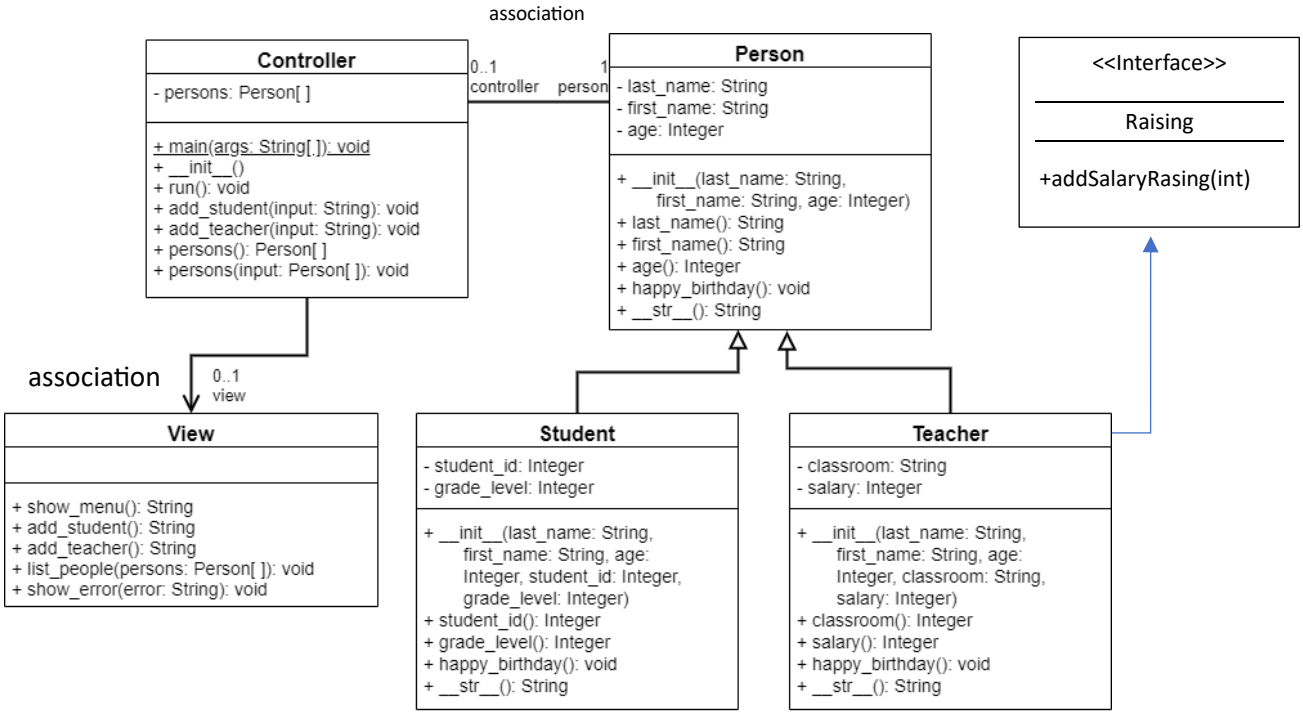
HW3



The purpose of each method will be further described below.

Note: all input / OUTPUT Should be from Files

`Person` Class (Abstract class)

- `__init__()` - constructor that initializes all attributes based on parameters
- `last_name()` - getter for `last_name` attribute—it should be implemented as a property
- `get_first_name()` - getter for `first_name` attribute—it should be implemented as a property
- `get_age()` - getter for `age` attribute—it should be implemented as a property
- `happy_birthday()` - method to increase person's `age` attribute by 1
- `__str__()` - method that overrides the built-in `Object` class `__str__()` method. It should return a string in the form `"first_name last_name: age"`

`Student` Class

- `__init__()` - constructor that initializes all attributes (including in super class) based on parameters
- `student_id()` - getter for `student_id` attribute—it should be implemented as a property
- `grade_level()` - getter for `grade_level` attribute—it should be implemented as a property
- `happy_birthday()` - method to increase student's `age` and `grade_level` attribute by 1
- `__str__()` - method that overrides the built-in `Object` class `__str__()` method. It should return a string in the form `"first_name last_name: age (student_id - grade_level)"`

`Teacher` Class

- `__init__()` - constructor that initializes all attributes (including in super class) based on parameters
- `classroom()` - getter for `classroom` attribute—it should be implemented as a property
- `salary()` - getter for `salary` attribute—it should be implemented as a property
- `happy_birthday()` - method to increase teacher's `age` by 1 and `salary` attribute by 1000
- `__str__()` - method that overrides the built-in `Object` class `__str__()` method. It should return a string in the form `"first_name last_name: age (classroom - $salary)"`

`View` Class

- `show_menu()` - a method to show a menu of options to the user. The user should be prompted to input **exactly** one of the options arrayed below, which is returned as a String. The wording of the menu is up to you. The method should return whatever was input by the user, without any error checking (that is done in the Controller)
    - "add student" - add a student
    - "add teacher" - add a teacher
    - "array people" - array the people
    - "exit" - exit the program
- `add_student()` - a method to add a new student to the system. The user should input a array of parameters for each attribute as they are arrayed in the constructor for `Student`, separated by spaces. The wording of the prompt is up to you. The method should return whatever was input by the user, without any error checking (that is done in the Controller)
    - Example: "Smith John 25 123456 13"
- `add_teacher()` - a method to add a new teacher to the system. The user should input a array of parameters for each attribute as they are arrayed in the constructor for `Teacher`, separated by spaces. The wording of the prompt is up to you. The method should return whatever was input by the user, without any error checking (that is done in the Controller)
- `array_people()` - a method to array all `Person` objects in the `persons` array given as a parameter. Each one should be prefixed by an index starting at 0, incrementing by one for each `Person` in the array.
    - Example: "0) Smith John: 25 (geology - $1000)
- `show_error()` - a method to display an error to the user. The parameter `error` should be printed to the screen, prefixed by "Error: "

Hint: use `sys.stdin.readline()` to read an entire line of input anywhere in your code. Don't forget to import `sys` as well!

`Controller` Class

- `main()` - the main method for this program. It should simply instantiate a new instance of the Controller class, and then call the `run()` method of that object.
- `__init__()` - the constructor for the Controller object. It initialize the `persons` attribute to an empty array, as well as a `View` object stored in the `view` attribute.
- `run()` - this method consists of a loop that will execute the program until it is terminated. It will call the `showMenu()` method of the view to show a menu to the user (see above). Finally, it will parse the string returned by the call to `showMenu()` and call additional appropriate methods in the `Controller` or `View` class to complete the operation. If the user inputs "exit" then it should terminate. Otherwise, the program will repeatedly display the menu to the user until "exit" is chosen. If at any time the user provides input that cannot be properly parsed, the controller should call the `showError()` method in the `View` class and restart the process (loop back to the beginning) by showing the menu again.
- `add_student()` - this method will receive the string input by the user from the `add_student()` method in `View`, parse the input, and call the appropriate methods to create a new `Student` object and add it to the first empty slot in the `persons` array.

- `add_teacher()` - this method will receive the string input by the user from the `add_teacher()` method in `View`, parse the input, and call the appropriate methods to create a new `Teacher` object and add it to the first empty slot in the `persons` array.
- `persons()` - these methods are a getter and setter for the `persons` attribute. They should be implemented as a property. It is for testing purposes only.

Interface Rasing: this interface has one abstract method called addRaisingSalary(int percent) that implement in inherited class that will increase the salary by percent received in the method