# Assignment 4

## 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:

o ASP.NET MVC – specifically Models o Data annotations o 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: o Comments o Documentation o Adherence to Standards

* Coding choices (i.e. refactoring, no repeated code, readability etc…)
* Implementation of Requirements o Verbal discussion of code

1

# Assignment 4

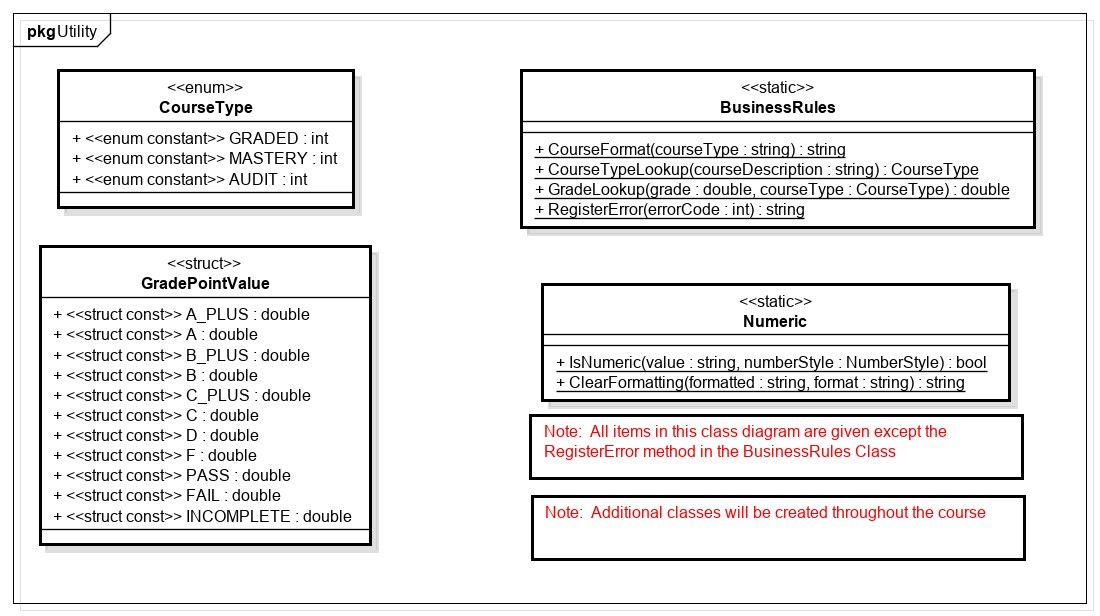
*Specifications*

## MVC Project

 Create an ASP.NET MVC 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) based on the following specifications:

## 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 reference from within the ***BITCollege\_FL*** project to the Utility project.

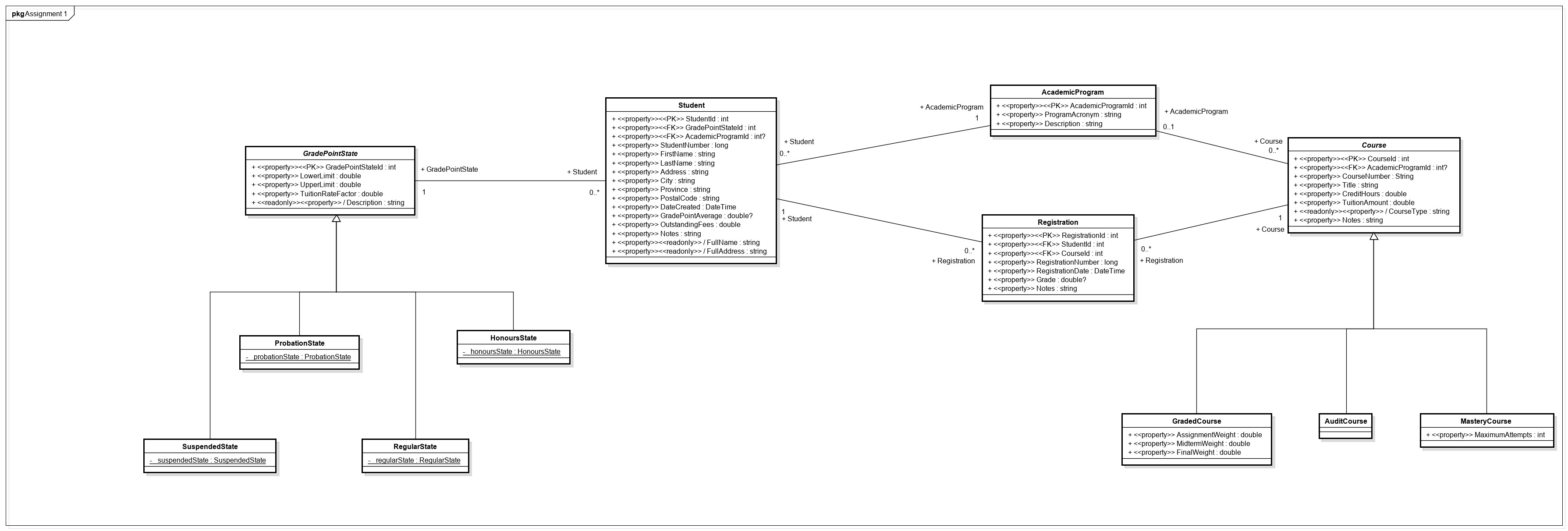
## Models

* In the Models folder of the ***BITCollege\_FL*** project, create a Class file called SchoolModels.cs.

o Remove the SchoolModels class declaration within this file.

* Within the SchoolModels.cs define the project models based on the following class diagram:

# Assignment 4



*Figure 2 (electronic version in the Assignment 4 Files folder)*

3

*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 Navigational 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.

## 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. o 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. o The StudentNumber value must be within the range of 10000000 and 99999999. o The Heading/Label for StudentNumber is “Student Number”
      * When displayed in an Index Views, the StudentNumber field must have a *heading* that includes a line feed.
      * When displayed in all other View types, the StudentNumber field must have a *label* that includes a space.

* + - FirstName is required. o FirstName data must be between 1 and 35 characters.
    - The Heading/Label for FirstName is “First Name”
      * When displayed in an Index Views, the FirstName field must have a *heading* that includes a line feed.
      * When displayed in all other View types, the FirstName field must have a *label* that includes a space.

* + - LastName is required. o LastName data must be between 1 and 35 characters.
    - The Heading/Label for LastName is “Last Name”
      * When displayed in an Index Views, the LastName field must have a *heading* that includes a line feed.
      * When displayed in all other View types, the LastName field must have a *label* that includes a space.

* + Address is required. o Address data must be between 1 and 35 characters. o City is required.
  + City data must be between 1 and 35 characters.

* + Province is required. o Province data must be 2 uppercase letters based on the valid province code abbreviations found [here.](http://www.comeexplorecanada.com/abbreviations.php) An appropriate error message must be displayed if a valid Canadian province code is not entered.

* + PostalCode is required.
  + PostalCode data must be **7** characters using a format of “*A9A 9A9*”.
    - Example: R3B 1K9
    - **Note**: Valid Canadian Postal codes do not include the letters D, F, I, O, Q or U, and the first position also does not make use of the letters W or Z.
    - An appropriate error message must be displayed if the format does not match this requirement. o The Heading/Label for PostalCode is “Postal Code”
    - When displayed in an Index Views, the PostalCode field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the PostalCode field must have a *label* that includes a space.

* DateCreated is required.
* The Heading/Label for DateCreated is “Date Created”
  + When displayed in an Index Views, the DateCreated field must have a *heading* that includes a line feed.
  + When displayed in all other View types, the DateCreated field must have a *label* that includes a space.
* 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”
  + When displayed in an Index Views, the GradePointAverage field must have a *heading* that includes a line feed following the word Point.
  + When displayed in all other View types, the GradePointAverage field must have a *label* that includes a space between each word. o 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 “Outstanding Fees”
  + - When displayed in an Index Views, the OutstandingFees field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the OutstandingFees field must have a *label* that includes a space.
* OutstandingFees data should be displayed to 2 decimal places in currency format.

* Notes has no additional requirements

* The Heading/Label for FullName is “Name” o FullName should return the full name in the format of:

*“FirstName LastName”*

* The Heading/Label for FullAddress is “Address” o FullAddress should return the full address in the format of: *“Address City Province, PostalCode”*

## 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”
    - When displayed in an Index Views, the Description field must have a *heading* (Program Name) that includes a line feed.
    - When displayed in all other View types, the Description field must have a *label* (Program Name) that includes a space.

## 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. o LowerLimit data should be displayed to 2 decimal places.
  + The Heading/Label for LowerLimit is “Lower Limit”
    - When displayed in an Index Views, the LowerLimit field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the LowerLimit field must have a *label* that includes a space.

* + UpperLimit is required. o UpperLimit data should be displayed to 2 decimal places.
  + The Heading/Label for UpperLimit is “Upper Limit”
    - When displayed in an Index Views, the UpperLimit field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the UpperLimit field must have a *label* that includes a space.

* + TuitionRateFactor is required. o TuitionRateFactor data should be displayed to 2 decimal places.
  + The Heading/Label for TuitionRateFactor is “Tuition Rate Factor”
    - When displayed in an Index Views, the TuitionRateFactor field must have a *heading* that includes a line feed following the word Rate.
    - When displayed in all other View types, the TuitionRateFactor field must have a *label* that includes a space between each word.

* + The Heading/Label for Description is “Grade Point State”
    - When displayed in an Index Views, the Description field must have a *heading* (Grade Point State) that includes a line feed following the word Point.
    - When displayed in all other View types, the Description field must have a *label* (Grade Point State) that includes a space between each word.

The readonly Description get method will return the name of the

GradePointState subtype (E.g. SuspendedState, ProbationState, etc…) of the current instance

* + - The get method will override the default get behavior by returning the ***GetType().Name*** property of the current instance
    - The GetType().Name property will provide the name in memory of the current instance (e.g. ProbationState### (where ### represents a unique numeric value allowing the current instance to have a unique name in memory)).
    - Ensure that only the text prior to the word “State” is returned. That is, instead of “ProbationState###”, return “Probation”.

## SuspendedState Model

 Define the SuspendedState Model based on the Class Diagram.

o No additional annotation requirements for the SuspendedState Model.

## ProbationState Model

 Define the ProbationState Model based on the Class Diagram.

o No additional annotation requirements for the ProbationState Model.

## RegularState Model

 Define the RegularState Model based on the Class Diagram.

o No additional annotation requirements for the RegularState Model.

## HonoursState Model

 Define the HonoursState Model based on the Class Diagram.

o 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. o CourseId must have a key annotation.

* + AcademicProgramId is a foreign key that corresponds to the navigation property depicted in the class diagram above. o 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.

* + CourseNumber is required.
  + The Heading/Label for CourseNumber is “Course Number”
    - When displayed in an Index Views, the CourseNumber field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the CourseNumber field must have a *label* that includes a space.

* Title is required.

* CreditHours is required. o CreditHours data should be displayed to 2 decimal places.
* The Heading/Label for CreditHours is “Credit Hours”
  + When displayed in an Index Views, the CreditHours field must have a *heading* that includes a line feed.
  + When displayed in all other View types, the CreditHours field must have a *label* that includes a space.

* TuitionAmount is required. o TuitionAmount data should be displayed to 2 decimal places in currency format. o The Heading/Label for TuitionAmount is “Tuition Amount”
  + When displayed in an Index Views, the TuitionAmount field must have a *heading* that includes a line feed.
  + When displayed in all other View types, the TuitionAmount field must have a *label* that includes a space.

* The Heading/Label for CourseType is “Course Type”
  + - When displayed in an Index Views, the CourseType field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the CourseType field must have a *label* that includes a space.

The readonly CourseType get method will return the name of the Course subtype (E.g. GradedCourse, AuditCourse, etc…) of the current instance

* + - The get method will override the default get behavior by returning the ***GetType().Name*** property of the current instance
    - The GetType().Name property will provide the name in memory of the current instance (e.g. GradedCourse### (where ### represents a unique numeric value allowing the current instance to have a unique name in memory)).
    - Ensure that only the text prior to the word “Course” is returned. That is, instead of “GradedCourse###”, return “Graded”.
* **You may notice that this requirement is very similar to the Description Readonly Property requirement for the GradePointState class. This is a great opportunity to refactor such that the duplicate code is eliminated. Consider creating a static method in the Utility project to accomplish this.**

* 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 “Assignment Weight”
    - When displayed in an Index Views, the AssignmentWeight field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the AssignmentWeight field must have a *label* that includes a space.
  + AssignmentWeight data should be displayed to 2 decimal places in percent format.

* + MidtermWeight is required.
  + The Heading/Label for MidtermWeight is “Midterm Weight”
    - When displayed in an Index Views, the MidtermWeight field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the MidtermWeight field must have a *label* that includes a space.
  + MidtermWeight data should be displayed to 2 decimal places in percent format.

* + FinalWeight is required.
  + The Heading/Label for FinalWeight is “Final Weight”
    - When displayed in an Index Views, the FinalWeight field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the FinalWeight field must have a *label* that includes a space.
  + FinalWeight 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”
    - When displayed in an Index Views, the MaximumAttempts field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the MaximumAttempts field must have a *label* that includes a space.

## AuditCourse Model

 Define the AuditCourse Model based on the Class Diagram.

o 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. o 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”
    - When displayed in an Index Views, the RegistrationNumber field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the RegistrationNumber field must have a *label* that includes a space.

* + RegistrationDate is required.
  + The Heading/Label for RegistrationDate is “Registration Date”
    - When displayed in an Index Views, the RegistrationDate field must have a *heading* that includes a line feed.
    - When displayed in all other View types, the RegistrationDate field must have a *label* that includes a space.
  + 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 your solution to ensure it meets all of the requirements as defined above.

*Backup*

* Backup your Solution.

## Evaluation

 This assignment must be evaluated on or before the due date specified in Learn.

* The Due Time for this assignment is based on your selection of an evaluation time slot
* It is the student’s responsibility to select an evaluation time slot o It is the student’s responsibility to be available for evaluation during their evaluation time slot
* If the assignment has been deemed incorrect/incomplete the student must make the appropriate modifications in order to proceed onto the next assignment

 In those cases, a follow-up evaluation may be necessary