```
1  import java.util.Locale;
2  import java.util.Scanner;
3
4  public class Observation {
5     ////////////////////////////////////
6     ///// Uppgift A1: Instansvaiabler/////
7     ...
8     ////////////////////////////////////////
9
10
11
12    ////////////////////////////////
13    /// Uppgift A2: Konstruktor ///
14    ...
15    ////////////////////////////////
16
17
18
19    ///////////////////////////////////
20    /// Uppgift A3: toString-metod ///
21    ...
22    ///////////////////////////////////
23
24
25
26    ///////////////////////////////////
27    /// Uppgift A4: Metoden getWind ///
28    ...
29    ///////////////////////////////////
30
31
32    /**
33     * Checks if the temperature is below a specified limit
34     * @param limit The limit value
35     * @return true if the temperature is below a specified limit else false
36     */
37    ////////////////////////////////////
38    /// Uppgift A5: Metoden belowTemp ///
39    ...
40    ////////////////////////////////////
41
42
43
44    ////////////////////////////////
45    /// Uppgift B2: Metoden read ///
46    /**
47     * Reads the observation data using an already created Scanner object.
48     *
49     * If the first item to be read is anything but a number null will be
50     * returned otherwise two numbers (wind and temperature) will be read and
51     * an Observation object with these values will be created and returned.
52     * If the first number is not followed by a number a RuntimeException
53     * will be thrown
54     *
55     * @param fsc A created Scanner object
56     * @return An observation object or null
57     * @throws RuntimeException if there are missing data items.
58     */
59     ...
60    ////////////////////////////////
61
62
63    /**
64     * Test and demonstration program
65     */
66    public static void main(String[] args) {
67
68       System.out.println("Testing the constructor, the toString and " +
69                          "the belowTemp methods");
70       Observation[] observations = new Observation[4];
71       observations[0] = new Observation(3.5, 5.0);
72       observations[1] = new Observation(4.1, -3.0);
73       observations[2] = new Observation(9.8, -5.2);
```



```
74       observations[3] = new Observation(3.5, 2.7);
75       for (int i=0; i<observations.length; i++) {
76          System.out.print(observations[i]);
77          if (observations[i].belowTemp(0)) {
78             System.out.println(" freezing");
79          } else {
80             System.out.println();
81          }
82       }
83
84       Locale.setDefault(Locale.US);
85       System.out.println("\nTesting the read method");
86       String str = "1.0 2.5 5.8 -1.6";
87       Scanner scan = new Scanner(str);
88       Observation obs = Observation.read(scan);
89       while (obs != null) {
90          System.out.println(obs);
91          obs = Observation.read(scan);
92       }
93
94       // Try to read an incomplete observation (just one value)
95       str = "1.5";
96       scan = new Scanner(str);
97       obs = Observation.read(scan);
98    }
99
100 }
101
102 /* Output:
103  *
104  Testing the constructor, the toString and the belowTemp methods
105  <3.5, 5.0>
106  <4.1, -3.0> freezing
107  <9.8, -5.2> freezing
108  <3.5, 2.7>
109
110  Testing the read method
111  <1.0, 2.5>
112  <5.8, -1.6>
113  java.lang.RuntimeException: Observation could not be read
114  at Observation.read(Observation.java:??)
115
116  */
```

```java
import java.util.ArrayList;
import java.util.Scanner;
import java.util.Locale;
import java.util.Arrays;

/**
 * Represents a weather-station
 */
public class Station {

   ////////////////////////////////////////////
   /// Uppgift A6. Instansvariableroch konstruktor
   ...
   ////////////////////////////////////////////


   ////////////////////////////////////
   /// Uppgift A7. numberOfObservations ///
   ...
   ////////////////////////////////////


   ///////////////////////////////
   /// Uppgift 8. addObservation ///
   ...
   }
   ///////////////////////////////


   /**
    * Gets the name of the station
    * @return The name of the station
    */
   public String getName() {
      return name;
   }


   ////////////////////////////
   /// Uppgift 9. medelvind ///
   ...
   ////////////////////////////


   ////////////////////////////
   /// Uppgift 10. meanUsable ///
   ...
   ////////////////////////////


   ////////////////////////////
   /// Uppgift 11. windArray ///
   ...
   ////////////////////////////


   /**
    * @return A string representation of the station
    * and its observations
    */
   public String toString() {
      String s = String.format("%-13s %3d %6.1f",
                               name,
                               numberOfObservations(),
                               meanWind()
                              );
      return s;
   }

   /**
    * Produces a printout of the station together with
    * its obeservations. The observations are written 5 per row.
    */
```



```java
   public void print() {
      System.out.print(name);
      for (int i=0; i<theObservations.size(); i++) {
         if (i%5==0) {
            System.out.print("\n\t\t  ");
         }
         System.out.print(theObservations.get(i) + " ");
      }
      System.out.println();
   }

   /**
    * Test and demonstrates the Station class
    */
   public static void main(String[] args) {
      Locale.setDefault(Locale.US);

      String str = "Landsort 2.6 6.0  8.0 4.7  3.4 5.7  25.5 3.4  7.3 5.7  7.2 5.9";
      Scanner scan= new Scanner(str);
      String name = scan.next();
      Station aStation = new Station(name);
      Observation obs = Observation.read(scan);
      while ( obs != null ) {
         aStation.addObservation(obs);
         obs = Observation.read(scan);
      }
      aStation.print();
      System.out.println("Mean wind     : " + aStation.meanWind());
      System.out.println("Number obs    : " + aStation.numberOfObservations());
      System.out.println("Mean usable wind : " + aStation.meanUsable(4., 25.));
      System.out.println("Wind array    : " + Arrays.toString(aStation.windArray()))
;
      System.out.println();

      // Test incomplete observation data
      scan = new Scanner("Nowhere 99.");
      name = scan.next();
      System.out.println(name + " should cause a RuntimeException\n");
      obs = Observation.read(scan);
   }

}

/* Output:
 *
Landsort
                 <2.6, 6.0> <8.0, 4.7> <3.4, 5.7> <25.5, 3.4> <7.3, 5.7>
                 <7.2, 5.9>
Mean wind        : 9.0
Number obs       : 6
Mean usable wind : 7.5
Wind array       : [2.6, 8.0, 3.4, 25.5, 7.3, 7.2]

Nowhere should cause a RuntimeException

java.lang.RuntimeException: Observation could not be read
 at Observation.read(Observation.java:63)
 at Station.main(Station.java:170)

 */
```

```java
import java.util.ArrayList;
import java.util.Scanner;
import java.util.Locale;

import java.io.*;

public class StationList {
   private ArrayList<Station> stations;

   /**
    * Construct a Station List
    */
   public StationList() {
      stations = new ArrayList<Station>();
   }


   ////////////////////
   /// A12. print ///
   /**
    * Prints the list of stations one per line using station's toString-method.
    */
   public void print() {
   ...
   }
   ////////////////////



   ///////////////////////////////////
   /// Uppgift B1. addNameSorted ///
   public Station addNameSorted(String name) {
      ...
   }
   ///////////////////////////////////


   ///////////////////////////////////
   /// Uppgift B3. load ///
   /**
    * Reads a file with station objects
    * @param filename The file name
    */
   public void load(String filename) throws IOException {
      ...
   }
   ///////////////////////////////////


   /**
    * Test and demonstration program
    */
   public static void main(String[] args) throws IOException{
      Locale.setDefault(Locale.US);
      StationList st = new StationList();
      st.load("stations.txt");
      st.print();
   }
}


/* Contents of stations.txt:

Oerskaer 3 4  6 8  10 12  8 5
Harstena 3 7  9 8  3 6  5 4
Landsort 2 4  3 8  6 12  11 8 13 7 12 6
Harstena 12 8 17 9 24 7 26 6 23 6
Almagrundet 2 5  7 9  13 15
Eggegrund 8 3  12 2 17 3 22 4
Utklippan 9 4 8 3 12 4 6 4 3 5 2 7 0 6
Nordkoster 12 4 13 5 11 5 10 7 3 6
Vinga 12 4 16 5 18 5 19 3 18 8
Hoburg  2 5 3 6 3 7 5 5 6 3 3 4
```



```java
Nidingen 4 5 3 6 7 5 11 6
Harstena 5 4 4 7 3 8 3 8 4 9
Vinga 8 25 7 26 8 19 7 17

*/


/* Output:

Stationslista:
                Antal Medel
Almagrundet         3    7.3
Eggegrund           4   14.8
Harstena           14   10.1
Hoburg              6    3.7
Landsort            6    7.8
Nidingen            4    6.3
Nordkoster          5    9.8
Oerskaer            4    6.8
Utklippan           7    5.7
Vinga               9   12.6

*/
```