```java
/**
 * @
 */

import java.util.Arrays;
import java.util.Scanner;

/**
 * This is the main class for the university data management app.
 *
 * @author Joshua Prout jnp207@exeter.ac.uk
 */

public class University {

    private ModuleDescriptor[] moduleDescriptors;

    private Student[] students;

    private Module[] modules;

    /**
     * Constructor for university class.
     * <p>
     * Takes no parameters, arrays are filled later
     */
    public University() {
        ModuleDescriptor[] blankDescriptor = {};
        Student[] blankStudents = {};
        Module[] blankModules = {};

        this.moduleDescriptors = blankDescriptor;
        this.students = blankStudents;
        this.modules = blankModules;
    }

    /**
     * Adds new module descriptor to module descriptors array.
     *
     * @param NewModuleDesc New module descriptor to add to array
     */
    public void addModuleDescriptor(ModuleDescriptor newModuleDesc) {
        ModuleDescriptor[] newModuleDescs = Arrays.copyOf(this.moduleDescriptors,
        this.moduleDescriptors.length + 1);
        newModuleDescs[newModuleDescs.length - 1] = newModuleDesc;
        this.moduleDescriptors = newModuleDescs;
    }

    /**
     * Adds new student to students array.
     *
     * @param Student newStudent, adds new student to array
     */
    public void addStudent(Student newStudent) {
        Student[] newStudents = Arrays.copyOf(this.students, this.students.length + 1);
        newStudents[newStudents.length - 1] = newStudent;
        this.students = newStudents;
    }

    /**
     * Adds new module to the array
     *
     * @param Module NewModule, adds new module to array
     */
    public void addModule(Module newModule) {
        Module[] newModules = Arrays.copyOf(this.modules, this.modules.length + 1);
        newModules[newModules.length - 1] = newModule;
        this.modules = newModules;
    }
```

```java

    /**
     * Returns number of students in the university system
     *
     * @return numberOfStudents The number of students registered in the system.
     */
    public int getTotalNumberStudents() {

        return students.length;
    }

    /**
     * Returns student with the highest GPA
     *
     * @return The student with the highest GPA.
     */
    public Student getBestStudent() {
        Student highestStudent = students[0];
        for (Student student : students) {
            if (student.getGpa() > highestStudent.getGpa()) {
                highestStudent = student;
            }
        }
        return highestStudent;
    }

    /**
     * @return The module with the highest average score.
     */
    public Module getBestModule() {
        Module highestModule = modules[0];
        for (Module module : modules) {
            if (module.getFinalAverageGrade() > highestModule.getFinalAverageGrade()) {
                highestModule = module;
            }
        }

        return highestModule;
    }

    /**
     * Checks if ID has already been taken by another student in the system
     *
     * @return true if ID is not available
     */
    public boolean checkId(int theId) {

        boolean found = false;
        for (Student student : students) {
            int id = student.getId();
            if (id == theId) {
                found = true;
                break;
            }
        }
        return found;
    }

    /**
     * @return the student with the given ID
     */
    public Student getStudentById(int theId) {

        for (Student student : students) {
            if (student.getId() == theId) {
                return student;
            }
        }
        return null;
```

```
138         }
139
140         /**
141          * Calls the updateAboveAverage method for each student.
142          * <p>
143          * This will compare their score for each student record against the average for
144          * the module
145          */
146         public void updateAboveAverage() {
147             for (Student student : students) {
148                 student.setAboveAverage();
149             }
150         }
151
152         /**
153          * Creates university object, and inputs and outputs data.
154          * <p>
155          * Creates a university object, to which the data will be added to
156          * Creates Module Descriptors and adds to array, creates Students and adds to array,
157          * creates modules and adds to array, then creates and adds student records to the
158          * module and student objects
159          */
160         public static void main(String[] args) {
161             // TODO - needs to be implemented
162             University uni = new University();
163
164             // Creates Module Descriptors
165             ModuleDescriptor ECM0002 = new ModuleDescriptor("ECM0002", "Real World
                Mathematics", new double[]{0.1, 0.3, 0.6});
166             ModuleDescriptor ECM1400 = new ModuleDescriptor("ECM1400", "Programming", new
                double[]{0.25, 0.25, 0.25, 0.25});
167             ModuleDescriptor ECM1406 = new ModuleDescriptor("ECM1406", "Data Structures",
                new double[]{0.25, 0.25, 0.5});
168             ModuleDescriptor ECM1410 = new ModuleDescriptor("ECM1410", "Object-Oriented
                Programming", new double[]{0.2, 0.3, 0.5});
169             ModuleDescriptor BEM2027 = new ModuleDescriptor("BEM2027", "Information
                Systems", new double[]{0.1, 0.3, 0.3, 0.3});
170             ModuleDescriptor PHY2023 = new ModuleDescriptor("PHY2023", "Thermal Physics",
                new double[]{0.4, 0.6});
171
172             // Adds Module Descriptors to array
173             uni.addModuleDescriptor(ECM0002);
174             uni.addModuleDescriptor(ECM1400);
175             uni.addModuleDescriptor(ECM1406);
176             uni.addModuleDescriptor(ECM1410);
177             uni.addModuleDescriptor(BEM2027);
178             uni.addModuleDescriptor(PHY2023);
179
180             // Creates students and adds to array
181             uni.addStudent(new Student(1000, "Ana", 'F', uni));
182             uni.addStudent(new Student(1001, "Oliver", 'M', uni));
183             uni.addStudent(new Student(1002, "Mary", 'F', uni));
184             uni.addStudent(new Student(1003, "John", 'M', uni));
185             uni.addStudent(new Student(1004, "Noah", 'M', uni));
186             uni.addStudent(new Student(1005, "Chico", 'M', uni));
187             uni.addStudent(new Student(1006, "Maria", 'F', uni));
188             uni.addStudent(new Student(1007, "Mark", 'X', uni));
189             uni.addStudent(new Student(1008, "Lia", 'F', uni));
190             uni.addStudent(new Student(1009, "Rachel", 'F', uni));
191
192             // Adds student records
193
194             Student student;
195             Module module;
196             StudentRecord record;
197
198             // ECM1400 2019 term 1
199             module = new Module(2019, (byte) 1, ECM1400);
200             uni.addModule(module);
```

```
201
202             student = uni.getStudentById(1000);
203             record = new StudentRecord(student, module, new double[]{9, 10, 10, 10});
204             module.addRecord(record);
205             student.addRecord(record);
206             student = uni.getStudentById(1001);
207             record = new StudentRecord(student, module, new double[]{8, 8, 8, 9});
208             module.addRecord(record);
209             student.addRecord(record);
210             student = uni.getStudentById(1002);
211             record = new StudentRecord(student, module, new double[]{5, 5, 6, 5});
212             module.addRecord(record);
213             student.addRecord(record);
214             student = uni.getStudentById(1003);
215             record = new StudentRecord(student, module, new double[]{6, 4, 7, 9});
216             module.addRecord(record);
217             student.addRecord(record);
218             student = uni.getStudentById(1004);
219             record = new StudentRecord(student, module, new double[]{10, 9, 10, 9});
220             module.addRecord(record);
221             student.addRecord(record);
222
223             //PHY 2023 2019 1
224             module = new Module(2019, (byte) 1, PHY2023);
225             uni.addModule(module);
226
227             student = uni.getStudentById(1005);
228             record = new StudentRecord(student, module, new double[]{9, 9});
229             module.addRecord(record);
230             student.addRecord(record);
231             student = uni.getStudentById(1006);
232             record = new StudentRecord(student, module, new double[]{6, 9});
233             module.addRecord(record);
234             student.addRecord(record);
235             student = uni.getStudentById(1007);
236             record = new StudentRecord(student, module, new double[]{5, 6});
237             module.addRecord(record);
238             student.addRecord(record);
239             student = uni.getStudentById(1008);
240             record = new StudentRecord(student, module, new double[]{9, 7});
241             module.addRecord(record);
242             student.addRecord(record);
243             student = uni.getStudentById(1009);
244             record = new StudentRecord(student, module, new double[]{8, 5});
245             module.addRecord(record);
246             student.addRecord(record);
247
248             //BEM2027 2019 2
249             module = new Module(2019, (byte) 2, BEM2027);
250             uni.addModule(module);
251
252             student = uni.getStudentById(1000);
253             record = new StudentRecord(student, module, new double[]{10, 10, 9.5, 10});
254             module.addRecord(record);
255             student.addRecord(record);
256             student = uni.getStudentById(1001);
257             record = new StudentRecord(student, module, new double[]{7, 8.5, 8.2, 8});
258             module.addRecord(record);
259             student.addRecord(record);
260             student = uni.getStudentById(1002);
261             record = new StudentRecord(student, module, new double[]{6.5, 7.0, 5.5, 8.5});
262             module.addRecord(record);
263             student.addRecord(record);
264             student = uni.getStudentById(1003);
265             record = new StudentRecord(student, module, new double[]{5.5, 5, 6.5, 7});
266             module.addRecord(record);
267             student.addRecord(record);
268             student = uni.getStudentById(1004);
269             record = new StudentRecord(student, module, new double[]{7, 5, 8, 6});
```

```
            module.addRecord(record);
            student.addRecord(record);

            //ECM1400 2019 2
            module = new Module(2019, (byte) 2, ECM1400);
            uni.addModule(module);

            student = uni.getStudentById(1005);
            record = new StudentRecord(student, module, new double[]{9, 10, 10, 10});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1006);
            record = new StudentRecord(student, module, new double[]{8, 8, 8, 9});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1007);
            record = new StudentRecord(student, module, new double[]{5, 5, 6, 5});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1008);
            record = new StudentRecord(student, module, new double[]{6, 4, 7, 9});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1009);
            record = new StudentRecord(student, module, new double[]{10, 9, 8, 9});
            module.addRecord(record);
            student.addRecord(record);


            // ECM1406 2020 1
            module = new Module(2020, (byte) 1, ECM1406);
            uni.addModule(module);

            student = uni.getStudentById(1000);
            record = new StudentRecord(student, module, new double[]{10, 10, 10});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1001);
            record = new StudentRecord(student, module, new double[]{8, 7.5, 7.5});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1002);
            record = new StudentRecord(student, module, new double[]{9, 7, 7});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1003);
            record = new StudentRecord(student, module, new double[]{9, 8, 7});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1004);
            record = new StudentRecord(student, module, new double[]{2, 7, 7});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1005);
            record = new StudentRecord(student, module, new double[]{10, 10, 10});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1006);
            record = new StudentRecord(student, module, new double[]{8, 7.5, 7.5});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1007);
            record = new StudentRecord(student, module, new double[]{10, 10, 10});
            module.addRecord(record);
            student.addRecord(record);
            student = uni.getStudentById(1008);
            record = new StudentRecord(student, module, new double[]{9, 8, 7});
            module.addRecord(record);
            student.addRecord(record);
```

```
339             student = uni.getStudentById(1009);
340             record = new StudentRecord(student, module, new double[]{8, 9, 10});
341             module.addRecord(record);
342             student.addRecord(record);
343 
344             //ECM1410 2020 1
345             module = new Module(2020, (byte) 1, ECM1410);
346             uni.addModule(module);
347 
348             student = uni.getStudentById(1000);
349             record = new StudentRecord(student, module, new double[]{10, 9, 10});
350             module.addRecord(record);
351             student.addRecord(record);
352             student = uni.getStudentById(1001);
353             record = new StudentRecord(student, module, new double[]{8.5, 9, 7.5});
354             module.addRecord(record);
355             student.addRecord(record);
356             student = uni.getStudentById(1002);
357             record = new StudentRecord(student, module, new double[]{10, 10, 5.5});
358             module.addRecord(record);
359             student.addRecord(record);
360             student = uni.getStudentById(1003);
361             record = new StudentRecord(student, module, new double[]{7, 7, 7});
362             module.addRecord(record);
363             student.addRecord(record);
364             student = uni.getStudentById(1004);
365             record = new StudentRecord(student, module, new double[]{5, 6, 10});
366             module.addRecord(record);
367             student.addRecord(record);
368 
369             //ECM0002 2020 2
370             module = new Module(2020, (byte) 2, ECM0002);
371             uni.addModule(module);
372             student = uni.getStudentById(1005);
373             record = new StudentRecord(student, module, new double[]{8, 9, 8});
374             module.addRecord(record);
375             student.addRecord(record);
376             student = uni.getStudentById(1006);
377             record = new StudentRecord(student, module, new double[]{6.5, 9, 9.5});
378             module.addRecord(record);
379             student.addRecord(record);
380             student = uni.getStudentById(1007);
381             record = new StudentRecord(student, module, new double[]{8.5, 10, 8.5});
382             module.addRecord(record);
383             student.addRecord(record);
384             student = uni.getStudentById(1008);
385             record = new StudentRecord(student, module, new double[]{7.5, 8, 10});
386             module.addRecord(record);
387             student.addRecord(record);
388             student = uni.getStudentById(1009);
389             record = new StudentRecord(student, module, new double[]{10, 6, 10});
390             module.addRecord(record);
391             student.addRecord(record);
392 
393             // Calculates if each student is above average
394             uni.updateAboveAverage();
395 
396             //Prints number of students
397             int numberOfStudents = uni.getTotalNumberStudents();
398             System.out.println("Number of students " + numberOfStudents);
399 
400             //Prints best student
401             Student bestStudent = uni.getBestStudent();
402             System.out.println("Best student " + bestStudent.getName());
403 
404             //Prints best student's transcript
405             System.out.println(bestStudent.printTranscript());
406 
407             //Prints best module
```

```
408             Module bestModule = uni.getBestModule();
409             System.out.println("Best Module " + bestModule.getModuleCode());
410         }
411     }
412
```