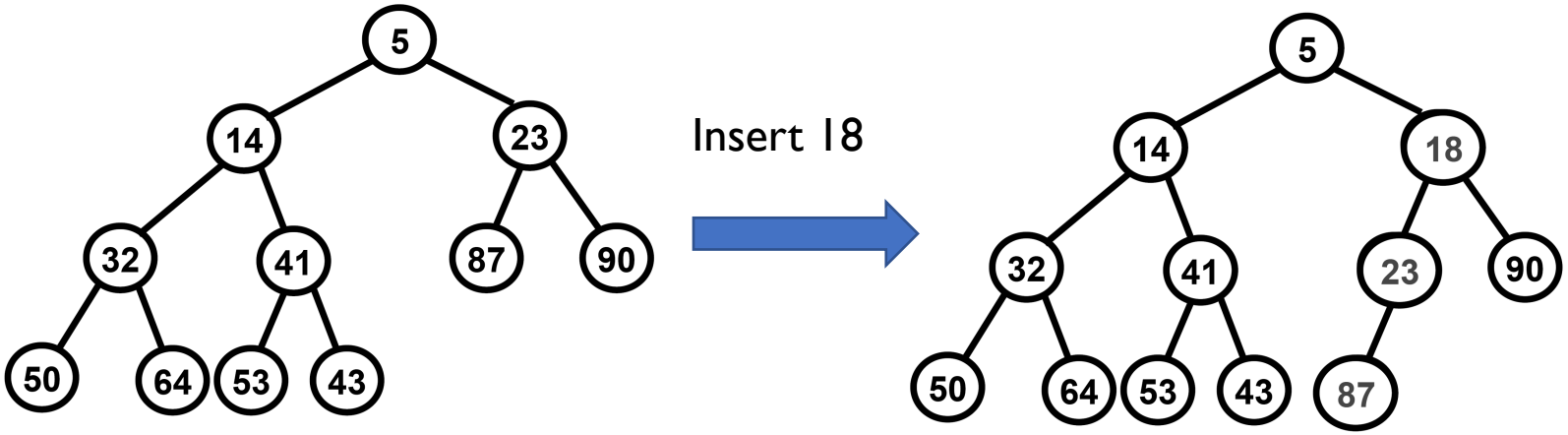
M15. Priority Queue with Min-Heap (1/5)

- A priority queue receives objects by the queue operation and then returns the objects in ascending order for the dequeue operation
- A Min-Heap is a binary tree where elements are partially ordered
  - the parent element is less than equal to its children elements
  - the root element is the most least element
  - both enqueue and dequeue operations are efficient
    - Both enqueue operation and dequeue operation take *m log(n)* steps when the heap contains *n* elements



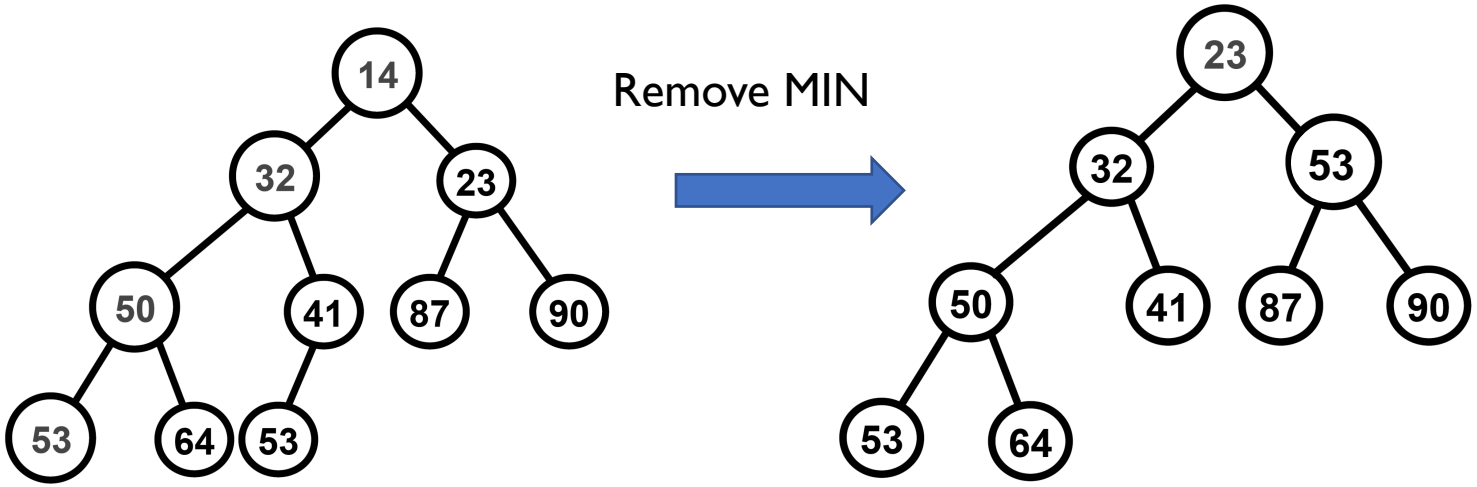
M15. Priority Queue with Min-Heap (2/5)

- Enqueue operation
  - Put a given element to the last point at a binary tree
  - Swap the new element with its parent until the heap property gets satisfied (i.e., the parent element must be less than or equal to its children elements)



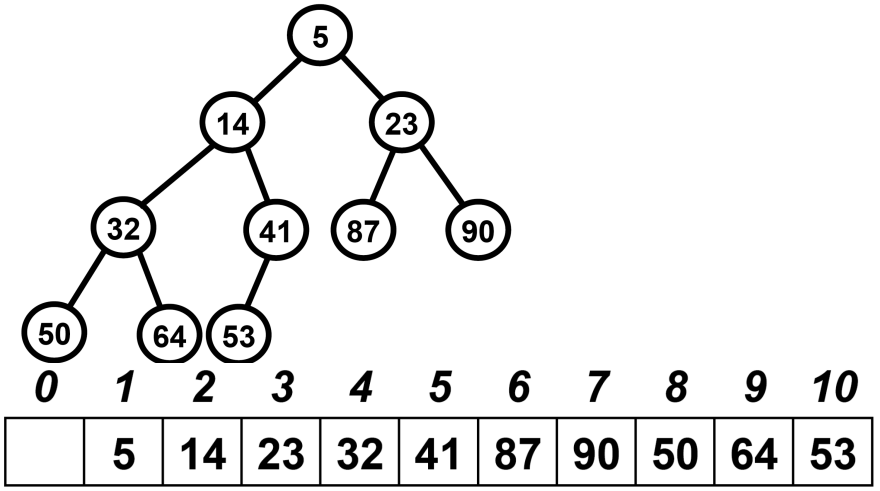
M15. Priority Queue with Min-Heap (3/5)

- Dequeue operation
  - Return the element at the root, which is the least element
  - Put the element at the last point to the root
  - Swap the element placed at the root with one of its children until the heap property gets satisfied



M15. Priority Queue with Min-Heap (4/5)

- Implement the Queue interface
  - void enqueue(Comparable e)
  - Comparable dequeue()
- Use an array as a container
- Methods
  - MinHeap(int capacity)
    - create a element container array with the given capacity size
    - throw a InvalidArguementException if capacity <= 0
  - void enqueue(Comparable e)
    - throw a OverflowException if the given element cannot be inserted since the elements are fully failed
  - Comparable dequeue()
    - throw a UnderflowException if there is no element in the heap
  - int size()



M15. Priority Queue with Min-Heap (5/5)

- The main class creates a Min-Heap object and then receive a sequence of commands from users
  - Receive the initial capacity as a command line argument
  - Two cammands
    - insert <str>
      - insert the given string <str> to a Min-Heap in a lexicographical order
    - delete
      - remove the least element in the heap