# Object Classes for Course Grade Maintenance

Mitsu Ogihara

Department of Computer Science
University of Miami

# Table of Contents

# Defining a New Class for Storing Course Grades

We want to write an application for keeping track of course grades

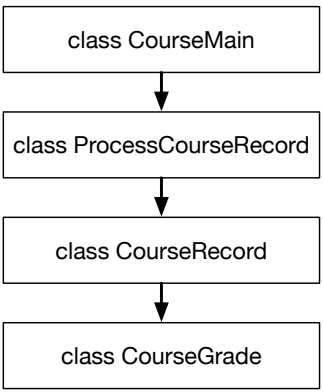
We want to be able to:

- view the list of course grades and their total gpa,
- add new course grades,
- remove course grades,
- read data from a file
- save the data to a file

# Class Organization

- `class CourseGrade`: class for recording one course grade
- `class CourseRecord`: class for maintaining a collection of courses
- `class ProcessCourseRecord`: class for interacting with a `CourseRecord` object
- `class CourseMain`: a simple main class for executing the main execution code in `class ProcessCourseRecord`

# Class Relations




the main method exists in here

the actual work is done in here

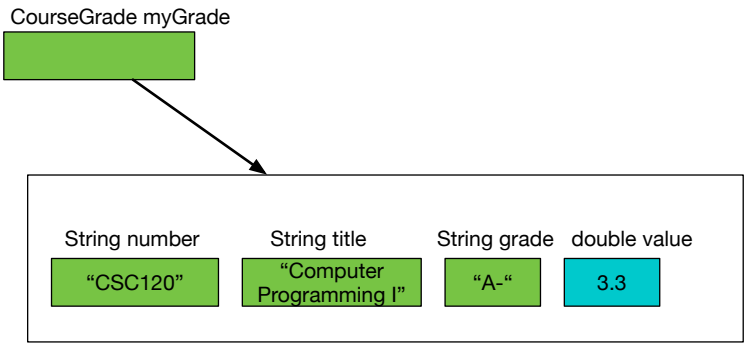
the class for maintaining a collection of course grades

the class for recording information for one course grade

# Class CourseGrade

- Four instance variables
  - `String number`: the course number
  - `String title`: the course title
  - `String grade`: the course letter grade
  - `double value`: the course letter grade converted to a number in 4-point scale
- Constuctor: only one; takes number, title, and grade
- Accessors: `getNumber`, `getTitle`, `getGrade`, `getValue`
- Mutator: `setNumber`, `setTitle`, `SetGrade`
- Special private mutator `convert` that calculates `value` based upon `grade`
  - `private static String[] gradeLetters`: a static list of possible letter grades
  - `private static double[] gradeValues`: an accompanying static list of correspinding 4.0 scale grade values
  - To convert, search for the index of the element in `gradeValues` that matches the grade, and use the value corresponding to the index

# Class CourseGrade



# Code for Class `CourseGrade`

```
1  public class CourseGrade {
2    private String number, title, grade;
3    private double gradeValue;
4
5    public CourseGrade( String number, String title, String grade ) {
6      setNumber( number );
7      setTitle( title );
8      setGrade( grade );
9      gradeValue = convert( grade );
10   }
11   public void setNumber( String number ) {
12     this.number = number;
13   }
14   public void setTitle( String title ) {
15     this.title = title;
16   }
17   public void setGrade( String grade ) {
18     this.grade = grade;
19   }
```

The instance variables

# Code for Class `CourseGrade`

```
1  public class CourseGrade {
2    private String number, title, grade;
3    private double gradeValue;
4
5    public CourseGrade( String number, String title, String grade ) {
6      setNumber( number );
7      setTitle( title );
8      setGrade( grade );
9      gradeValue = convert( grade );
10   }
11   public void setNumber( String number ) {
12     this.number = number;
13   }
14   public void setTitle( String title ) {
15     this.title = title;
16   }
17   public void setGrade( String grade ) {
18     this.grade = grade;
19   }
```

The header of the constructor; takes only the number, the title, and the letter grade

# Code for Class `CourseGrade`

```
1  public class CourseGrade {
2    private String number, title, grade;
3    private double gradeValue;
4
5    public CourseGrade( String number, String title, String grade ) {
6      setNumber( number );
7      setTitle( title );
8      setGrade( grade );
9      gradeValue = convert( grade );
10   }
11   public void setNumber( String number ) {
12     this.number = number;
13   }
14   public void setTitle( String title ) {
15     this.title = title;
16   }
17   public void setGrade( String grade ) {
18     this.grade = grade;
19   }
```

The parameters are assigned to their respective instance variables

# Code for Class `CourseGrade`

```
1  public class CourseGrade {
2    private String number, title, grade;
3    private double gradeValue;
4
5    public CourseGrade( String number, String title, String grade ) {
6      setNumber( number );
7      setTitle( title );
8      setGrade( grade );
9      gradeValue = convert( grade );
10   }
11   public void setNumber( String number ) {
12     this.number = number;
13   }
14   public void setTitle( String title ) {
15     this.title = title;
16   }
17   public void setGrade( String grade ) {
18     this.grade = grade;
19   }
```

For the grade value, use the method `convert` to store the value returned to the variable for the grade value

# Code for Class `CourseGrade`

```
1  public class CourseGrade {
2    private String number, title, grade;
3    private double gradeValue;
4
5    public CourseGrade( String number, String title, String grade ) {
6      setNumber( number );
7      setTitle( title );
8      setGrade( grade );
9      gradeValue = convert( grade );
10   }
11   public void setNumber( String number ) {
12     this.number = number;
13   }
14   public void setTitle( String title ) {
15     this.title = title;
16   }
17   public void setGrade( String grade ) {
18     this.grade = grade;
19   }
```

The method for assigning a value to `number`; use `this`.

# Code for Class `CourseGrade`

```
1  public class CourseGrade {
2    private String number, title, grade;
3    private double gradeValue;
4
5    public CourseGrade( String number, String title, String grade ) {
6      setNumber( number );
7      setTitle( title );
8      setGrade( grade );
9      gradeValue = convert( grade );
10   }
11   public void setNumber( String number ) {
12     this.number = number;
13   }
14   public void setTitle( String title ) {
15     this.title = title;
16   }
17   public void setGrade( String grade ) {
18     this.grade = grade;
19   }
```

The method for assigning a value to `title`; use `this`.

# Code for Class `CourseGrade`

```
1  public class CourseGrade {
2    private String number, title, grade;
3    private double gradeValue;
4
5    public CourseGrade( String number, String title, String grade ) {
6      setNumber( number );
7      setTitle( title );
8      setGrade( grade );
9      gradeValue = convert( grade );
10   }
11   public void setNumber( String number ) {
12     this.number = number;
13   }
14   public void setTitle( String title ) {
15     this.title = title;
16   }
17   public void setGrade( String grade ) {
18     this.grade = grade;
19   }
```

The method for assigning a value to `grade`; use `this`.

# Code for Class `CourseGrade`

```
21    private static String[] gradeLetters = new String[]{
22      "A+", "A", "A-", "B+", "B", "B-", "C+", "C", "C-",
23      "D+", "D", "D-", "F", "I"
24    };
25    private static double[] gradeValues = new double[]{
26      4.0, 4.0, 3.7,  3.3, 3.0, 2.7, 2.3, 2.0, 1.7,
27      1.3, 1.0, 0.7,  0.0, 0.0
28    };
29
30    public double convert( String w ) {
31      for ( int i = 0; i < gradeLetters.length; i ++ ) {
32        if ( w.compareTo( gradeLetters[ i ] ) == 0 ) {
33          return gradeValues[ i ];
34        }
35      }
36      return 0;
37    }
```

The static array of letter grades

# Code for Class `CourseGrade`

```
21    private static String[] gradeLetters = new String[]{
22      "A+", "A", "A-", "B+", "B", "B-", "C+", "C", "C-",
23      "D+", "D", "D-", "F", "I"
24    };
25    private static double[] gradeValues = new double[]{
26      4.0, 4.0, 3.7,  3.3, 3.0, 2.7, 2.3, 2.0, 1.7,
27      1.3, 1.0, 0.7,  0.0, 0.0
28    };
29
30    public double convert( String w ) {
31      for ( int i = 0; i < gradeLetters.length; i ++ ) {
32        if ( w.compareTo( gradeLetters[ i ] ) == 0 ) {
33          return gradeValues[ i ];
34        }
35      }
36      return 0;
37    }
```

The accompanying array of grade values

# Code for Class `CourseGrade`

```
21    private static String[] gradeLetters = new String[]{
22      "A+", "A", "A-", "B+", "B", "B-", "C+", "C", "C-",
23      "D+", "D", "D-", "F", "I"
24    };
25    private static double[] gradeValues = new double[]{
26      4.0, 4.0, 3.7,  3.3, 3.0, 2.7, 2.3, 2.0, 1.7,
27      1.3, 1.0, 0.7,  0.0, 0.0
28    };
29
30    public double convert( String w ) {
31      for ( int i = 0; i < gradeLetters.length; i ++ ) {
32        if ( w.compareTo( gradeLetters[ i ] ) == 0 ) {
33          return gradeValues[ i ];
34        }
35      }
36      return 0;
37    }
```

The method for converting a letter grade to its value; returns a doble

# Code for Class `CourseGrade`

```
21    private static String[] gradeLetters = new String[]{
22      "A+", "A", "A-", "B+", "B", "B-", "C+", "C", "C-",
23      "D+", "D", "D-", "F", "I"
24    };
25    private static double[] gradeValues = new double[]{
26      4.0, 4.0, 3.7,  3.3, 3.0, 2.7, 2.3, 2.0, 1.7,
27      1.3, 1.0, 0.7,  0.0, 0.0
28    };
29
30    public double convert( String w ) {
31      for ( int i = 0; i < gradeLetters.length; i ++ ) {
32        if ( w.compareTo( gradeLetters[ i ] ) == 0 ) {
33          return gradeValues[ i ];
34        }
35      }
36      return 0;
37    }
```

Scan the letter grade array to find a match; if there is a match, return the value element corresponding to the index

# Code for Class `CourseGrade`

```
21    private static String[] gradeLetters = new String[]{
22      "A+", "A", "A-", "B+", "B", "B-", "C+", "C", "C-",
23      "D+", "D", "D-", "F", "I"
24    };
25    private static double[] gradeValues = new double[]{
26      4.0, 4.0, 3.7,  3.3, 3.0, 2.7, 2.3, 2.0, 1.7,
27      1.3, 1.0, 0.7,  0.0, 0.0
28    };
29
30    public double convert( String w ) {
31      for ( int i = 0; i < gradeLetters.length; i ++ ) {
32        if ( w.compareTo( gradeLetters[ i ] ) == 0 ) {
33          return gradeValues[ i ];
34        }
35      }
36      return 0;
37    }
```

Otherwise, return 0

# Code for Class `CourseGrade`

```
39    public String getNumber() {
40      return number;
41    }
42    public String getTitle() {
43      return title;
44    }
45    public String getGrade() {
46      return grade;
47    }
48    public double getValue() {
49      return gradeValue;
50    }
```

Accessor for the course number

# Code for Class `CourseGrade`

```
39    public String getNumber() {
40      return number;
41    }
42    public String getTitle() {
43      return title;
44    }
45    public String getGrade() {
46      return grade;
47    }
48    public double getValue() {
49      return gradeValue;
50    }
```

Accessor for the course title

# Code for Class `CourseGrade`

```
39    public String getNumber() {
40      return number;
41    }
42    public String getTitle() {
43      return title;
44    }
45    public String getGrade() {
46      return grade;
47    }
48    public double getValue() {
49      return gradeValue;
50    }
```

Accessor for the course letter grade

# Code for Class `CourseGrade`

```
39    public String getNumber() {
40      return number;
41    }
42    public String getTitle() {
43      return title;
44    }
45    public String getGrade() {
46      return grade;
47    }
48    public double getValue() {
49      return gradeValue;
50    }
```

Accessor for the course grade value

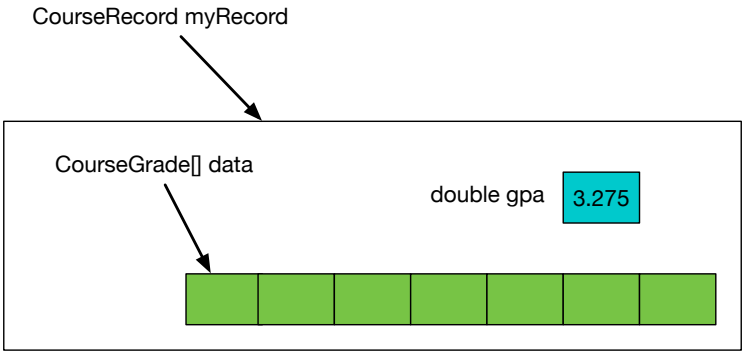
# Class `CourseRecord`

- Two instance variables
  - `CourseGrade[] data`: record the courses as an array
  - `double gpa`: maintain the gap of the courses on the array
- Constructor: there will be only one, which takes no parameter; it initializes the data to an empty array
- Mutators
  - `void merge( CourseGrade[] addition )`: add data from the given array to the collection; update the GPA using `calculateGPA`
  - `void read( File f)`: read data from the specified file; read course grades from the file to an array and then call `merge` to add the array to the data
  - `void remove( int index )`: remove the element at position `index`; update the GPA using `calculateGPA`
  - `calculateGPA`: caluclate and store the GPA value

# Class `CourseRecord` (cont'd)

- Accessors
  - `CourseGrade get( int i )`: return the course grade at position `i`
  - `int size()`: return the number of elements
  - `double getGPA()`: return the GPA
- Special method `void write( File f )`: write the data to the file

# Class CourseRecord



# Code for Class `CourseRecord`

```
1  import java.util.*;
2  import java.io.*;
3
4  public class CourseRecord {
5    private CourseGrade[] data;
6    private double gpa;
7
8    public CourseRecord() {
9      data = new CourseGrade[ 0 ];
10   }
```

Imports

# Code for Class `CourseRecord`

```
1  import java.util.*;
2  import java.io.*;
3
4  public class CourseRecord {
5    private CourseGrade[] data;
6    private double gpa;
7
8    public CourseRecord() {
9      data = new CourseGrade[ 0 ];
10   }
```

Private instance variables

# Code for Class `CourseRecord`

```
1  import java.util.*;
2  import java.io.*;
3
4  public class CourseRecord {
5    private CourseGrade[] data;
6    private double gpa;
7
8    public CourseRecord() {
9      data = new CourseGrade[ 0 ];
10   }
```

Constructor: set `data` to a 0-element array

# Code for Class `CourseRecord` (cont'd)

```
12    public void merge( CourseGrade[] addition ) {
13      CourseGrade[] updated
14          = new CourseGrade[ data.length + addition.length ];
15      for ( int i = 0; i < data.length; i ++ ) {
16        updated[ i ] = data[ i ];
17      }
18      for ( int i = 0; i < addition.length; i ++ ) {
19        updated[ i + data.length ] = addition[ i ];
20      }
21      data = updated;
22      calculateGPA();
23    }
```

The `merge` method; takes an array of `CourseGrade` objects as the parameter

# Code for Class `CourseRecord` (cont'd)

```
12    public void merge( CourseGrade[] addition ) {
13      CourseGrade[] updated
14          = new CourseGrade[ data.length + addition.length ];
15      for ( int i = 0; i < data.length; i ++ ) {
16        updated[ i ] = data[ i ];
17      }
18      for ( int i = 0; i < addition.length; i ++ ) {
19        updated[ i + data.length ] = addition[ i ];
20      }
21      data = updated;
22      calculateGPA();
23    }
```

Create a new array to hold the current and the additional elements together

# Code for Class `CourseRecord` (cont'd)

```
12    public void merge( CourseGrade[] addition ) {
13      CourseGrade[] updated
14          = new CourseGrade[ data.length + addition.length ];
15      for ( int i = 0; i < data.length; i ++ ) {
16        updated[ i ] = data[ i ];
17      }
18      for ( int i = 0; i < addition.length; i ++ ) {
19        updated[ i + data.length ] = addition[ i ];
20      }
21      data = updated;
22      calculateGPA();
23    }
```

Copy all the elements from the current array to the new array

# Code for Class `CourseRecord` (cont'd)

```
12    public void merge( CourseGrade[] addition ) {
13      CourseGrade[] updated
14          = new CourseGrade[ data.length + addition.length ];
15      for ( int i = 0; i < data.length; i ++ ) {
16        updated[ i ] = data[ i ];
17      }
18      for ( int i = 0; i < addition.length; i ++ ) {
19        updated[ i + data.length ] = addition[ i ];
20      }
21      data = updated;
22      calculateGPA();
23    }
```

Copy all the elements from the addition to the new array after the current elements

# Code for Class `CourseRecord` (cont'd)

```
12    public void merge( CourseGrade[] addition ) {
13      CourseGrade[] updated
14          = new CourseGrade[ data.length + addition.length ];
15      for ( int i = 0; i < data.length; i ++ ) {
16        updated[ i ] = data[ i ];
17      }
18      for ( int i = 0; i < addition.length; i ++ ) {
19        updated[ i + data.length ] = addition[ i ];
20      }
21      data = updated;
22      calculateGPA();
23    }
```

Change the current to the new data

# Code for Class `CourseRecord` (cont'd)

```
12    public void merge( CourseGrade[] addition ) {
13      CourseGrade[] updated
14          = new CourseGrade[ data.length + addition.length ];
15      for ( int i = 0; i < data.length; i ++ ) {
16        updated[ i ] = data[ i ];
17      }
18      for ( int i = 0; i < addition.length; i ++ ) {
19        updated[ i + data.length ] = addition[ i ];
20      }
21      data = updated;
22      calculateGPA();
23    }
```

Call `calculateGPA` to update the GPA value

# Code for Class `CourseRecord` (cont'd)

```
25    private void calculateGPA() {
26      gpa = 0;
27      for ( CourseGrade g : data ) {
28        gpa += g.getValue() / size();
29      }
30    }
```

The method for calculating GPA

# Code for Class `CourseRecord` (cont'd)

```
25    private void calculateGPA() {
26      gpa = 0;
27      for ( CourseGrade g : data ) {
28        gpa += g.getValue() / size();
29      }
30    }
```

Use a cumulative algorithm: set gpa to 0

# Code for Class `CourseRecord` (cont'd)

```
25    private void calculateGPA() {
26      gpa = 0;
27      for ( CourseGrade g : data ) {
28        gpa += g.getValue() / size();
29      }
30    }
```

Use the "for each" loop on the array, add to gpa, the grade value divided by the number of courses

# Code for Class `CourseRecord` (cont'd)

```
32    public void read( File f ) throws FileNotFoundException {
33      Scanner fileScanner = new Scanner( f );
34      int size = Integer.parseInt( fileScanner.nextLine() );
35      CourseGrade[] addition = new CourseGrade[ size ];
36      for ( int i = 0; i < size; i ++ ) {
37        addition[ i ] = new CourseGrade( fileScanner.nextLine(),
38          fileScanner.nextLine(), fileScanner.nextLine() );
39      }
40      fileScanner.close();
41      merge( addition );
42    }
```

The `read` method; throws `FileNotFoundException`

# Code for Class `CourseRecord` (cont'd)

```
32    public void read( File f ) throws FileNotFoundException {
33      Scanner fileScanner = new Scanner( f );
34      int size = Integer.parseInt( fileScanner.nextLine() );
35      CourseGrade[] addition = new CourseGrade[ size ];
36      for ( int i = 0; i < size; i ++ ) {
37        addition[ i ] = new CourseGrade( fileScanner.nextLine(),
38          fileScanner.nextLine(), fileScanner.nextLine() );
39      }
40      fileScanner.close();
41      merge( addition );
42    }
```

Create a Scanner out of the File given

# Code for Class `CourseRecord` (cont'd)

```
32    public void read( File f ) throws FileNotFoundException {
33      Scanner fileScanner = new Scanner( f );
34      int size = Integer.parseInt( fileScanner.nextLine() );
35      CourseGrade[] addition = new CourseGrade[ size ];
36      for ( int i = 0; i < size; i ++ ) {
37        addition[ i ] = new CourseGrade( fileScanner.nextLine(),
38          fileScanner.nextLine(), fileScanner.nextLine() );
39      }
40      fileScanner.close();
41      merge( addition );
42    }
```

Parse the very first line of the data as an integer using `Integer.parseInt`; this is the number of elements stored in the file

# Code for Class `CourseRecord` (cont'd)

```
32    public void read( File f ) throws FileNotFoundException {
33      Scanner fileScanner = new Scanner( f );
34      int size = Integer.parseInt( fileScanner.nextLine() );
35      CourseGrade[] addition = new CourseGrade[ size ];
36      for ( int i = 0; i < size; i ++ ) {
37        addition[ i ] = new CourseGrade( fileScanner.nextLine(),
38          fileScanner.nextLine(), fileScanner.nextLine() );
39      }
40      fileScanner.close();
41      merge( addition );
42    }
```

Create an array to store the elements to be read from the file

# Code for Class `CourseRecord` (cont'd)

```
32    public void read( File f ) throws FileNotFoundException {
33      Scanner fileScanner = new Scanner( f );
34      int size = Integer.parseInt( fileScanner.nextLine() );
35      CourseGrade[] addition = new CourseGrade[ size ];
36      for ( int i = 0; i < size; i ++ ) {
37        addition[ i ] = new CourseGrade( fileScanner.nextLine(),
38          fileScanner.nextLine(), fileScanner.nextLine() );
39      }
40      fileScanner.close();
41      merge( addition );
42    }
```

Use a for loop to fill the array

# Code for Class `CourseRecord` (cont'd)

```
32    public void read( File f ) throws FileNotFoundException {
33      Scanner fileScanner = new Scanner( f );
34      int size = Integer.parseInt( fileScanner.nextLine() );
35      CourseGrade[] addition = new CourseGrade[ size ];
36      for ( int i = 0; i < size; i ++ ) {
37        addition[ i ] = new CourseGrade( fileScanner.nextLine(),
38          fileScanner.nextLine(), fileScanner.nextLine() );
39      }
40      fileScanner.close();
41      merge( addition );
42    }
```

Read the remainder of the lines in triples to create the elements

# Code for Class `CourseRecord` (cont'd)

```
32    public void read( File f ) throws FileNotFoundException {
33      Scanner fileScanner = new Scanner( f );
34      int size = Integer.parseInt( fileScanner.nextLine() );
35      CourseGrade[] addition = new CourseGrade[ size ];
36      for ( int i = 0; i < size; i ++ ) {
37        addition[ i ] = new CourseGrade( fileScanner.nextLine(),
38          fileScanner.nextLine(), fileScanner.nextLine() );
39      }
40      fileScanner.close();
41      merge( addition );
42    }
```

Close the scanner; this `close` method is needed since we may be reading from and writing to the same file

# Code for Class `CourseRecord` (cont'd)

```
32    public void read( File f ) throws FileNotFoundException {
33      Scanner fileScanner = new Scanner( f );
34      int size = Integer.parseInt( fileScanner.nextLine() );
35      CourseGrade[] addition = new CourseGrade[ size ];
36      for ( int i = 0; i < size; i ++ ) {
37        addition[ i ] = new CourseGrade( fileScanner.nextLine(),
38          fileScanner.nextLine(), fileScanner.nextLine() );
39      }
40      fileScanner.close();
41      merge( addition );
42    }
```

Call the merge method

# Code for Class `CourseRecord` (cont'd)

```
44    public void remove( int position ) {
45      CourseGrade[] update = new CourseGrade[ data.length - 1 ];
46      for ( int i = 0; i < position; i ++ ) {
47        update[ i ] = data[ i ];
48      }
49      for ( int i = position + 1; i < data.length; i ++ ) {
50        update[ i - 1 ] = data[ i ];
51      }
52      data = update;
53      calculateGPA();
54    }
```

The remove method

# Code for Class `CourseRecord` (cont'd)

```
44    public void remove( int position ) {
45      CourseGrade[] update = new CourseGrade[ data.length - 1 ];
46      for ( int i = 0; i < position; i ++ ) {
47        update[ i ] = data[ i ];
48      }
49      for ( int i = position + 1; i < data.length; i ++ ) {
50        update[ i - 1 ] = data[ i ];
51      }
52      data = update;
53      calculateGPA();
54    }
```

Use the already known method for removing an element from an array

# Code for Class `CourseRecord` (cont'd)

```
44    public void remove( int position ) {
45      CourseGrade[] update = new CourseGrade[ data.length - 1 ];
46      for ( int i = 0; i < position; i ++ ) {
47        update[ i ] = data[ i ];
48      }
49      for ( int i = position + 1; i < data.length; i ++ ) {
50        update[ i - 1 ] = data[ i ];
51      }
52      data = update;
53      calculateGPA();
54    }
```

Recalculate GPA

# Code for Class `CourseRecord` (cont'd)

```
56    public CourseGrade get( int i ) {
57      return data[ i ];
58    }
59    public int size() {
60      return data.length;
61    }
62    public double getGPA() {
63      return gpa;
64    }
```

The method that returns a `CourseGrade` at a given position

# Code for Class `CourseRecord` (cont'd)

```
56    public CourseGrade get( int i ) {
57      return data[ i ];
58    }
59    public int size() {
60      return data.length;
61    }
62    public double getGPA() {
63      return gpa;
64    }
```

Return the number of data stored

# Code for Class `CourseRecord` (cont'd)

```
56    public CourseGrade get( int i ) {
57      return data[ i ];
58    }
59    public int size() {
60      return data.length;
61    }
62    public double getGPA() {
63      return gpa;
64    }
```

The method that returns the GPA

# Code for Class `CourseRecord` (cont'd)

```
66    public void write( File f ) throws FileNotFoundException {
67      PrintStream stream = new PrintStream( f );
68      stream.println( data.length );
69      for ( CourseGrade g : data ) {
70        stream.println( g.getNumber() );
71        stream.println( g.getTitle() );
72        stream.println( g.getGrade() );
73      }
74      stream.close();
75    }
```

The method for writing to file; may throw `FileNotFoundException`

# Code for Class `CourseRecord` (cont'd)

```
66    public void write( File f ) throws FileNotFoundException {
67      PrintStream stream = new PrintStream( f );
68      stream.println( data.length );
69      for ( CourseGrade g : data ) {
70        stream.println( g.getNumber() );
71        stream.println( g.getTitle() );
72        stream.println( g.getGrade() );
73      }
74      stream.close();
75    }
```

Create a `PrintStream` object out of the given file

# Code for Class `CourseRecord` (cont'd)

```
66    public void write( File f ) throws FileNotFoundException {
67      PrintStream stream = new PrintStream( f );
68      stream.println( data.length );
69      for ( CourseGrade g : data ) {
70        stream.println( g.getNumber() );
71        stream.println( g.getTitle() );
72        stream.println( g.getGrade() );
73      }
74      stream.close();
75    }
```

Write the size information

# Code for Class `CourseRecord` (cont'd)

```
66    public void write( File f ) throws FileNotFoundException {
67      PrintStream stream = new PrintStream( f );
68      stream.println( data.length );
69      for ( CourseGrade g : data ) {
70        stream.println( g.getNumber() );
71        stream.println( g.getTitle() );
72        stream.println( g.getGrade() );
73      }
74      stream.close();
75    }
```

Use the "for each" loop to go through the elements of the collection and write their three main instance variables to the file, one value per line

# Code for Class `CourseRecord` (cont'd)

```
66    public void write( File f ) throws FileNotFoundException {
67      PrintStream stream = new PrintStream( f );
68      stream.println( data.length );
69      for ( CourseGrade g : data ) {
70        stream.println( g.getNumber() );
71        stream.println( g.getTitle() );
72        stream.println( g.getGrade() );
73      }
74      stream.close();
75    }
```

Close the `PrintStram`

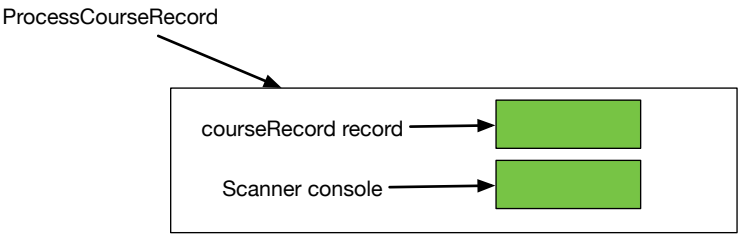
# Class `ProcessCourseRecord`

The class for handling course record modification and examination

- Prompt the user to select an action from viewing data, adding data, removing data, reading data, writing data, and quitting
- Perform the selected task
- Repeat this until the user selects to quit

Use instance variables: `CourseRecord record` and `Scanner console`

# Class CourseRecord



# Code for Class `ProcessCourseRecord`

```
1  import java.util.*;
2  import java.io.*;
3
4  public class ProcessCourseRecord {
5    private CourseRecord record;
6    private Scanner console;
7
8    public ProcessCourseRecord() {
9      record = new CourseRecord();
10     console = new Scanner( System.in );
11   }
```

Instance variables

# Code for Class `ProcessCourseRecord`

```
1  import java.util.*;
2  import java.io.*;
3
4  public class ProcessCourseRecord {
5    private CourseRecord record;
6    private Scanner console;
7
8    public ProcessCourseRecord() {
9      record = new CourseRecord();
10     console = new Scanner( System.in );
11   }
```

Constructor; call the constuctor of `CourseRecord`

# Code for Class `ProcessCourseRecord` (cont'd)

```
13    public void exec() throws FileNotFoundException {
14      int action = 1;
15      while ( action != 5 ) {
16        System.out.println( "====== Course Record Program" );
17        System.out.println( "====== Choose Action" );
18        System.out.println( "0. view" );
19        System.out.println( "1. add" );
20        System.out.println( "2. remove" );
21        System.out.println( "3. read" );
22        System.out.println( "4. write" );
23        System.out.println( "5. quit" );
24        System.out.print( "Enter choice: " );
25        action = Integer.parseInt( console.nextLine() );
26        if ( action == 0 ) { view(); }
27        else if ( action == 1 ) { add(); }
28        else if ( action == 2 ) { remove(); }
29        else if ( action == 3 ) { read(); }
30        else if ( action == 4 ) { write(); }
31      }
32    }
```

The method for performing the actions

# Code for Class `ProcessCourseRecord` (cont'd)

```
13    public void exec() throws FileNotFoundException {
14      int action = 1;
15      while ( action != 5 ) {
16        System.out.println( "====== Course Record Program" );
17        System.out.println( "====== Choose Action" );
18        System.out.println( "0. view" );
19        System.out.println( "1. add" );
20        System.out.println( "2. remove" );
21        System.out.println( "3. read" );
22        System.out.println( "4. write" );
23        System.out.println( "5. quit" );
24        System.out.print( "Enter choice: " );
25        action = Integer.parseInt( console.nextLine() );
26        if ( action == 0 ) { view(); }
27        else if ( action == 1 ) { add(); }
28        else if ( action == 2 ) { remove(); }
29        else if ( action == 3 ) { read(); }
30        else if ( action == 4 ) { write(); }
31      }
32    }
```

Use this variable to record the action to perfomr as an integer

# Code for Class `ProcessCourseRecord` (cont'd)

```
13    public void exec() throws FileNotFoundException {
14      int action = 1;
15      while ( action != 5 ) {
16        System.out.println( "====== Course Record Program" );
17        System.out.println( "====== Choose Action" );
18        System.out.println( "0. view" );
19        System.out.println( "1. add" );
20        System.out.println( "2. remove" );
21        System.out.println( "3. read" );
22        System.out.println( "4. write" );
23        System.out.println( "5. quit" );
24        System.out.print( "Enter choice: " );
25        action = Integer.parseInt( console.nextLine() );
26        if ( action == 0 ) { view(); }
27        else if ( action == 1 ) { add(); }
28        else if ( action == 2 ) { remove(); }
29        else if ( action == 3 ) { read(); }
30        else if ( action == 4 ) { write(); }
31      }
32    }
```

The while loop is repeated the action chosen is 5; since the initial value is 1, the while loop body is executed at least once

# Code for Class `ProcessCourseRecord` (cont'd)

```
13    public void exec() throws FileNotFoundException {
14      int action = 1;
15      while ( action != 5 ) {
16        System.out.println( "====== Course Record Program" );
17        System.out.println( "====== Choose Action" );
18        System.out.println( "0. view" );
19        System.out.println( "1. add" );
20        System.out.println( "2. remove" );
21        System.out.println( "3. read" );
22        System.out.println( "4. write" );
23        System.out.println( "5. quit" );
24        System.out.print( "Enter choice: " );
25        action = Integer.parseInt( console.nextLine() );
26        if ( action == 0 ) { view(); }
27        else if ( action == 1 ) { add(); }
28        else if ( action == 2 ) { remove(); }
29        else if ( action == 3 ) { read(); }
30        else if ( action == 4 ) { write(); }
31      }
32    }
```

Prompt the user to enter action number; use `Integer.parseInt` to interpret the input as an integer

# Code for Class `ProcessCourseRecord` (cont'd)

```
13    public void exec() throws FileNotFoundException {
14      int action = 1;
15      while ( action != 5 ) {
16        System.out.println( "====== Course Record Program" );
17        System.out.println( "====== Choose Action" );
18        System.out.println( "0. view" );
19        System.out.println( "1. add" );
20        System.out.println( "2. remove" );
21        System.out.println( "3. read" );
22        System.out.println( "4. write" );
23        System.out.println( "5. quit" );
24        System.out.print( "Enter choice: " );
25        action = Integer.parseInt( console.nextLine() );
26        if ( action == 0 ) { view(); }
27        else if ( action == 1 ) { add(); }
28        else if ( action == 2 ) { remove(); }
29        else if ( action == 3 ) { read(); }
30        else if ( action == 4 ) { write(); }
31      }
32    }
```

Perform the action chosen; nothing happens if an invalid choice is made

# Code for Class `ProcessCourseRecord` (cont'd)

```
34    public void view() {
35      for ( int i = 0; i < record.size(); i ++ ) {
36        System.out.printf( "%03d:\t%s\t%s\t%s\t%4.2f%n",
37            i, record.get( i ).getNumber(), record.get( i ).getTitle(),
38            record.get( i ).getGrade(), record.get( i ).getValue() );
39      }
40      System.out.printf( "GPA: %4.2f%n", record.getGPA() );
41    }
```

Method for viewing the data

# Code for Class `ProcessCourseRecord` (cont'd)

```
34    public void view() {
35      for ( int i = 0; i < record.size(); i ++ ) {
36        System.out.printf( "%03d:\t%s\t%s\t%s\t%4.2f%n",
37            i, record.get( i ).getNumber(), record.get( i ).getTitle(),
38            record.get( i ).getGrade(), record.get( i ).getValue() );
39      }
40      System.out.printf( "GPA: %4.2f%n", record.getGPA() );
41    }
```

Use a for loop to go through the elements

# Code for Class `ProcessCourseRecord` (cont'd)

```
34    public void view() {
35      for ( int i = 0; i < record.size(); i ++ ) {
36        System.out.printf( "%03d:\t%s\t%s\t%s\t%4.2f%n",
37            i, record.get( i ).getNumber(), record.get( i ).getTitle(),
38            record.get( i ).getGrade(), record.get( i ).getValue() );
39      }
40      System.out.printf( "GPA: %4.2f%n", record.getGPA() );
41    }
```

Use some formatting to print the data along with their index values

# Code for Class `ProcessCourseRecord` (cont'd)

```
34    public void view() {
35      for ( int i = 0; i < record.size(); i ++ ) {
36        System.out.printf( "%03d:\t%s\t%s\t%s\t%4.2f%n",
37            i, record.get( i ).getNumber(), record.get( i ).getTitle(),
38            record.get( i ).getGrade(), record.get( i ).getValue() );
39      }
40      System.out.printf( "GPA: %4.2f%n", record.getGPA() );
41    }
```

Print the GPA

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

Method for adding data

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

Prompt the user to receive the number of elements to add; using `Integer.parseInt` here, too

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

If the number is positive perform the addition

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

Create an array of the declared number of elements

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

Use a for loop to receive elements

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

Prompt the user to enter data for the three variables need for the `CourseGrade` constructor

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

Receive the three values

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

Create a new object and store it to the current position in the array

# Code for Class `ProcessCourseRecord` (cont'd)

```
43    public void add() {
44      System.out.print( "How many?: " );
45      int count = Integer.parseInt( console.nextLine() );
46      if ( count > 0 ) {
47        CourseGrade[] addition = new CourseGrade[ count ];
48        for ( int i = 0; i < count; i ++ ) {
49          System.out.printf( "----- Grade No. %d%n", i );
50          System.out.print( "Enter number: " );
51          String number = console.nextLine();
52          System.out.print( "Enter title: " );
53          String title = console.nextLine();
54          System.out.print( "Enter grade: " );
55          String grade = console.nextLine();
56          addition[ i ] = new CourseGrade( number, title, grade );
57        }
58        record.merge( addition );
59      }
```

Call `merge` to add the data

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Method for removing one data element

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Prompt the user to receive index

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Call the remove method

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Method for reading data

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Prompt the user to enter a file path

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Call the read method on a new `File` create from the String entered by the user

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Method for writing data

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Prompt the user to enter a file path

# Code for Class `ProcessCourseRecord` (cont'd)

```
62    public void remove() {
63      System.out.print( "Enter index: " );
64      int index = Integer.parseInt( console.nextLine() );
65      if ( index >=0 && index <= record.size() - 1 ) {
66        record.remove( index );
67      }
68    }
69
70    public void read() throws FileNotFoundException {
71      System.out.print( "Enter a file path: " );
72      record.read( new File( console.nextLine() ) );
73    }
74
75    public void write() throws FileNotFoundException {
76      System.out.print( "Enter a file path: " );
77      record.write( new File( console.nextLine() ) );
78    }
```

Call the write method on a new `File` create from the String entered by the user

# Code for Class `CourseMain`

The method for executing the `exec` method of the class `ProcessCourseRecord`

```
1  import java.io.*;
2  public class CourseMain {
3    public static void main( String[] args )
4        throws FileNotFoundException {
5      ( new ProcessCourseRecord() ).exec();
6    }
7  }
```

The main method has a single line to execute
Create a new `ProcessCourseRecord` object and call the method `exec` on it