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

public class Observation {

  // Instansvariabler
     ...

  // Konstruktor
     ...



  public String toString() {
    ...
  }


  public double getWind() {
    ...
  }


  public boolean belowTemp(double limit) {
    ...
  }


  /**
   * Reads the observation data using an already createdScanner object.
   *
   * If the first item to be read not is a number, null will be returned
   * otherwise two numbers (wind and temperature) will be read and an
   * Observation object with these values will be created and returned.
   *
   * If the first number is not followed by a number (i e there is a
   * wind item but not a temperature item) a RuntimeException
   * will be thrown
   *
   * @param fsc A created Scanner object
   * @return An observation object or null
   * @throws RuntimeException if there are missing data items.
   */
  public static Observation read(Scanner fsc) {
     ...
  }

  /**
   * Test and demonstration program
   */
  public static void main(String[] args) {

    System.out.println("Testing the constructor, the toString and " +
                       "the belowTemp methods");
    Observation[] observations = new Observation[4];
    observations[0] = new Observation(3.5, 5.0);
    observations[1] = new Observation(4.1, -3.0);
    observations[2] = new Observation(9.8, -5.2);
    observations[3] = new Observation(3.5, 2.7);
    for (int i=0; i<observations.length; i++) {
      System.out.print(observations[i]);
      if (observations[i].belowTemp(0)) {
        System.out.println(" freezing");
      } else {
        System.out.println();
      }
    }

// Code below this point demonstrates things for task 5 and 6
```



```
    Locale.setDefault(Locale.US);
    System.out.println("\nTesting the read method");
    String str = "1.0 2.5 5.8 -1.6";
    Scanner scan = new Scanner(str);
    Observation obs = Observation.read(scan);
    while (obs != null) {
      System.out.println(obs);
      obs = Observation.read(scan);
    }

    // Try to read an incomplete observation (just one value)
    str = "1.5";
    scan = new Scanner(str);
    obs = Observation.read(scan);
  }

}

/* Output:

Testing the constructor, the toString and the belowTemp methods
<3.5, 5.0>
<4.1, -3.0> freezing
<9.8, -5.2> freezing
<3.5, 2.7>

Testing the read method
<1.0, 2.5>
<5.8, -1.6>
java.lang.RuntimeException: Observation could not be read
       at Observation.read(Observation.java:??)

 */
```

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

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

  // Instansvariabler
  ...

  /**
   * Constructs a weather station
   * @param name The name of the station
   */
  public Station(String name) {
    ...
  }

  /**
   * Adds an observation to the list of obervations
   * @param obs The observation to be added
   */
  public void addObservation(Observation obs) {
    ...
  }

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

  /**
   * Computes the mean wind
   * @return The mean wind
   */
  public double meanWind() {
    ...
  }

  /**
   * Gets the number of observations
   * @return The number of observations
   */
  public int numberOfObservations() {
    ...
  }

  /**
   * Computes the mean value of the usable wind observations
   * @param low The lowest usable wind speed
   * @param high The highest usable wind speed
   * @return The mean of the usable observations
   */
  public double meanUsable(double low, double high) {
    ...
  }

  /**
   * Returns the wind observations as an array
   * @return An array with wind observations
   */
  public double[] 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.
   */
  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:??)

*/
```

```
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>();
  }


  /**
   * Prints the list of stations one per line using station's toString-method.
   */
  public void print() {
    System.out.println("\nStationslista:");
    System.out.println("      Antal Medel ");
    for (Station s: stations) {
      System.out.println(s);
    }
  }

  public Station addNameSorted(String name) {
     ...
  }

  /**
   * 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
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

*/
```