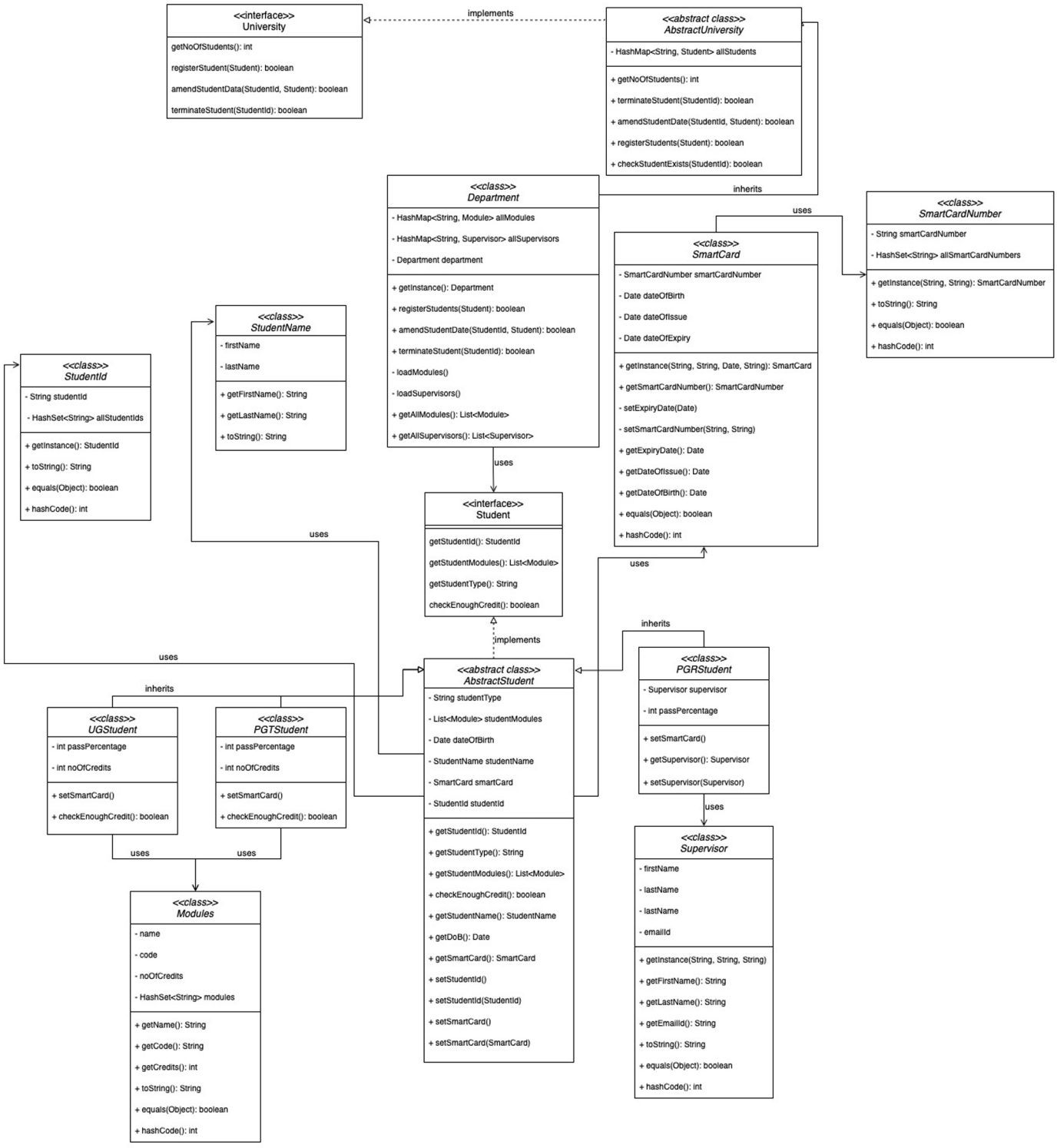
University Department

CSC8404 Assessed Coursework

Submitted By:

Name: Aswin Kambian Veettil
Student Id #: 210378557

UML Class Diagram



Implementation

The classes and interface used for the implementation are:

Name	Type	Implements	Extends
University	Interface		
AbstractUniversity	Abstract Class	University	
Department	Class		AbstractUniversity
Student	Interface		
AbstractStudent	Abstract Class	Student	
UGStudent	Class		AbstractStudent
PGTStudent	Class		AbstractStudent
PGRStudent	Class		AbstractStudent

- **University - *Interface***

The University interface defines the basic functionalities of the University including registration, updation and termination of students.

- **AbstractUniversity - *Abstract Class***

The AbstractUniversity abstract class implements the functionalities defined by the University interface. This class acts as the parent of the different departments in the University.

- **Department - *Class***

Department class by default implements the functionalities defined in the parent AbstractUniversity. Additionally, the custom logic required for individual department is implemented on this class. Loading of modules and supervisors of the department is done in this class. Validation of student data and initiation of student id and smart card generation is done here.

- **Student - *Interface***

The Student interface defines the basic attributes of Student including studentId, student type, list of modules the student has chosen.

- **AbstractStudent - *Abstract Class***

The AbstractStudent abstract class implements the functionalities defined by the student interface. Additional methods required to store student information is also defined and implemented here. The studentId and smart card is assigned to the student here.

- **UGStudent - *Class***

The UGStudent class by default implements all the functionalities in the AbstractStudent class. Additionally, the custom logic required specifically to UGStudents is implemented here. This

includes logic to check if the student has registered for enough modules and if the student is of the minimum age to receive a smart card.

- **PGTStudent - *Class***

The PGTStudent class is similar to UGStudent class. The class by default implements all the functionalities in the AbstractStudent class. Additionally, the custom logic required specifically to PGTStudents is implemented here. This includes logic to check if the student has registered for enough modules and if the student is of the minimum age to receive a smart card.

- **PGRStudent - *Class***

The PGRStudent by default implements all the functionalities in the AbstractStudent class. Additionally, the custom logic required specifically to PGTStudents is implemented here. This includes logic to check if the student is of the minimum age to receive a smart card. The PGRStudent class has the functionality to allow the system to assign a supervisor to the student.

Helper Classes

- **StudentId**

An object of type StudentId is used in Student class to store the studentId assigned to each student. The class contains the logic to generate a unique studentId for each student. A static HashSet is used to ensure the uniqueness of the studentIds generated.

- **StudentName**

An object of type StudentName is used in Student class to store the student name.

- **SmartCard**

An object of type SmartCard is used in Student class to store the details of the smart card assigned to the student. The smart card contains information like student name, date of birth, smart card number and the date of issue and expiry of the smart card. The date of expiry is auto calculated by the object depending on the type of student. If the student is an undergraduate, the smart card expires 4 years from the date of issue. If the student is a postgraduate, the smart card expires a year from the date of issue.

- **SmartCardNumber**

An object of type SmartCardNumber is used in SmartCard class to store the smart card number. The class contains the logic to generate a unique smart card number for each student. A static HashSet is used to ensure the uniqueness of the smart card numbers generated.

- **Supervisor**

An object of type Supervisor is used in Postgraduate Student(PGRStudent) class to store the details of the supervisor assigned to the student by the Department. A HashSet on the supervisor email id is used to ensure no duplicates are loaded onto the system. In addition, in order to facilitate comparison between two Supervisor objects, equals and hashCode method have been overridden.

A list of objects of type Supervisor is maintained by the Department class to store the list of all available Supervisors. The list of supervisors is available to the Department in the form of a CSV file which is then loaded onto the application using a *loadSupervisors* method and stored for easy allocation to PG Research students.

- **Module**

An object of type Module is used in Student class to store the details of the modules assigned to the student by the Department.A HashSet on the supervisor module code is used to ensure no duplicates are loaded onto the system. In addition, in order to facilitate comparison between two Module objects, equals and hashCode method have been overridden. \

A list of objects of type Module is maintained by the Department class to store the list of all available Modules. The list of modules is available to the Department in the form of a CSV file which is then loaded onto the application using a *loadModules* method and stored for easy allocation to students.

Validation

Validation of all the inputs, i.e., the student data is done at the entry point of the application using a **Validate class**. A method call to the validate class at the entry point in the application ensures that none of the essential attributes of student is invalid. This ensures that no unexpected errors occur in the application.

How it Works

- **Registration of Student**

An object of Student class is created with the minimum details populated including student name, date of birth and student type. This object is then passed onto the *registerStudents* method in Department class.

After validation of the student data, the *registerStudent* method of the Department class initiates the logic to generate studentId and smartCard for the student object and is then assigned to it. If it's a Postgraduate Research Student, a supervisor is assigned randomly to the student from the list of available supervisors.

Finally, *registerStudents* method of the super class of Department, that is, the AbstractUniversity class is invoked which stores the details of the student object onto a static map of students and completes the registration process.

- **Updation of Student**

An object of Student data to be updated and its student Id is passed onto the *amendStudentData* method in Department class.

After validation of the student data and its student Id, the *amendStudentData* method of the parent AbstractUniversity is invoked where the old data is removed and the new data is inserted for the corresponding student Id.

- **Termination of Student**

A studentId is passed onto the *terminateStudent* method in Department class. After validating that it’s a valid Student Id, the *terminateStudent* method in AbstractUniversity is invoked where the student data is removed from the application.