物件導向軟體工程

(OBJECT-ORIENTED SOFTWARE ENGINEERING)

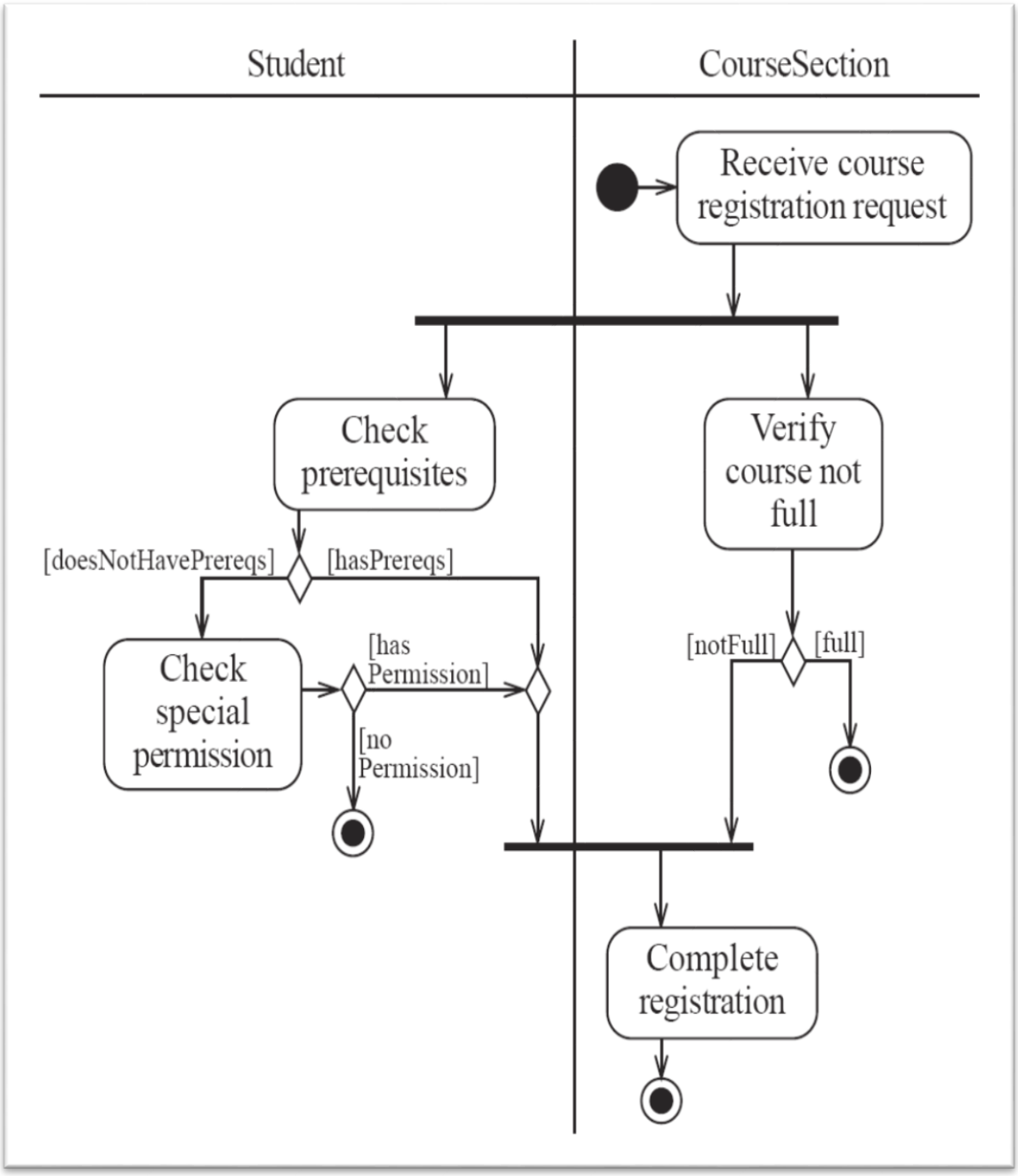
Homework 7

(Implementing Classes Based on Activity diagrams)

日期:2018/01/18
學號:P76064538
姓名:簡君聿
(程式碼以上傳至 https://github.com/Alex-CHUN-YU/OOP)

内容

Activity Diagrams



Course Registration System Demo

```
"C:\Program Files\Java\jdk1.8.0_121\bin\java" ...
透過 MultiThread 的方式進行驗證選課成功與否之 Demo:
---------------------------------------
```

```
有完成先修課程 and 課程人數未滿 and 沒有特殊權限:
Starting (verify course not full) thread
Starting (check prerequisites) thread
Thread (verify course not full) exiting.
Thread (check prerequisites) exiting.
恭喜~選課成功!
學號:P76064538 學生:蠟筆小新 特殊選課權限:false
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:1
```

```
沒有完成先修課程 and 課程人數未滿 and 沒有特殊權限:
Starting (check prerequisites) thread
Thread (check prerequisites) exiting.
Starting (verify course not full) thread
Thread (verify course not full) exiting.
選課失敗!
學號:P76064539 學生:阿兩 特殊選課權限:false
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
```

```
有完成先修課程 and 課程人數未滿 and 有特殊權限:
Starting (verify course not full) thread
Thread (verify course not full) exiting.
Starting (check prerequisites) thread
Thread (check prerequisites) exiting.
恭喜~選課成功!
學號:P76064540 學生:陳妍希 特殊選課權限:true
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:1
```

```
沒有完成先修課程 and 課程人數未滿 and 有特殊權限:
Starting (verify course not full) thread
Thread (verify course not full) exiting.
Starting (check prerequisites) thread
Thread (check prerequisites) exiting.
恭喜~選課成功!
學號:P76064540 學生:陳妍希 特殊選課權限:true
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
```

```
有完成先修課程 and 課程人數未滿 and 有特殊權限:
Starting (verify course not full) thread
Thread (verify course not full) exiting.
Starting (check prerequisites) thread
Thread (check prerequisites) exiting.
恭喜~選課成功!
學號:P76064559 學生:簡君聿 特殊選課權限:true
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
```

```
有完成先修課程 and 課程人數已滿 and 有特殊權限:
Starting (verify course not full) thread
Thread (verify course not full) exiting.
Starting (check prerequisites) thread
Thread (check prerequisites) exiting.
課號:P750321 課名:專題討論（二） 上限人數:2 開課要求人數:1
此課程已滿 !!學號:P76064588 學生:志明 特殊選課權限:true
```

RegistrationThreadApplicationTest Class

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

/**
 * Registration Thread Application Test Base On Activity Diagrams.
 * @version 1.0 2018 年 01 月 07 日
 * @author ALEX-CHUN-YU
 */
class RegistrationThreadApplicationTest {
    /**
     * This Is Multi Thread 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, 1);
        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", "蠟筆小新", false);
        studentOne.setHistoryCourse(priorCourseOne);

        Student studentSecond = new Student("P76064539", "阿兩",
```

```
false);
        studentSecond.setHistoryCourse(priorCourseSecond);

        Student studentThird = new Student("P76064540", "陳妍希",
true);
        studentThird.setHistoryCourse(priorCourseOne);

        Student studentFourth = new Student("P76064559", "簡君聿",
true);
        studentFourth.setHistoryCourse(priorCourseOne);
        studentFourth.setHistoryCourse(priorCourseSecond);

        Student studentFifth = new Student("P76064560", "春嬌", false);
        studentFifth.setHistoryCourse(priorCourseOne);
        studentFifth.setHistoryCourse(priorCourseSecond);

        Student studentSixth = new Student("P76064588", "志明", true);
        studentSixth.setHistoryCourse(priorCourseOne);
        studentSixth.setHistoryCourse(priorCourseSecond);

        // 開始進行選課
        System.out.println("透過 MultiThread 的方式進行驗證選課成
功與否之 Demo:");
        CourseSection courseSectionOne = new
CourseSection(courseOne);
        CourseSection courseSectionSecond = new
CourseSection(courseSecond);
        courseSectionOne.openRegistration();
        courseSectionSecond.openRegistration();
        // 選課成功
        System.out.println("-----------------------------------");
        System.out.println("有完成先修課程 and 課程人數未滿 and
沒有特殊權限:");
        courseSectionOne.requestToRegister(studentOne);
        // 選課失敗
        System.out.println("-----------------------------------");
        System.out.println("沒有完成先修課程 and 課程人數未滿 and
沒有特殊權限:");
```

```
            courseSectionOne.requestToRegister(studentSecond);
            // 選課成功
            System.out.println("-----------------------------------");
            System.out.println("有完成先修課程 and 課程人數未滿 and
有特殊權限:");
            courseSectionOne.requestToRegister(studentThird);
            // 選課成功
            System.out.println("-----------------------------------");
            System.out.println("沒有完成先修課程 and 課程人數未滿 and
有特殊權限:");
            courseSectionSecond.requestToRegister(studentThird);
            // 選課成功
            System.out.println("-----------------------------------");
            System.out.println("有完成先修課程 and 課程人數未滿 and
有特殊權限:");
            courseSectionSecond.requestToRegister(studentFourth);
            // 選課失敗
            System.out.println("-----------------------------------");
            System.out.println("有完成先修課程 and 課程人數已滿 and
有特殊權限:");
            courseSectionOne.requestToRegister(studentSixth);
        }
}
```

CourseSection Class And RegistrationVerifyThread Class

```
package registration_information;

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

/**
 * Registration Verify Thread.
 * @version 1.0 2018 年 01 月 06 日
 * @author ALEX-CHUN-YU
 */
class RegistrationVerifyThread extends Thread {
	/**
	 * Thread Function Name.
	 */
	private String threadName;

	/**
	 * Course Section.
	 */
	private CourseSection	courseSection;

	/**
	 * Constructor.
	 * @param name function name
	 * @param courseSection course section
	 */
	RegistrationVerifyThread(String name, CourseSection courseSection) {
		this.threadName = name;
		this.courseSection = courseSection;
	}

	/**
	 * Thread Run And Synchronization.
	 */
	public void run() {
```

```
            System.out.println("Starting " +    threadName + " thread");
            synchronized(courseSection) {
                if
(threadName.equals(ConstantField.VERIFY_COURSE_NOT_FULL)) {
                    courseSection.verifyCourseNotFull();
                } if
(threadName.equals(ConstantField.CHECK_PREREQUISITES)) {
                    courseSection.checkPrerequisites();
                }
            }
            System.out.println("Thread " +    threadName + " exiting.");
        }
}

/**
 * Course Section.
 * @version 1.0 2018 年 01 月 06 日
 * @author ALEX-CHUN-YU
 */
public class CourseSection {
    /**
     * Registration List.
     */
    private ArrayList<Registration> registrationList;

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

    /**
     * Has Prerequisites Flag.
     */
    private boolean hasPrereqs = false;

    /**
     * Open Course Registration Flag.
     */
```

```
private boolean open = false;

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

/**
 * 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) {
            this.student = student;
// Fork
            Thread threadOne = new
RegistrationVerifyThread(ConstantField.VERIFY_COURSE_NOT_FULL,
this);
            Thread threadSecond = new
```

```
RegistrationVerifyThread(ConstantField.CHECK_PREREQUISITES, this);
            threadOne.start();
            threadSecond.start();
            // wait for threads to end
// Join
            try {
                  threadOne.join();
                  threadSecond.join();
            } catch (Exception e) {
                  System.out.println("Interrupted");
            }
            if (open) {
                  if (hasPrereqs) {
                        new Registration(this, student);
                  } else {
                        System.out.println("選課失敗!");
                        student.showInformationOfStudent();
                        System.out.println("此課程建議先修課程:");

course.getPreviousCourse().showInformationOfCourse();
                  }
            } else {
                  course.showInformationOfCourse();
                  System.out.print("此課程已滿 !!");
                  student.showInformationOfStudent();
            }
      }

      /**
       * Check Prerequisites.
       */
      void checkPrerequisites() {
            Course prereq = course.getPrerequisite();
            hasPrereqs = student.hasPassedCourse(prereq);
            if (!hasPrereqs) {
                  hasPrereqs = student.checkSpecialPermission();
            }
      }
```

```
    /**
     * Verify Course Not Null.
     */
    void verifyCourseNotFull() {
        if (registrationList.size() >= course.getMaximum()) {
            open = false;
            closedOrCanceled = true;
        }
    }

    /**
     * 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~");
        }
    }
}
```

Course Class

```
package registration_information;

/**
 * Course Class.
 * @version 1.0 2018 年 01 月 06 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;
    }

    /**
     * Get 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("------------------------------");
    }
}
```

Registration Class

```
package registration_information;

/**
 * Registration.
 * @version 1.0 2018 年 01 月 06 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("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 07 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;

	/**
	 * Special Permission.
	 */
	private boolean hasPermission = false;

	/**
	 * History Course List.
	 */
```

```
    private ArrayList<Course> historyCourses = new
ArrayList<Course>();

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

    /**
     * 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);
	}

	/**
	 * Check Special Permission.
	 */
	public boolean checkSpecialPermission() {
		return hasPermission;
	}

	/**
	 * 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.print(" 學生:" + this.getStudentName());
		System.out.println(" 特殊選課權限:" +
this.checkSpecialPermission());
		//System.out.println("------------------------------");
	}
}
```

ConstantField Class

```
package registration_information;

/**
 * Constant Field.
 * @version 1.0 2018 年 01 月 06 日
 * @author ALEX-CHUN-YU
 */
public class ConstantField {
	public final static String VERIFY_COURSE_NOT_FULL = "verify
course not null";
	public final static String CHECK_PREREQUISITES = "check
prerequisites";
}
```