CIS 2168 Data Structures **Fall 2014**

Lecture 3: Sep. 9 &11

Lecturer: Anwar Mamat

Disclaimer: *These notes may be distributed outside this class only with the permission of the Instructor.*

3.1 Comparable and Comparator

Listing 1: Date Class

```
1  /*
2   * Simple Date Class
3   */
4  public class Date implements Comparable<Date> {
5      private static final int[]
6          DAYS = {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
7      private final int month;    // month (between 1 and 12)
8      private final int day;      // day   (between 1 and DAYS[month]
9      private final int year;     // year
10
11     /**
12      * Initializes a new date from the month, day, and year.
13      * @param month the month (between 1 and 12)
14      * @param day the day (between 1 and 28-31, depending on the month)
15      * @param year the year
16      * @throws IllegalArgumentException if the date is invalid
17      */
18     public Date(int month, int day, int year) {
19         if (!isValid(month, day, year)) {System.out.println("Invalid date");}
20     }
21     /**
22      * @return month return the month
23      */
24     public int getMonth(){
25         return month;
26     }
27     /**
28      * @return year return the year
29      */
30     public int getYear(){
31         return year;
32     }
33     /**
34      * @return day return the day
35      */
36     public int getDay(){
```

```
37          return day;
38      }
39
40      // is the given date valid?
41      private static boolean isValid(int m, int d, int y) {
42          if (m < 1 || m > 12)       {return false;}
43          if (d < 1 || d > DAYS[m]) {return false;}
44          if (m == 2 && d == 29 && !isLeapYear(y)) {return false;}
45          return true;
46      }
47
48      /**
49       * Is year y a leap year?
50       * @return true if y is a leap year; false otherwise
51       */
52      private static boolean isLeapYear(int y) {
53          if (y % 400 == 0) {return true;}
54          if (y % 100 == 0) {return false;}
55          return y % 4 == 0;
56      }
57
58      /**
59       * Compare this date to that date.
60       * @return { a negative integer, zero, or a positive integer }, depending
61       *    on whether this date is { before, equal to, after } that date
62       */
63      @Override
64      public int compareTo(Date that) {
65          if (this.year  < that.year)  {return -1;}
66          if (this.year  > that.year)  {return +1;}
67          if (this.month < that.month) {return -1;}
68          if (this.month > that.month) {return +1;}
69          if (this.day   < that.day)   {return -1;}
70          if (this.day   > that.day)   {return +1;}
71          return 0;
72      }
73
74      /**
75       * Return a string representation of this date.
76       * @return the string representation in the foramt MM/DD/YYYY
77       */
78      @Override
79      public String toString() {
80          return month + "/" + day + "/" + year;
81      }
82
83  }
```

Listing 2: Compare two date objects by month

```
1  import java.util.Comparator;
```

```
2  public class MonthComparator implements Comparator<Date> {
3      @Override
4      public int compare(Date d1, Date d2) {
5          if (d1.getMonth() < d2.getMonth()) {return - 1};
6          if (d1.getMonth() > d2.getMonth()) {return + 1};
7          return 0;
8      }
9  }
```

Listing 3: Compare two date objects by day

```
1  import java.util.Comparator;
2  public class DayComparator implements Comparator<Date> {
3      @Override
4      public int compare(Date d1, Date d2) {
5          if (d1.getDay() < d2.getDay()) {return -1};
6          if (d1.getDay() > d2.getDay()) {return +1};
7          return 0;
8      }
9  }
```

Listing 4: Test Date Class

```
1  /**
2   * Date class unit test
3   */
4  package date;
5  import java.util.Arrays;
6  public class DateTest {
7      public static void main(String[] args){
8          Date[] date = new Date[3];
9          date[0] = new Date(5,21,2014);
10         date[1] = new Date(10,28,2013);
11         date[2] = new Date(1,25,2011);
12         //sort the dates by non-decreasing order
13         Arrays.sort(date,new DayComparator());
14         for(Date d:date ){
15             System.out.println(d);
16         }
17         //sort the dates by month order
18         Arrays.sort(date,new MonthComparator());
19         for(Date d:date ){
20             System.out.println(d);
21         }
22         //sort the dates by day order
23         Arrays.sort(date,new DayComparator());
24         for(Date d:date ){
25             System.out.println(d);
26         }
27     }
28 }
```

Table 3.1: Output

Before Sort	After sort	Sort by month	Sort by day
5/21/2014	1/25/2011	1/25/2011	5/21/2014
10/28/2013	10/28/2013	5/21/2014	1/25/2011
1/25/2011	5/21/2014	10/28/2013	10/28/2013

3.2 SortedBag Class

Listing 5: SortedBag Class

```
1  public class SortedBag<E extends Comparable<E>>
2              extends Bag<E> {
3      /**
4      *  Insert new items into the bag in sorted order
5      * @param item the new item to be inserted.
6      */
7      @Override
8      public void insert(E item)
9          {
10                 if(arraySize == capacity){
11                         resize();
12                 }
13                 if(arraySize ==0){
14                     items[0] = item;
15                     arraySize++;
16                     return;
17                 }
18                 int index = arraySize - 1;
19                 while(items[index].compareTo(item) > 0){
20                     items[index+1] = items[index];
21                     index--;
22                     if(index < 0) break;
23                 }
24                 items[++index] = item;
25                 arraySize++;
26         }
27
28         /**
29         *   Does the bag contains a given item?
30         * @return true if bag contains the item. false otherwise
31         */
32         @Override
33         public boolean contains(E item)
34         {
35             int m = 0;
36             int s = 0;
37             int t = arraySize - 1;
38             while(s <= t){
39                 m = (t + s) / 2;
```

```
40                  if(items[m].equals(item)) return true;
41                  if(items[m].compareTo(item) > 0){
42                      t = m - 1;
43                  }else{
44                      s = m + 1;
45                  }
46              }
47              return true;
48          }
49  }
```

Listing 6: Test SortedBag

```
1  Bag<Integer> bag = new SortedBag();
```