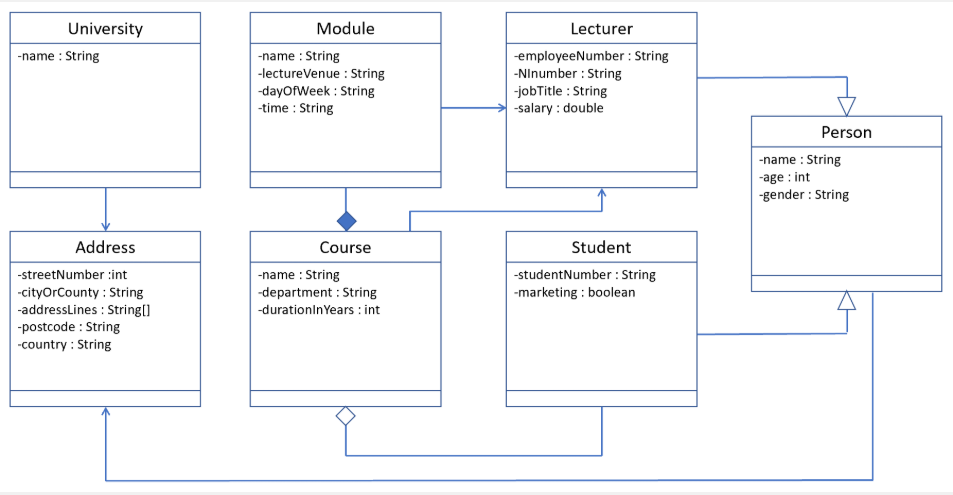
Code Kata: University   
K1602155

Although the Kata in and of itself is quite easy and straightforward, it does require a lot of different understanding and it’s a nice demonstration of that. It’s probably one of the most useful and simple Katas to show how Object Orientation, encapsulation and relationships all work together, as well as showing the usefulness of UML when paired with programming.

The program does not display or do much apart from following this UML diagram and creating an empty data model to this specification: 

By empty, I mean that every attribute and relationship is created and ready to accept inputs, but there are currently no inputs and the Main class does nothing. The only thing this Kata does is create the data model with the correct attributes, encapsulation, the correct associations etc. I will run through each class and quickly describe what it does.

Main – empty, just a place holder so the program can run. This could be used for making the program actually do something useful, or for testing/debugging the code and making sure the inputs are correct.  
University – apart from a single private string attribute and getters/setters for it, it also has an instance of the class “Address” so every “university” has an “address”.  
Address – contains 5 private attributes and their getters/setters, one of which is an array.  
Module – contains 4 private attributes and one instance of class “Lecturer”, and their getters/setters. Each “Module” has a “Lecturer”  
Course – this class is where it starts to get interesting, as this class has 3 private attributes of its own, as well as two arrays, one of class Module, and another of class Student, as well as an instance of class Lecturer. This means that every “Course” has multiple instances of “Module” and “Student”, and one “Lecturer”. And all of them of course have their getters/setters  
Lecturer- this class has 4 private attributes and getters/setters. However, the class itself extends another class, the “Person” class. What this means, is essentially every “Lecturer” class is a “Person” class with the extra 4 attributes. So every “Lecturer” will also have an “age”, a “name” and a “gender”, however the class itself does not have those attributes because they are ‘extended’ from the “Person” class. Its an example of inheritance and it’s a good way of ensuring efficient code and reducing repetition.  
Student – this class has 2 private attributes and getters/setters. It also ‘extends’ the “Person” class. Just like every “Lecturer” is a “Person”, every “Student” is a “Person”.  
Person – this class has 3 private attributes, and an instance of “Address”, so every “Person” has an “Address”. As mentioned just above, this class is a super(parent) class for 2 of the sub(child) classes, “Lecturer” and “Student”.

When writing this out, a lot of it sounds like common sense because of the context of this Kata. Obviously, a lecturer and student are both people, and a course is made up of multiple modules and students and (in this example) a single lecturer. But this may not always be so, and it was very useful to understand exactly what inheritance and association mean in plain English. It was also very useful to implement the data model from the UML diagram.

Just in case I missed anything out, every attribute used in every class is private, with their own getters/setters, and so are any instances of classes used, as well as the “Module” and “Student” arrays in “Course”