```
1  import java.util.ArrayList;
2  import java.util.Scanner;
3
4  /**
5   * Represents the result for a single student
6   */
7  public class Exam {
8    private ArrayList<Integer> scoresPartA;
9    private ArrayList<Integer> scoresPartB;
10   private String student;
11
12   public Exam(String student) {
13      // Uppgift
14   }
15
16   public String getStudent() {
17     return student;
18   }
19
20   /**
21    * Appends a score to part A
22    * @param score The score to be appended
23    */
24   public void addPartA(int score) {
25     // Uppgift
26   }
27
28   /**
29    * Appends a score to part B
30    * @param score The score to be appended
31    */
32   public void addPartB(int score) {
33     // Uppgift
34   }
35
36   /**
37    * Computes the sum of scores on part B
38    * @return The sum of scores
39    */
40   public int sumPartB() {
41     // Uppgift
42   }
43
44   /**
45    * Returns part A as a String
46    * @return A String representation of the object
47    */
48   public String toStringPartA() {
49     // Uppgift
50   }
51
52   /**
53    * Returns part B as a String
54    * @return A String representation of part B
55    */
56   public String toStringPartB() {
57    // Behöver inte skrivas
58   }
59
60   /**
61    * Checks if the exam has passed
62    * @param minScore The required minimum value for the individual scores
63    * @param limit3 The required sum on part A to pass
64    * @return true if this exam has passed, else false
65    */
66   public boolean passed(int minScore, int limit3) {
67     // Uppgift
68   }
69
70   /**
71    * Computes and returns the grade
72    * @param minScore The minimal score for individual score on part A
73    * @param limit3 Required score on part A to get grade 3 (or higher)
```



```
74    * @param limit4 Required score on part B to get grade 4
75    * @param limit5 Required score on part B to get grade 5
76    * @return The grade
77    */
78   public int getGrade(int minScore, int limit3, int limit4, int limit5) {
79     // Uppgift
80   }
81
82   /**
83    * Constructs a String representation of an exam object
84    * @return A string representing the object
85    */
86    public String toString() {
87     return String.format("%-10s \t%-15s \t%-20s",
88                          student,
89                          toStringPartA(),
90                          toStringPartB());
91   }
92
93   /**
94    * A main method for a small demonstration of the methods
95    */
96   public static void main(String[] args) {
97     Exam exam = new Exam("Kalle");
98     exam.addPartA(7);
99     exam.addPartA(8);
100    exam.addPartA(1);
101    exam.addPartB(0);
102    exam.addPartB(3);
103    exam.addPartB(5);
104    System.out.println("Student  :" + exam.getStudent());
105    System.out.println("Poäng del A: " + exam.toStringPartA());
106    System.out.println("Poäng del B: " + exam.toStringPartB());
107    System.out.println("toString  :" + exam.toString());
108    System.out.println("Godkänd  :" + exam.passed(1, 5));
109    System.out.println("Betyg  :" + exam.getGrade(1, 5, 5, 10));
110    System.out.println("\nMed högre minimigräns");
111    System.out.println("Godkänd  :" + exam.passed(2, 5));
112    System.out.println("Betyg  :" + exam.getGrade(2, 5, 5, 10));
113  }
114 }
115
116 /* Output:
117
118 Student    : Kalle
119 Poäng del A: [7, 8, 1]
120 Poäng del B: [0, 3, 5]
121 toString   : Kalle       [7, 8, 1]        [0, 3, 5]
122 Godkänd    : true
123 Betyg      : 4
124
125 Med högre minimigräns
126 Godkänd    : false
127 Betyg      : 0
128
129 */
130
```

```
1  import java.io.IOException;
2  import java.io.FileReader;
3  import java.util.ArrayList;
4  import java.util.Scanner;
5
6  /**
7   * Represents a set of exams
8   */
9   public class ExamCollection {
10    private ArrayList<Exam> theCollection;
11
12    /**
13     * Constructs an ExamCollection using information from a file
14     * using the ExamReader class.
15     * @param filename Name of the file to be used for reading exams
16     */
17    public ExamCollection(String filename) throws IOException {
18      // Uppgift
19    }
20
21    /**
22     * Prints a list of students with grades and scores
23     * @param minGrade The required minimum individual score in part A
24     * @param limit3 The required sum in part A to pass the exam
25     * @param limit4 The required sum in part B to get grade 4
26     * @param limit5 The required sum in part B to get grade 5
27     */
28    public void printGradeList(int minGrade, int limit3,
29                               int limit4, int limit5) {
30       System.out.println("Betygslista");
31       System.out.format("Gränser: 3: %-2d, 4: %-2d, 5: %-2d\n",
32                         limit3,  limit4, limit5);
33       System.out.println("Minsta poäng per uppgift på del A för godkänd: " + minGrade);
34       System.out.println("Namn     Betyg     Poäng A        Poäng B");
35       for (Exam exam : theCollection) {
36           System.out.format("%-10s %2d \t %-20s %-20s\n",
37                             exam.getStudent(),
38                             exam.getGrade(minGrade, limit3, limit4, limit5),
39                             exam.toStringPartA(),
40                             exam.toStringPartB());
41       }
42    }
43
44    /**
45     * Computes and prints the grade distribution
46     * @param minGrade The required minimum individual score in part A
47     * @param limit3 The required sum in part A to pass the exam
48     * @param limit4 The required sum in part B to get grade 4
49     * @param limit5 The required sum in part B to get grade 5
50     */
51    public void printStatistics(int minGrade, int limit3,
52                                int limit4, int limit5) {
53
54      // Uppgift
55      // Koden nedan är given
56       System.out.println("\nStatistik");
57       System.out.format(" U:%2d \n 3:%2d \n 4:%2d \n 5:%2d \n",
58                         distribution[0],
59                         distribution[3],
60                         distribution[4],
61                         distribution[5]);
62    }
63
64    /**
65     * A main method that reads a file with exams and prints the result
66     */
67    public static void main(String[] args) throws IOException {
68       ExamCollection exCollection = new ExamCollection("exams.txt");
69       exCollection.printGradeList(2, 20, 7, 15);
70       exCollection.printStatistics(2, 20, 7, 15);
71    }
72 }
73
```



```
74 /* Output:
75
76 Betygslista
77 Gränser: 3: 20, 4: 7 , 5: 15
78 Minsta poäng per uppgift på del A för godkänd: 2
79 Namn      Betyg      Poäng A              Poäng B
80 Kalle       3        [8, 5, 8]            []
81 Lisa        5        [7, 7, 6]            [8, 6, 8, 7]
82 Olle        0        [9, 2, 5]            [1, 7, 10, 2]
83 Pelle       0        []                   []
84 Lotta       4        [7, 10, 8]           [3, 0, 6, 5]
85 Urban       0        [12, 0, 8]           [1, 10, 10, 5]
86
87 Statistik
88  U: 3
89  3: 1
90  4: 1
91  5: 1
92 */
93
```

Mar 12, 15 7:28 **ExamReader.java** Page 1/1

```java
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

/**
 * Creates exam objects by reading information from a file
 * An Exam object is created from three lines:
 * Line 1: Student code or name
 * Line 2: A series of integers giving the score for each task in part A
 * Line 3: A series of integers giving the score for each task in part B
 * Example:
 *    Kalle
 *    3 7 5
 *    2 0 9 3 1
 */
public class ExamReader {

  private Scanner fileScanner;

  /**
   * Connects the reader to a file
   * @param filename Name of file to be used for reading exam information
   */
  public ExamReader(String filename) throws IOException {
    fileScanner = new Scanner(new FileReader(filename));
  }

  /**
   * Reads information about one exam and creates an Exam-object
   * @return A created Exam-object or null if end of file is reached
   */
  public Exam next() {
    if (!fileScanner.hasNext()) {
      return null;
    }
    String name = fileScanner.nextLine();
    Exam exam = new Exam(name);

    Scanner lineScanner;

    // Part A
    lineScanner = new Scanner(fileScanner.nextLine());
    while (lineScanner.hasNextInt()) {
      exam.addPartA(lineScanner.nextInt());
    }

    // Part B
    lineScanner = new Scanner(fileScanner.nextLine());
    while (lineScanner.hasNextInt()) {
      exam.addPartB(lineScanner.nextInt());
    }
    return exam;
  }

  /**
   * A small test method
   */
  public static void main(String[] args) throws IOException {
    ExamReader er = new ExamReader("exams.txt");
    Exam exam = er.next();
    while (exam != null) {
      System.out.println(exam);
      exam = er.next();
    }
  }
}
```

Mar 11, 15 13:33 **exams.txt** Page 1/1

```
Kalle
8 5 8

Lisa
7 7 6
8 6 8 7
Olle
9 2 5
1 7 10 2
Pelle


Lotta
7 10 8
3 0 6 5
Urban
12 0 8
1 10 10 5
```