物件導向軟體工程

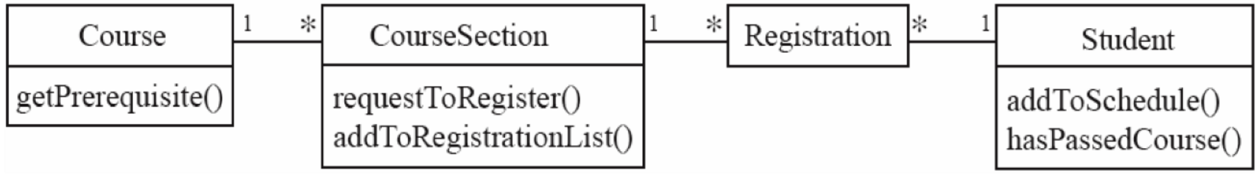
(OBJECT-ORIENTED SOFTWARE ENGINEERING)

Homework 6

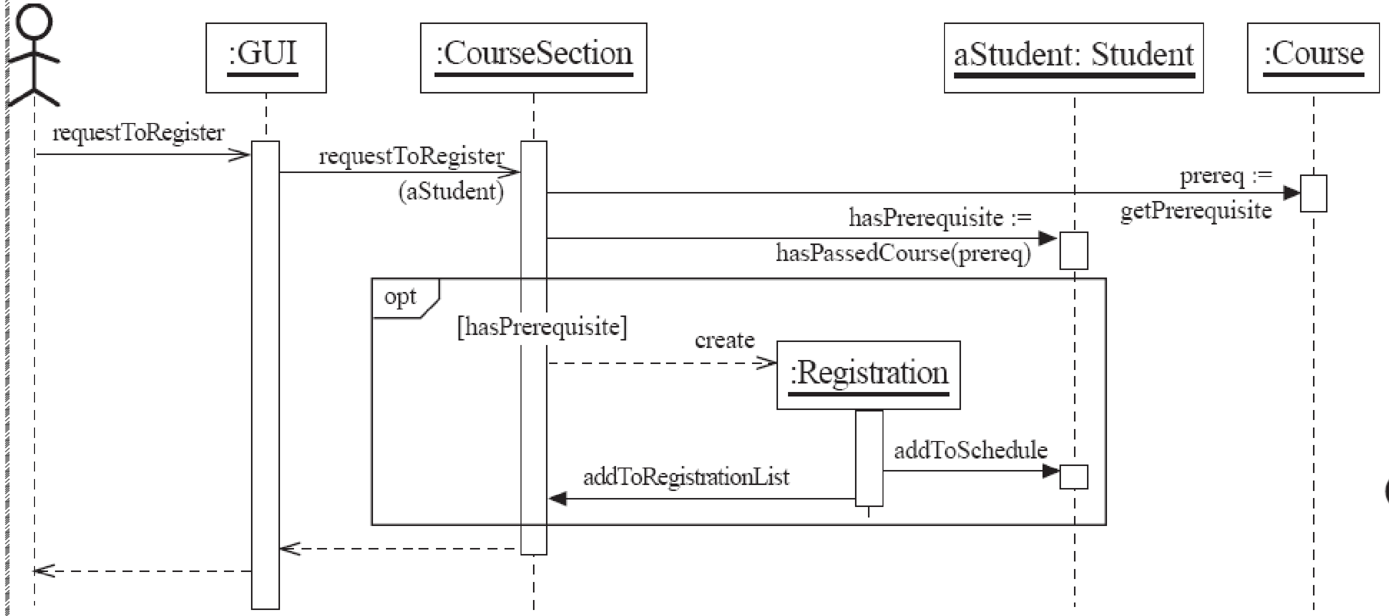
(Implementing Classes Based on Interaction and State Diagrams)

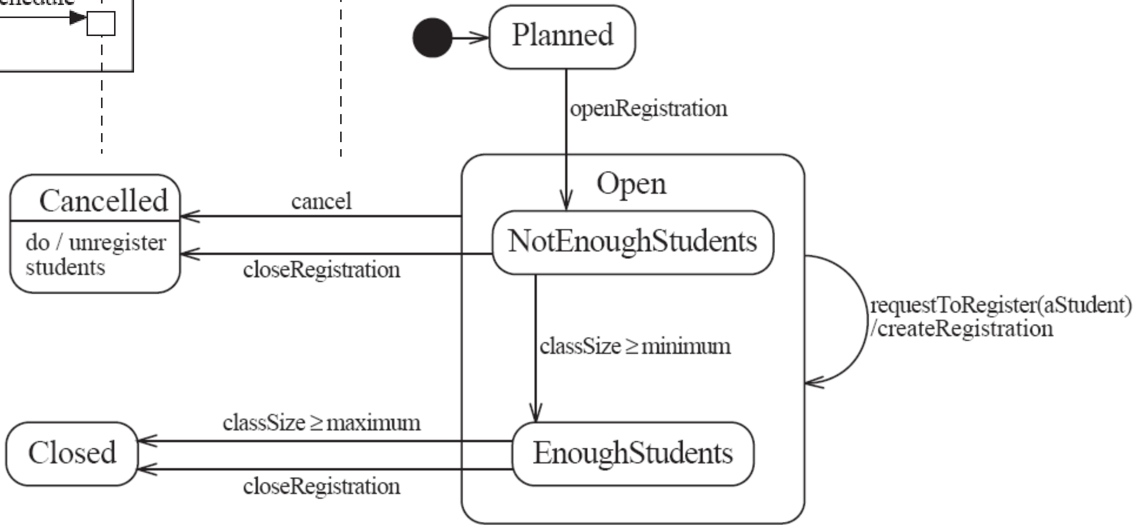
日期:2018/01/04
學號:P76064538
姓名:簡君聿
(程式碼以上傳至 https://github.com/Alex-CHUN-YU/OOP)

內容



Interaction and State Diagrams

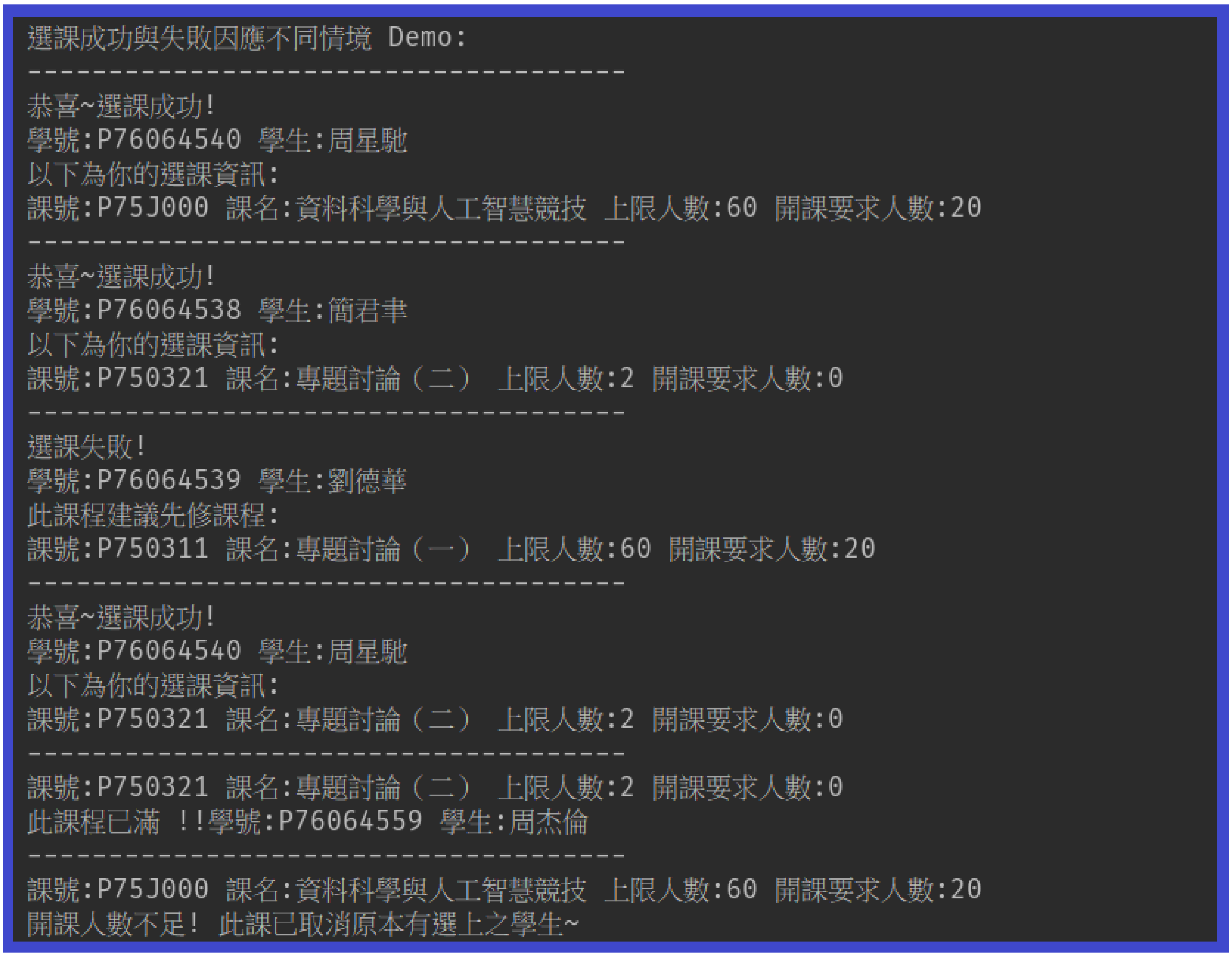




Course Registration System Demo

根據不同選課情況所造成的成功與失

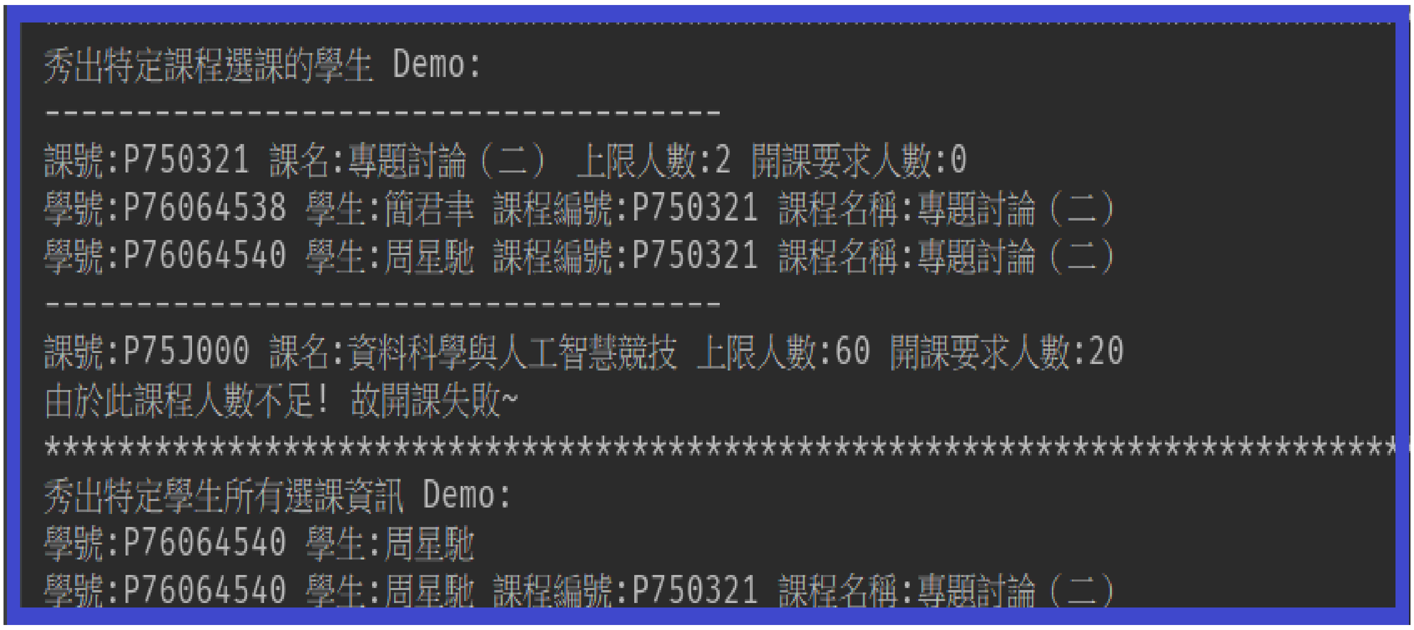
敗所實作的 Demo 呈現。



```
選課成功與失敗因應不同情境 Demo:
------------------------------------
恭喜~選課成功!
學號:P76064540 學生:周星馳
以下為你的選課資訊:
課號:P75J000 課名:資料科學與人工智慧競技 上限人數:60 開課要求人數:20
------------------------------------
恭喜~選課成功!
學號:P76064538 學生:簡君聿
以下為你的選課資訊:
課號:P750321 課名:專題討論（二） 上限人數:2 開課要求人數:0
------------------------------------
選課失敗!
學號:P76064539 學生:劉德華
此課程建議先修課程:
課號:P750311 課名:專題討論（一） 上限人數:60 開課要求人數:20
------------------------------------
恭喜~選課成功!
學號:P76064540 學生:周星馳
以下為你的選課資訊:
課號:P750321 課名:專題討論（二） 上限人數:2 開課要求人數:0
------------------------------------
課號:P750321 課名:專題討論（二） 上限人數:2 開課要求人數:0
此課程已滿 !!學號:P76064559 學生:周杰倫
------------------------------------
課號:P75J000 課名:資料科學與人工智慧競技 上限人數:60 開課要求人數:20
開課人數不足! 此課已取消原本有選上之學生~
```

課程所有學生以及學生所選上之課程

實作的 Demo 呈現。



```
秀出特定課程選課的學生 Demo:
--------------------------------------
課號:P750321 課名:專題討論（二） 上限人數:2 開課要求人數:0
學號:P76064538 學生:簡君聿 課程編號:P750321 課程名稱:專題討論（二）
學號:P76064540 學生:周星馳 課程編號:P750321 課程名稱:專題討論（二）
--------------------------------------
課號:P75J000 課名:資料科學與人工智慧競技 上限人數:60 開課要求人數:20
由於此課程人數不足! 故開課失敗~
****************************************************************************
秀出特定學生所有選課資訊 Demo:
學號:P76064540 學生:周星馳
學號:P76064540 學生:周星馳 課程編號:P750321 課程名稱:專題討論（二）
```

RegistrationTest Class

```
import registration_information.Course;
import registration_information.CourseSection;
import registration_information.Student;

/**
 * Base on Sequence diagrams 所設計的 GUI Test.
 * @version 1.0 2018 年 01 月 02 日
 * @author ALEX-CHUN-YU
 */
class RegistrationTest {
	/**
	 * This is test.
	 * @param args system default
	 */
	public static void main(String[] args) {
		// 上學期課程
		Course priorCourseOne = new Course("P750311", "專題討論
（一）", 60 , 20);
		Course priorCourseSecond = new Course("P764600", "資料探勘
", 50 , 20);

		// 這學期課程
		Course courseOne = new Course("P750321", "專題討論（二）",
2, 0);
		courseOne.setPreviousCourse(priorCourseOne);
		Course courseSecond = new Course("P75J000", "資料科學與人
工智慧競技", 60 , 20);
		courseSecond.setPreviousCourse(priorCourseSecond);

		// 學生以及每位學生已經通過的課程
		Student studentOne = new Student("P76064538", "簡君聿");
		studentOne.setHistoryCourse(priorCourseOne);

		Student studentSecond = new Student("P76064539", "劉德華");
		studentSecond.setHistoryCourse(priorCourseSecond);
```

```
            Student studentThird = new Student("P76064540", "周星馳");
            studentThird.setHistoryCourse(priorCourseOne);
            studentThird.setHistoryCourse(priorCourseSecond);

            Student studentFourth = new Student("P76064559", "周杰倫");
            studentFourth.setHistoryCourse(priorCourseOne);

            // 開始進行選課

System.out.println("**********************************************
********************************************");
            System.out.println("選課成功與失敗因應不同情境 Demo:");
            CourseSection courseSectionOne = new
CourseSection(courseOne);
            CourseSection courseSectionSecond = new
CourseSection(courseSecond);
            courseSectionOne.openRegistration();
            courseSectionSecond.openRegistration();
            // 選課成功 Demo
            courseSectionSecond.requestToRegister(studentThird);
            // 選課成功 Demo
            courseSectionOne.requestToRegister(studentOne);
            // 選課失敗 Demo
            courseSectionOne.requestToRegister(studentSecond);
            // 選課成功 Demo
            courseSectionOne.requestToRegister(studentThird);
            // 選課人數已滿 Demo
            courseSectionOne.requestToRegister(studentFourth);
            courseSectionSecond.closeRegistration();

System.out.println("**********************************************
********************************************");
            System.out.println("秀出特定課程選課的學生 Demo:");
            // 秀出全部選課的學生 Demo
            courseSectionOne.getAllRegistrationOfCourse();
            courseSectionSecond.getAllRegistrationOfCourse();
```

```
System.out.println("***************************************************
*************************************************");
          System.out.println("秀出特定學生所有選課資訊 Demo:");
          // 秀出學生所有選上的課程 Demo
          studentThird.getAllRegistrationOfStudent();
     }
}
```

Course Class

```
package registration_information;

/**
 * Course Class.
 * @version 1.0 2018 年 01 月 01 日
 * @author ALEX-CHUN-YU
 */
public class Course {
	/**
	 * Course ID.
	 */
	private String courseID;

	/**
	 * Course Name.
	 */
	private String courseName;

	/**
	 * Course Maximum Number.
	 */
	private int maximum;

	/**
	 * Course Minimum Number.
	 */
	private int minimum;

	/**
	 * Course.
	 */
	private Course course;

	/**
	 * Constructor.
```

```
     * @param courseID courseID
     * @param courseName courseName
     * @param maximum maximum
     * @param minimum minimum
     */
    public Course(String courseID, String courseName, int maximum, int
minimum) {
        this.courseID = courseID;
        this.courseName = courseName;
        this.maximum = maximum;
        this.minimum = minimum;
    }

    /**
     * Get Course ID.
     * @return courseID course id
     */
    public String getCourseID() {
        return courseID;
    }

    /**
     * Get Course Name.
     * @return courseName course name
     */
    public String getCourseName() {
        return courseName;
    }

    /**
     * Get Course Maximum Number.
     */
    public int getMaximum() {
        return maximum;
    }

    /**
     * Get Course Minimum Number.
```

```
  */
public int getMinimum() {
      return minimum;
}

/**
  * Get Pre Requisite.
  */
public Course getPrerequisite() {
      return this;
}

/**
  * Set Previously Course.
  */
public void setPreviousCourse(Course course) {
      this.course = course;
}

/**
  * Get Previously Course.
  */
public Course getPreviousCourse() {
      return this.course;
}

/**
  * Show Course Information.
  */
public void showInformationOfCourse() {
      System.out.print("課號:" + this.getCourseID());
      System.out.print("  課名:" + this.getCourseName());
      System.out.print("  上限人數:" + this.getMaximum());
      System.out.println("  開課要求人數:" + this.getMinimum());
      //System.out.println("-------------------------------");
}}
```

CourseSection Class

```
package registration_information;

import java.util.ArrayList;
import java.util.Iterator;

/**
 * Course Section.
 * @version 1.0 2018 年 01 月 01 日
 * @author ALEX-CHUN-YU
 */
public class CourseSection {
	/**
	 * Registration List.
	 */
	private ArrayList<Registration> registrationList;

	/**
	 * Registration Course.
	 */
	public Course course;

	/**
	 * Open Course Registration Flag.
	 */
	private boolean open = false;

	/**
	 * Closed Or Canceled Course Registration Flag.
	 */
	private boolean closedOrCanceled = false;

	/**
	 * Constructor.
	 */
	public CourseSection(Course course) {
		this.course = course;
		registrationList = new ArrayList<Registration>();
	}

	/**
	 * Open Course Registration.
	 */
	public void openRegistration() {
		if(!closedOrCanceled) {
			open = true;
		}
	}

	/**
	 * Closed Course Registration.
	 */
	public void closeRegistration() {
		open = false;
		closedOrCanceled = true;
		if (registrationList.size() < course.getMinimum()) {
			System.out.println("-----------------------------------");
			course.showInformationOfCourse();
			System.out.println("開課人數不足! 此課已取消原本有選
上之學生~");
			unregisterStudents();
		}
	}

	/**
	 * Canceled Course Registration.
	 */
	public void cancel() {
		open = false;
		closedOrCanceled = true;
		unregisterStudents();
	}

	/**
```

```
	 * Unregister Students.
	 */
	private void unregisterStudents() {
		Iterator it = registrationList.iterator();
		while (it.hasNext()) {
			Registration r = (Registration)it.next();
			r.unregisterStudent();
			it.remove();
		}
	}

	/**
	 * Request To Register.
	 * @param student student
	 */
	public void requestToRegister(Student student) {
		if (open) {
			Course prereq = course.getPrerequisite();
			if (student.hasPassedCourse(prereq)) {
				new Registration(this, student);
			} else {
				System.out.println("------------------------------------");
				System.out.println("選課失敗!");
				student.showInformationOfStudent();
				System.out.println("此課程建議先修課程:");

course.getPreviousCourse().showInformationOfCourse();
			}
			// Course Full, Check for automatic transition to 'Closed'
state.
			if (registrationList.size() >= course.getMaximum()) {
				open = false;
				closedOrCanceled = true;
			}
		} else {
			System.out.println("------------------------------------");
			course.showInformationOfCourse();
			System.out.print("此課程已滿 !!");
```

```
                student.showInformationOfStudent();
            }
        }

        /**
         * Add To Registration List(link).
         * @param registrationOne book one
         */
        void addToRegistrationList(Registration registrationOne) {
            registrationList.add(registrationOne);
        }

        /**
         * Get All Registration Of Course.
         */
        public void getAllRegistrationOfCourse() {
            System.out.println("-------------------------------------");
            course.showInformationOfCourse();
            Iterator<Registration> iterator = registrationList.iterator();
            while (iterator.hasNext()) {
                iterator.next().showInformationOfRegistration();
            }
            if (registrationList.size() == 0) {
                System.out.println("由於此課程人數不足! 故開課失敗~");
            }
        }
    }
```

Registration Class

```
package registration_information;

/**
 * Registration.
 * @version 1.0 2018 年 01 月 01 日
 * @author ALEX-CHUN-YU
 */
public class Registration {
	/**
	 * Course Section.
	 */
	private CourseSection courseSection;

	/**
	 * Student.
	 */
	private Student student;

	/**
	 * Constructor(link).
	 * @param courseSection courseSection
	 * @param student student
	 */
	Registration(CourseSection courseSection, Student student) {
		System.out.println("------------------------------------");
		System.out.println("恭喜~選課成功!");
		student.showInformationOfStudent();
		this.courseSection = courseSection;
		this.courseSection.addToRegistrationList(this);
		this.student = student;
		this.student.addToSchedule(this);
		System.out.println("以下為你的選課資訊:");
		courseSection.course.showInformationOfCourse();
	}
	/**
	  * Unregister Student.
	  */
	public void unregisterStudent() {
		student.removeToSchedule(this);
	}

	/**
	  * Show Registration Information.
	  */
	public void showInformationOfRegistration() {
		System.out.print("學號:" + student.getStudentID());
		System.out.print("  學生:" + student.getStudentName());
		System.out.print("  課程編號:" + 
courseSection.course.getCourseID());
		System.out.println("  課程名稱:" + 
courseSection.course.getCourseName());
	}
}
```

Student Class

```
package registration_information;

import java.util.ArrayList;
import java.util.Iterator;

/**
  * Students Class.
  * @version 1.0 2018 年 01 月 01 日
  * @author ALEX-CHUN-YU
  */
public class Student {
	/**
	  * Registrations List.
	  */
	private ArrayList<Registration> registrationList = new
ArrayList<Registration>();

	/**
	  * Student ID.
	  */
	private String studentID;

	/**
	  * Student Name.
	  */
	private String studentName;

	/**
	  * History Course List.
	  */
	private ArrayList<Course> historyCourses = new
ArrayList<Course>();

	/**
	  * Constructor.
  * @param studentID studentID
  * @param studentName studentName
  */
public Student(String studentID, String studentName) {
        this.studentID = studentID;
        this.studentName = studentName;
}

/**
  * Get Student ID.
  * @return studentID student id
  */
public String getStudentID() {
        return studentID;
}

/**
  * Get Student Name.
  * @return studentName student name
  */
public String getStudentName() {
        return studentName;
}

/**
  * Add To Schedule(link).
  * @param registrationOne registration One
  */
void addToSchedule(Registration registrationOne) {
        registrationList.add(registrationOne);
}

/**
  * Remove To Schedule(link).
  * @param registrationOne registration One
  */
void removeToSchedule(Registration registrationOne) {
        registrationList.remove(registrationOne);
```

```
	}

	/**
	 * Has Passed Course Of Student(association).
	 */
	public boolean hasPassedCourse(Course course) {
		return
this.getHistoryCourse().contains(course.getPreviousCourse());
	}

	/**
	 * Has Passed Course Of Student(association).
	 */
	public void setHistoryCourse(Course course) {
		historyCourses.add(course);
	}

	/**
	 * Has Passed Course History Of Student.
	 */
	public ArrayList<Course> getHistoryCourse() {
		return historyCourses;
	}

	/**
	 * Get All Registration Of Student.
	 */
	public void getAllRegistrationOfStudent() {
		this.showInformationOfStudent();
		Iterator<Registration> iterator = registrationList.iterator();
		while (iterator.hasNext()) {
			iterator.next().showInformationOfRegistration();
		}
	}

	/**
	 * Show Student Information.
	 */
```

```
        public void showInformationOfStudent() {
                System.out.print("學號:" + this.getStudentID());
                System.out.println("  學生:" + this.getStudentName());
                //System.out.println("--------------------------------");
        }
}
```