# Assignment 3: EXAMPLE

Implement the next release of your term project. You will incorporate generics. The same instructions as in Assignment 2 applies to this completed Word document, the gray text, the 5 page limit, appendices, JUnit tests, and a ReadMe file.

## 3.1 SUMMARY DESCRIPTION, UPDATED AS NECESSARY

One- or two-paragraph overall description of your proposed term project. Color red the parts changed from Assignment 2.

Knowla (Knowledge assessment and learning) is a scrambling/unscrambling tool for instructors and students to learn material that is between a paragraph and several pages in length. Knowla facilitates testing on the material and being automatically graded, which is the focus of this project.

This project concerns the grading of Knowla tests, which we will call *K-Grade*. K-Grade allows the instructor to assign points to qualities that they want in students answers—expressed in terms of *rubric sets*. The grading philosophy is to assign only positive points for designated favorable qualities in the test-taker’s answer (i.e., not to subtract for deficiencies).

## 3.2 ADDITIONAL REQUIREMENTS (FEATURES) IMPLEMENTED IN THIS RELEASE

Title and one or two sentences per requirement. Don’t repeat requirements implemented for prior assignments unless they are necessary to provide context—in which case, make it clear which are new vs. old.

### 3.2.1 Test-taker I/O (OLD REQUIREMENT)

K-Grade requests from the user a sequence of integers (possibly repeated).

### 3.2.2 Tester Specifications: Consecutive (OLD REQUIREMENT)

K-Grade allows the tester to customize points awarded for designated consecutive pairs in the student’s solution, together with feedback for each.

### 3.2.3 Tester Specifications: Order (OLD REQUIREMENT)

K-Grade allows the tester to customize points awarded for designated pairs in the student’s solution to appear in desired order (i.e., not necessarily consecutive), together with feedback for each.

### 3.2.4 Score (NEW)

K-Grade computes the test-taker’s score based on the rubrics, and reports it on the console.

### 3.2.5 Explanation reported (OLD REQUIREMENT)

K-Grade computes the test-taker’s score based on the rubrics, and reports it on the console.

## 3.3 I/O EVIDENCE OF ACCOMPLISHING THE REQUIREMENTS LISTED ABOVE

Provide an example of input / output illustrating the features above

### Input

File order-rubrics.txt

4 6 8 You are correct that #4 occurs before #6 (8 points).

4 8 5 You are correct that #2 occurs before #4 (5 points).

1 3 7 You are correct that #1 occurs before #3 (7 points).

File consecutive-rubrics.txt

1 2 7 You are correct that #1 and #2 are consecutive (7 points).

2 3 2 You are correct that #2 and #3 are consecutive (2 points).

3 4 8 You are correct that #3 and #4 are consecutive (8 points).

### Input / Output

Console I/O:

feedback.txt...being built from scratch.

Please enter student answer as in '2 3 4 6 8 1': 2 3 4 6 8 1

Student answer is--->2 3 4 6 8 1

You are correct that #4 occurs before #6 (8 points).<--written to feedback.txt

You are correct that #2 occurs before #4 (5 points).<--written to feedback.txt

Ordering grade is 13.0 out of 20.0

You are correct that #2 and #3 are consecutive (2 points).<--written to feedback.txt

You are correct that #3 and #4 are consecutive (8 points).<--written to feedback.txt

Consecutive grade is 10.0 out of 17.0

Overall grade is 23.0 out of 37.0

Check file feedback.txt for your feedback.

### Output File

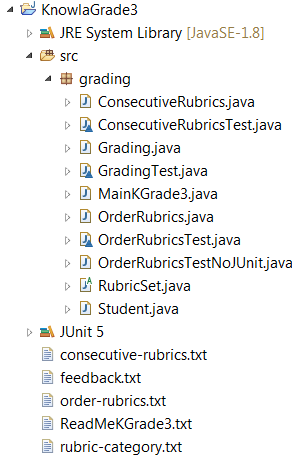
File feedback.txt

You are correct that #2 occurs before #4.

You are correct that #5 and #6 are consecutive.

## 3.4 YOUR DIRECTORY

Show a screenshot of your directory. This should include a parallel directory of JUnit tests—package-by-package, class-by-class, and method-by-method, except for trivial and inapplicable ones.



## 3.5 YOUR UPDATED CLASS MODEL AND CLARIFICATION OF HOW THE EXECUTION WORKS

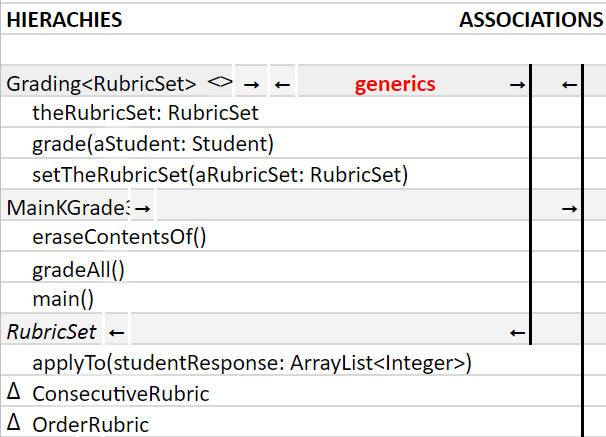
Supply a main use case, the class model, and the sequence diagram corresponding to the use case. These should be consistent. Indicate clearly in your class model where you applied generics. To do this use tools, PowerPoint, or a combine models as in [this example](https://docs.google.com/spreadsheets/d/1HrzhVzA0jcYGlZcdZ4j7613Ey_8HmMSbLT7r5FIifr8/edit?usp=sharing) (which you are free to cut and paste from). Insert indications in red to show where generics apply (as in the code below).

Please see [here](https://docs.google.com/spreadsheets/d/1Tfd6fgJn7rRZlEmHcd0T9j2wNnBpIDUlUV3VoDDc5i8/edit?usp=sharing).

## 3.6 WHERE GENERICS ARE IMPLEMENTED

### 3.6.1 Class model fragment showing generic class

Explain where and how you applied *generic classes* in your class model.



### 3.6.2 Code (including test code), input (if applicable), and output showing generics

Explain why the use of *generics* is appropriate here.

Grading is performed with various rubric sets. The overall functional sequence is the same in each case—hence the use of generics.

CODE:

// Base class for sets of rubrics of one type

**public** **class** Grading<ARubricSet **extends** RubricSet> {

**private** ARubricSet theRubricSet;

**public** RubricSet getTheRubricSet(){

**return** **this**.theRubricSet;

}

**public** **void** setTheRubricSet(ARubricSet aRubricSet){

**this**.theRubricSet=aRubricSet;

}

**public** **double**[] grade(Student aStudent) {

ArrayList<Integer> studentAnswer =

aStudent.studentAnswer;

**return** getTheRubricSet().apply(studentAnswer);

}

}

CODE:

## 3.7 YOUR CODE

Unless your facilitator arranges another method, copy your Eclipse project to your file system, zip it, and attach it. Please contact your facilitator in advance if you want to request another transmission process (e.g., github).

## 3.8 INSTRUCTOR’S EVALUATION

