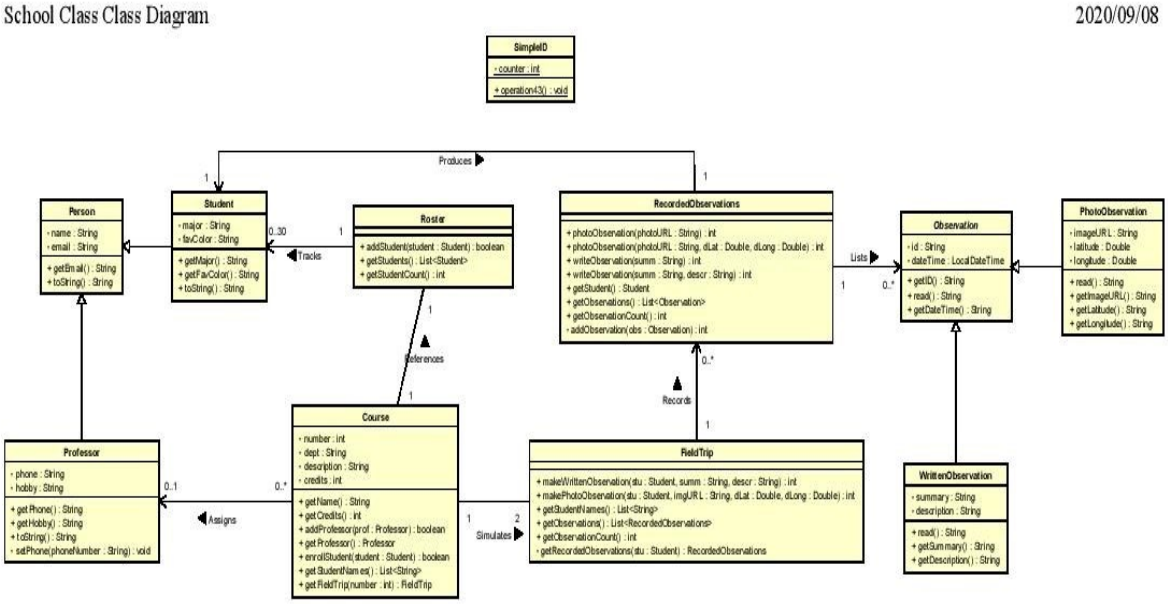
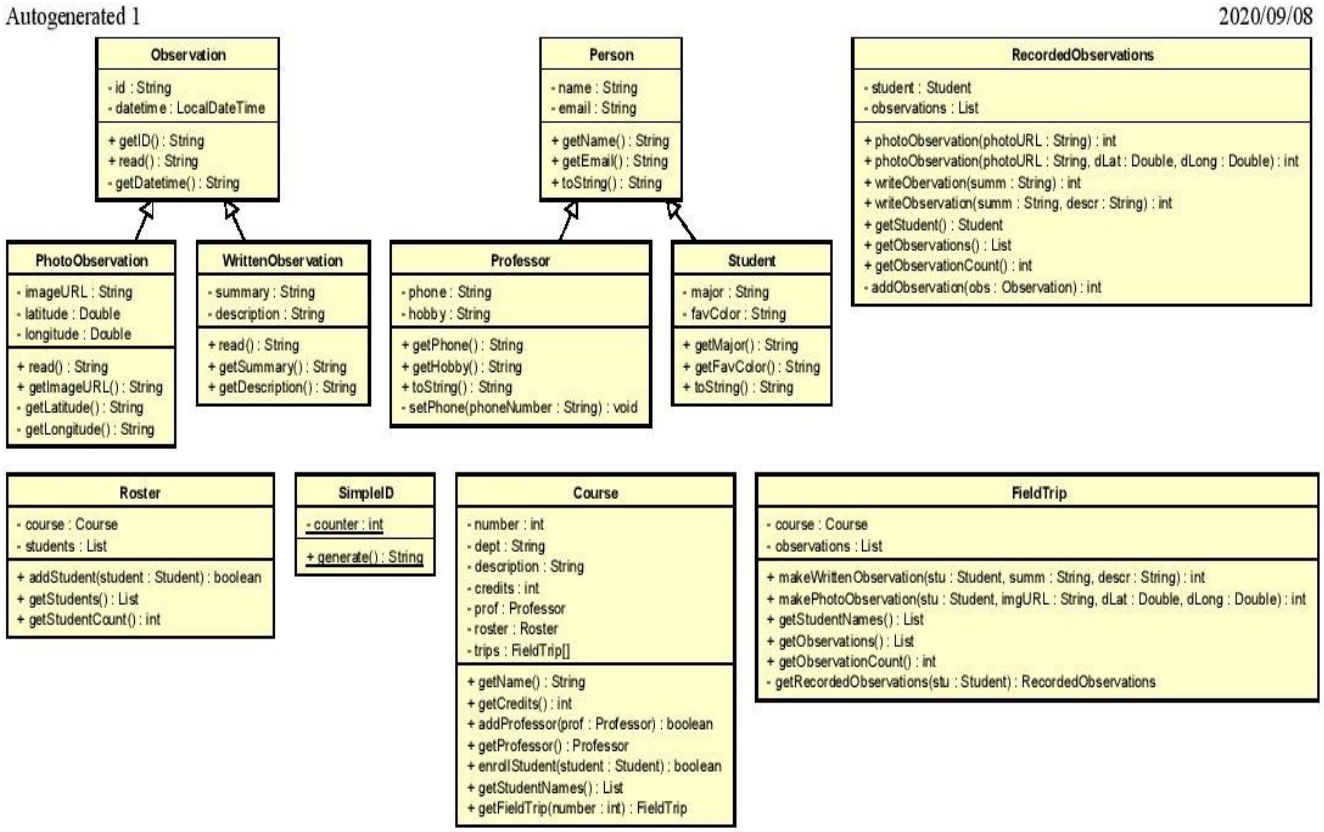
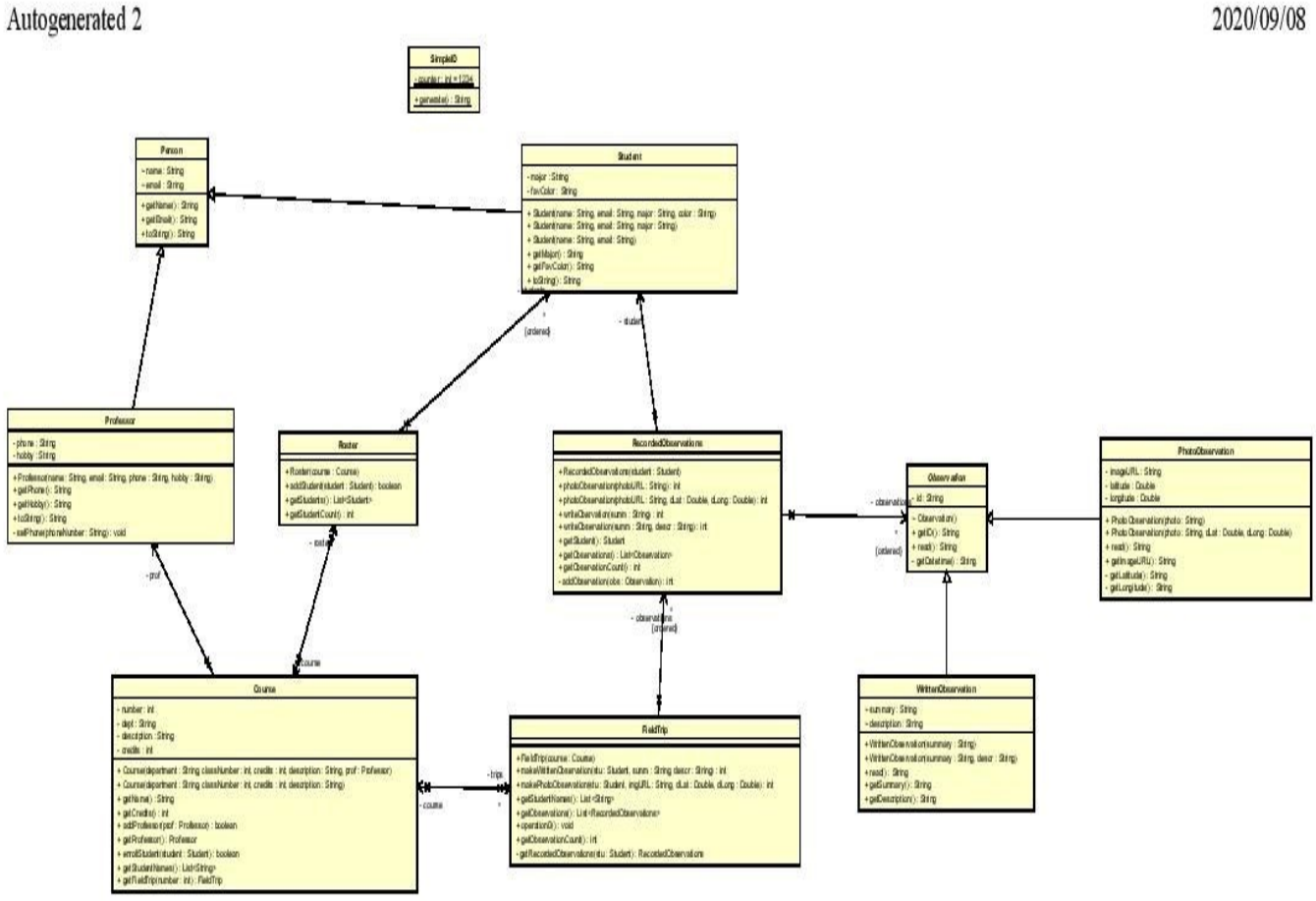
Task 1: Manual Reverse Engineering



Task 2.1: Automatic Reverse Engineering 1



Task 2.2: Automatic Reverse Engineering 2



1. There main difference between the 3 diagrams is the level of abstraction used when representing the existing system. The third diagram (2nd automatically generated diagram) is the least abstract representation of the system due to its inclusion of constructor methods in the behavior sections of classes as well as its inclusion of roles.

However, this diagram lacks multiplicities in its associations, which is a different approach to abstraction when compared to the manually generated diagram, which utilizes multiplicities as well as reading directions, which helps emphasize relevant details of the structure of the system while remaining legible. The first automatically generated diagram is the most abstract representation of the system. It only displays the existing classes with their behaviors and attributes, as well as any existing generalizations. No associations are displayed, which can make it difficult in determining how the system components might interact with each other. Instead, attributes are utilized to represent the relations between classes, which can make the visualization less abstract at the same time.

2. Despite the visual differences between the three diagrams, they still represent the same system based on the common inclusion of visual representations of the same classes. Because the same classes are represented in each diagram, it is clear that the same system is being described in each diagram. The main differences between the diagrams, as indicated previously, are how much information is omitted when representing the system and how the same information across the three diagrams is represented. Despite these visual differences, they tell the same essential story of the system structure.
3. An association is essentially the same as an attribute because both describe the visibility that classes have between each other. Having an attribute representing an array of students would be the same as using multiplicities and associations to indicate that a

particular class is associated with a collection of multiple students since both indicate that a particular class has access to a collection belonging to another class.

Task 3: Automatic Forward Engineering

Export2-1 seems a bit funky with classes like FieldTrip[].java. Additionally, some types are not represented as originally, using inappropriate keywords like final for certain attributes that may need to be modified. This is not a perfect template.

Export1 includes new classes that should be part of the standard libraries imported into Java, which is wasteful for template code. Additionally, some associations are represented as arrays instead of lists, as in the original code. However, the representation is better overall than Export2-1 due to appropriate keywords used in variable declaration.

Export2-2 also uses differing representations of associations, such as array collections instead of list collections. Additionally, due to the automatic generation process, there exist random stub methods with nondescriptive method names, which is wasteful if used in the context of a template.

Overall, the best export would be Export1, the code generation from the manual reverse engineering process since it best represents the original code structure in terms of associations and keywords while refraining from random generation of nondescriptive method stubs. However, it would benefit the template if the diagram included stubs for constructors and refrained from generating classes of existing library packages.