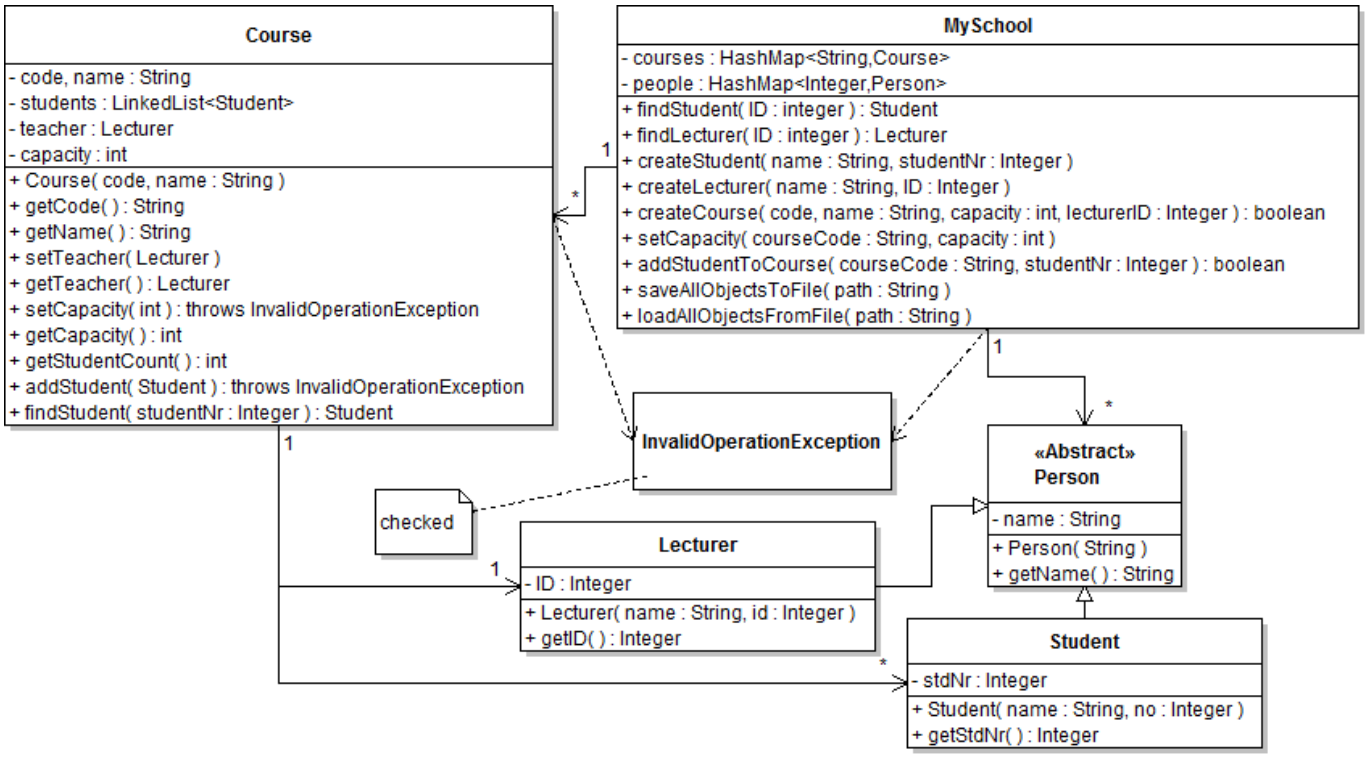
BLM2562 OBJECT ORIENTED PROG., Gr.2 & 3, 1ST MIDTERM (06/04/2016)

Duration:	90 mins.			**Score:**				**Student Nr:**	**Signature:**
Grading:	**1**	**2**	**3**	**4**	**5**	**6**	**Group**	**Name, Surname:**	
	10	10	10	10	60				

QUESTIONS



Answer the following questions according to the UML class schema given above. You may need to extract hidden information from the schema and add necessary code while answering the questions.

Question 1: Write the source code of classes Person and Student.
Question 2: Write the source code of InvalidOperationException.
Question 3: Write the source code of the method setCapacity of class Course. If one attempts to set the capacity of a course to a number that is smaller than the students already enrolled, an exception with detailed information must be generated.
Question 4: Write the source code of the method addStudent of class Course. A student must not be added to a course more than once. Moreover, the capacity must not be full. If one of those cases is attempted, an exception with detailed information must be generated.
Question 5: Write the source code of the following methods of class MySchool:

- findStudent: Carry out the task.
- createStudent: If a student with the given number already exists, an exception with detailed information must be generated. Carry out the task otherwise.
- createCourse: The course with the given code must not already exist and the lecturer with the given ID must exist in order to carry out the task. The method must return false otherwise.
- setCapacity: Carry out the task.
- addStudentToCourse: The student with given ID and the course with given code must already exist in order to carry out the task. The method must return false otherwise.
- saveAllObjectsToFile: Carry out the task.

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 InvalidOperationException.

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

Question 3: Write the source code of the method setCapacity of class Course.

```
	public void setCapacity( int capacity ) throws InvalidOperationException {
		if( capacity >= students.size() )
			this.capacity = capacity;
		else
			throw new InvalidOperationException( students.size() +
					"Students already enrolled in course " + code +
					", cannot decrease capacity to: " + capacity);
	}
```

Question 4: Write the source code of the method addStudent of class Course.

```
	public void addStudent( Student std ) throws InvalidOperationException {
		if( findStudent(std.getStdNr()) != null )
			throw new InvalidOperationException("A student with number "
					+ std.getStdNr() + " already exits in course " + code );
		if( capacity <= students.size() )
			throw new InvalidOperationException("Capacity of course " + code
					+ " is not enough.");
		students.add(std);
    }
```

Question 5: Write the source code of the following methods of class MySchool

```
public Student findStudent( Integer ID ) {
	Person p = people.get(ID);
	if( p != null && p instanceof Student )
		return (Student)p;
	return null;
}
public void createStudent( String name, Integer ID ) throws InvalidOperationException {
	if( findStudent(ID) == null )
		people.put( ID, new Student(name,ID) );
	else throw new InvalidOperationException("Cannot create a student with ID "
		+ ID + ", a student with the same ID already exists.");
}
public boolean createCourse( String code, String name, int capacity, Integer lecturerID ) {
	try {
		if( courses.get(code) != null ) return false;
		Course course = new Course(code, name);
		course.setCapacity(capacity);
		Lecturer teacher = findLecturer(lecturerID);
		if( teacher == null ) return false;
		course.setTeacher(teacher);
		courses.put(code, course);
	}
	catch (InvalidOperationException e) {
		e.printStackTrace();
	}
	return true;

}
public void setCapacity( String code, int capacity ) {
	try {
		courses.get(code).setCapacity(capacity);
	}
	catch (InvalidOperationException e) {
		e.printStackTrace();
	}
	catch (NullPointerException e) {
		e.printStackTrace();
	}
  }
public boolean addStudentToCourse( String courseCode, Integer studentNr ) {
	try {
		Student std = findStudent(studentNr);
		Course c = courses.get(courseCode);
		if( std != null && c != null ) {
			c.addStudent(std);
		}
		else return false;
	}
	catch (InvalidOperationException e) {
		e.printStackTrace();
	}
	return true;
}
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();
	}
}
```