Programming 3 Assignment 1

Assignment Overview

- The specifications associated with this assignment will incorporate the Model View Controller design pattern and specifically ASP.NET MVC to begin the process building a CRUD application along with a relational database. This assignment will focus on the Model component of the MVC pattern.
- Work associated with this assignment will reinforce the student's learning in the following areas:
    - ASP.NET MVC – specifically Models
    - Data annotations
    - Object Relational Model

Evaluation

- The evaluation of this assignment will be in the form of a Code Review. During the Code Review evaluation will be based on:
    - Comments
    - Documentation
    - Adherence to Standards
    - Coding choices (i.e. refactoring, no repeated code, readability etc.)
    - Implementation of Requirements

Programming 3 Assignment 1

Specifications

MVC Project

- Create an ASP.NET MVC Web Application project called **BITCollege_FL** where **F** represents your First Name and **L** represents your Last Name (Example: ***BITCollege_TJ*** would be the name of Tom Jones' project).

Utility Project

- Add the given Utility Project to the assignment solution. **Ensure the Utility project folder is in your solution directory.**
- Below is a Class Diagram describing the files contained within the Utility Project.

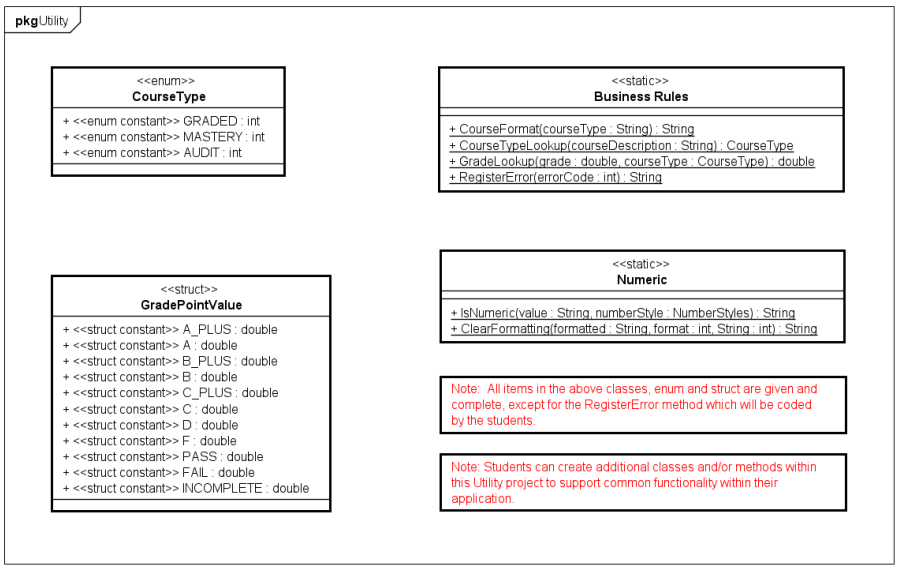

Figure 1 - Given Utility Project - no code required at this time (electronic version in the Assignment 1 Files folder)

- The Utility Project contains classes, as well as an enum and a struct containing business rules that will assist in completing this and future assignments.
    - Familiarize yourself with this project.
    - Feel free to add common functionality to this project at any time.
- Create a project reference from within the ***BITCollege_FL*** project to the Utility project.

Models

- In the Models folder of the ***BITCollege_FL*** project, create a Class file called SchoolModels.cs.
    - Remove the SchoolModels class declaration within this file.
- Within the SchoolModels.cs define the project models based on the following class diagram:

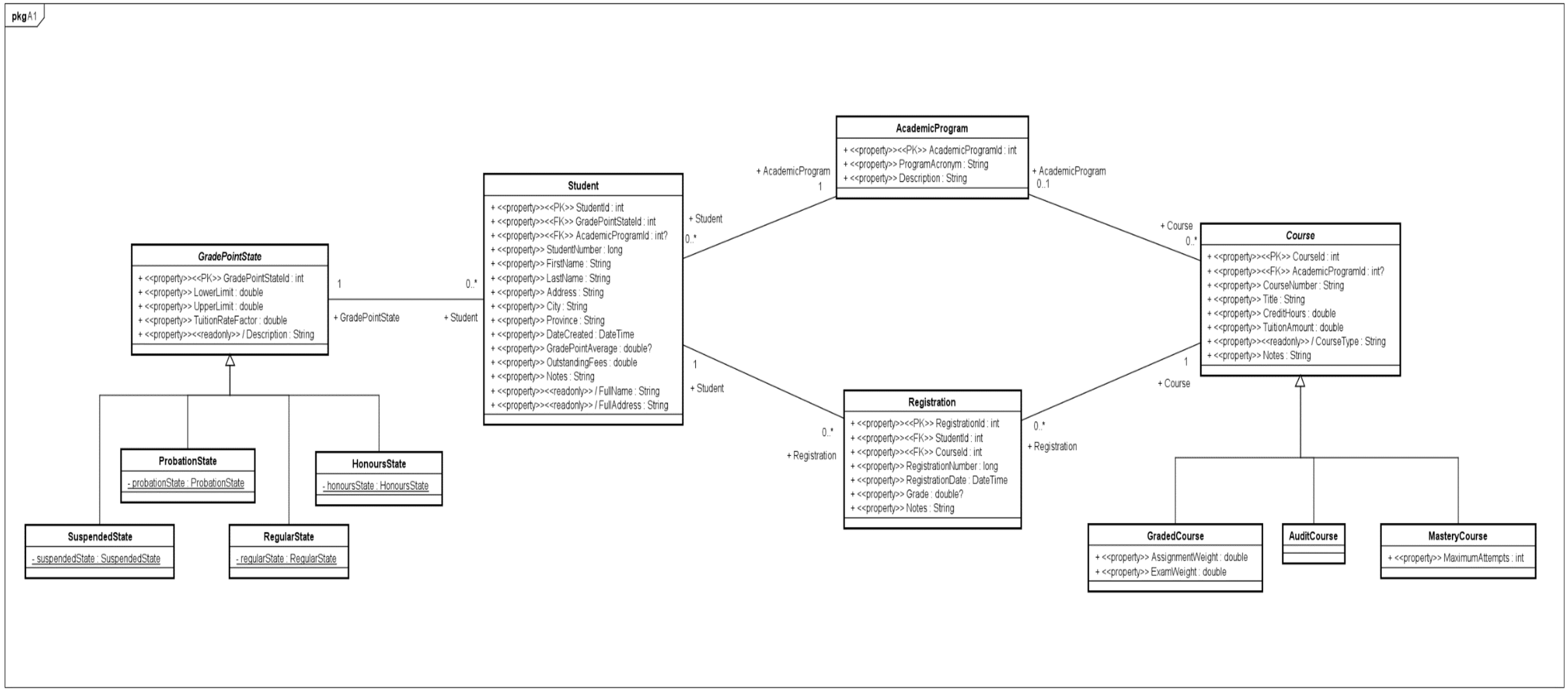

Figure 2 (electronic version in the Assignment 1 Files folder)

Programming 3 Assignment 1

Diagram Interpretation:

- Attributes defined using the <<property>> stereotype are Auto-Implemented and are therefore shown in the diagram as public.
- Attributes defined without the <<property>> stereotype are instance variables.
- The Navigation Properties must be defined such that they support the Entity Framework concept of Lazy Loading.
- Properties whose names are preceded by a forward slash (/) contain values derived from other properties. Do not include the forward slash in the name of the property.
- Attributes defined with the <<readonly>> stereotype must have the get method implemented. Implementation requirements are defined below.
- Attributes defined with a <<PK>> stereotype are Primary Keys.
- Attributes defined with a <<FK>> stereotype are Foreign Keys.
- Specific instructions for each Model are defined below.

Programming 3 Assignment 1

Student Model

- Define the Student Model based on the Class Diagram including any requirements for Navigation Properties.
- Ensure the Student Model depicted in the class diagram above satisfies the following:

    - StudentId must be automatically generated as an identity field

    - GradePointStateId is required.
    - GradePointStateId is a foreign key that corresponds to the navigation property depicted in the class diagram above.

    - The ***int?*** type for AcademicProgramId defines this primitive type as nullable. This allows for a student to exist but not be assigned to a specific academic program
    - AcademicProgramId is a foreign key that corresponds to the navigation property depicted in the class diagram.

    - StudentNumber is required.
    - The StudentNumber value must be within the range of 10000000 and 99999999.
    - The Heading/Label for StudentNumber is “Student Number”
        - Use a line feed (\n) in place of any word spacing. E.g. Student\nNumber. This will be re-visited in assignment 2.

    - FirstName is required.
    - The Heading/Label for FirstName is “First Name”
        - Use a line feed (\n) in place of any word spacing.

    - LastName is required.
    - The Heading/Label for LastName is “Last Name”
        - Use a line feed (\n) in place of any word spacing.

    - Address is required.

    - City is required.

    - Province is required.
    - Province data must use the following regular expression:
        - "^(N[BLSTU]|[AMN]B|[BQ]C|ON|PE|SK|YT)"
        - An appropriate error message must be displayed if a valid Canadian province code is not entered.

    - DateCreated is required.
    - The Heading/Label for DateCreated is “Date”

- DateCreated data can be any short date format but must supress the timestamp.

- The ***double?*** type for GradePointAverage defines this primitive type as nullable. This allows for a student's GradePointAverage to remain NULL until the completion of at least one course.

- The Heading/Label for GradePointAverage is "Grade Point Average"
  - Use a line feed (\n) in place of any word spacing.
- GradePointAverage data should be displayed to 2 decimal places.
- GradePointAverage, when entered, must be between 0 and 4.5.

- OutstandingFees is required.
- The Heading/Label for OutstandingFees is "Fees"
- OutstandingFees data should be displayed to 2 decimal places in currency format.

- Notes has no additional requirements

- The Heading/Label for FullName is "Name"
- The FullName accessor should return the full name in the format of:
  "FirstName LastName"

- The Heading/Label for FullAddress is "Address"
- The FullAddress accessor should return the full address in the format of:
  "Address City Province"

AcademicProgram Model

- Define the AcademicProgram Model based on the Class Diagram including any requirements for Navigation Properties.
- Ensure the AcademicProgram Model depicted in the class diagram above satisfies the following:

  - AcademicProgramId must be automatically generated as an identity field

  - ProgramAcronym is required.
  - The Heading/Label for ProgramAcronym is "Program"

  - Description is required.
  - The Heading/Label for Description is "Program Name"
    - Use a line feed (\n) in place of any word spacing.

Programming 3 Assignment 1

GradePointState Model

- Define the GradePointState Model based on the Class Diagram including any requirements for Navigation Properties.
- Ensure the GradePointState Model depicted in the class diagram above satisfies the following:

    - GradePointStateId must be automatically generated as an identity field.
    - GradePointStateId must have a key annotation.

    - LowerLimit is required.
    - LowerLimit data should be displayed to 2 decimal places.
    - The Heading/Label for LowerLimit is “Lower Limit”
        - Use a line feed (\n) in place of any word spacing.

    - UpperLimit is required.
    - UpperLimit data should be displayed to 2 decimal places.
    - The Heading/Label for UpperLimit is “Upper Limit”
        - Use a line feed (\n) in place of any word spacing.

    - TuitionRateFactor is required.
    - TuitionRateFactor data should be displayed to 2 decimal places.
    - The Heading/Label for TuitionRateFactor is “Tuition Rate Factor”
        - Use a line feed (\n) in place of any word spacing.

    - The Heading/Label for Description is “State”
    - The readonly Description property will return the name of the GradePointState subtype (E.g. SuspendedState, ProbationState, etc…) of the current instance
        - For now, return ***GetType().Name***. This will be re-visited in assignment 2.

SuspendedState Model

- Define the SuspendedState Model based on the Class Diagram.
    - No additional annotation requirements for the SuspendedState Model.

ProbationState Model

- Define the ProbationState Model based on the Class Diagram.
    - No additional annotation requirements for the ProbationState Model.

RegularState Model

- Define the RegularState Model based on the Class Diagram.
    - No additional annotation requirements for the RegularState Model.

HonoursState Model

- Define the HonoursState Model based on the Class Diagram.
    - No additional annotation requirements for the HonoursState Model.

Course Model

- Define the Course Model based on the Class Diagram including any requirements for Navigation Properties.
- Ensure the Course Model depicted in the class diagram above satisfies the following:

    - CourseId must be automatically generated as an identity field.
    - CourseId must have a key annotation.

    - The ***int?*** type for AcademicProgramId defines this primitive type as nullable. This allows for a course to exist but not be assigned to a specific program.
    - AcademicProgramId is a foreign key that corresponds to the navigation property depicted in the class diagram above.

    - CourseNumber is required.
    - The Heading/Label for CourseNumber is “Course Number”
        - Use a line feed (\n) in place of any word spacing.

    - Title is required.

    - CreditHours is required.
    - CreditHours data should be displayed to 2 decimal places.
    - The Heading/Label for CreditHours is “Credit Hours”
        - Use a line feed (\n) in place of any word spacing.

    - TuitionAmount is required.

- TuitionAmount data should be displayed to 2 decimal places in currency format.
- The Heading/Label for TuitionAmount is “Tuition”

- The Heading/Label for CourseType is “Course Type”
  - Use a line feed (\n) in place of any word spacing.

- The readonly CourseType property will return the name of the Course subtype (E.g. GradedCourse, AuditCourse, etc…) of the current instance
  - For now, return ***GetType().Name***. This will be re-visited in assignment 2.

- Notes has no additional requirements

GradedCourse Model

- Define the GradedCourse Model based on the Class Diagram.
- Ensure the GradedCourse Model depicted in the class diagram above satisfies the following:

  - AssignmentWeight is required.
  - The Heading/Label for AssignmentWeight is “Assignments”
  - AssignmentWeight data should be displayed to 2 decimal places in percent format.

  - ExamWeight is required.
  - The Heading/Label for ExamWeight is “Exams”
  - ExamWeight data should be displayed to 2 decimal places in percent format.

MasteryCourse Model

- Define the MasteryCourse Model based on the Class Diagram.
- Ensure the MasteryCourse Model depicted in the class diagram above satisfies the following:

  - MaximumAttempts is required.
  - The Heading/Label for MaximumAttempts is “Maximum Attempts”
    - Use a line feed (\n) in place of any word spacing.

Programming 3 Assignment 1

AuditCourse Model

- Define the AuditCourse Model based on the Class Diagram.
    - No additional annotation requirements for the AuditCourse Model.

Registration Model

- Define the Registration Model based on the Class Diagram including any requirements for Navigation Properties.
- Ensure the Registration Model depicted in the class diagram above satisfies the following:

    - RegistrationId must be automatically generated as an identity field

    - StudentId is required.
    - StudentId is a foreign key that corresponds to the navigation property depicted in the class diagram above.

    - CourseId is required.
    - CourseId is a foreign key that corresponds to the navigation property depicted in the class diagram above.

    - RegistrationNumber is required.
    - The Heading/Label for RegistrationNumber is “Registration Number”
        - Use a line feed (\n) in place of any word spacing.

    - RegistrationDate is required.
    - The Heading/Label for RegistrationDate is “Date”
    - RegistrationDate data can be any short date format but must supress the timestamp.

    - The ***double?*** type for Grade defines this primitive type as nullable. This allows for a registration record to have a NULL grade until the completion of the course. This will prevent incomplete courses from being included in the GradePointAverage calculation.
    - Ensure that when the Grade is null, it is displayed as “Ungraded”.
        - **Hint**: Look at the DisplayFormat overloads.
    - The Grade value (when entered) must be within the range of 0 and 1.

    - Notes has no additional requirements

Test

- Desk Check (proofread) your solution to ensure it meets all of the requirements as defined above.

Backup

- Backup your Solution.

Evaluation: In Person/Hybrid Delivery

- This assignment must be submitted on or before the due date/time specified in Learn.