0112562 OBJECT ORIENTED PROG., Section 1, 1ST MIDTERM (02/04/2014)

Question 1: Write the source code of classes Person and Lecturer. (10p)

```
public abstract class Person {
    private String name;
    public Person(String name) { this.name = name; }
    public String getName() { return name; }
}

public class Lecturer extends Person implements java.io.Serializable {
    private static final long serialVersionUID = 1L;
    private Integer ID;

    public Lecturer(String name, Integer ID) {
        super(name);
        this.ID = ID;
    }
    public Integer getID() { return ID; }
}
```

Question 2: Write the source code of StudentAlreadyExistsException and InvalidCapacityException. (10p)

```
import java.io.IOException;
public class StudentAlreadyExistsException extends IOException {
    private static final long serialVersionUID = 1L;
    public StudentAlreadyExistsException(String arg0) {
        super(arg0);
    }
}

import java.io.IOException;
public class InvalidCapacityException extends IOException {
    private static final long serialVersionUID = 1L;
    public InvalidCapacityException(String arg0) {
        super(arg0);
    }
}
```

Question 3: Write the source code of the methods setCapacity and addStudent of class Course. (20p)

```
    public void setCapacity( int capacity ) throws InvalidCapacityException {
        if( capacity >= students.size() )
            this.capacity = capacity;
        else
            throw new InvalidCapacityException("Students already enrolled: "
                    + students.size() + ", desired capacity: " + capacity);
    }
    public void addStudent( Student std ) throws StudentAlreadyExistsException {
        if( findStudent(std.getStdNr()) != null && capacity > students.size() )
            students.put(std.getStdNr(), std);
        else
            throw new StudentAlreadyExistsException("A student with number "
                    + std.getStdNr() + " already exits.");
    }
```

Question 4: Write the source code of class USIS. (60p)

```
import java.io.*;
import java.util.*;
public class USIS{
    private HashMap<String,Course> courses;
    private HashMap<Integer,Person> people;

    public USIS() {
        courses = new HashMap<String,Course>() ;
        people = new HashMap<Integer,Person>();
    }
    public void createStudent( String name, Integer ID ) {
        people.put( ID, new Student(name,ID) );
    }
    public void createLecturer( String name, Integer ID ) {
        people.put( ID, new Lecturer(name,ID) );
    }
```

```
    public void createCourse(String code, String name, int capacity, Integer lecturerID){
        try {
            Course course = new Course(code, name);
            course.setCapacity(capacity);
            Lecturer teacher = (Lecturer) people.get(lecturerID);
            course.setTeacher(teacher);
            courses.put(code, course);
        }
        catch (InvalidCapacityException e) {
            e.printStackTrace();
        }
    }
    public void setCapacity( String code, int capacity ) {
        try {
            courses.get(code).setCapacity(capacity);
        }
        catch (InvalidCapacityException e) {
            e.printStackTrace();
        }
        catch (NullPointerException e) {
            e.printStackTrace();
        }
    }
    public void addStudentToCourse( String courseCode, Integer studentNr ) {
        try {
            Person p = people.get(studentNr);
            Course c = courses.get(courseCode);
            if( p != null && c != null && p instanceof Student ) {
                Student std = (Student) p;
                c.addStudent(std);
            }
        }
        catch (StudentAlreadyExistsException e) {
            e.printStackTrace();
        }
    }
    public void saveAllObjectsToFile( String path ) {
        try {
            ObjectOutputStream stream = new ObjectOutputStream(
                    new FileOutputStream(path) );
            stream.writeObject(courses);
            stream.writeObject(people);
            stream.close();
        }
        catch ( IOException e ) {
            e.printStackTrace();
        }
    }
    @SuppressWarnings("unchecked")
    public void loadAllObjectsFromFile( String path ) {
        try {
            ObjectInputStream stream = new ObjectInputStream(
                    new FileInputStream(path) );
            courses = (HashMap<String, Course>) stream.readObject();
            people = (HashMap<Integer, Person>) stream.readObject();
            stream.close();
        }
        catch ( IOException e ) {
            e.printStackTrace();
        }
        catch ( ClassNotFoundException e ) {
            e.printStackTrace();
        }
    }
}
```

Note: USIS.createXXX methods need exception handling as well. However, as the relationship is not shown in the UML class diagram, I have not included that in grading.