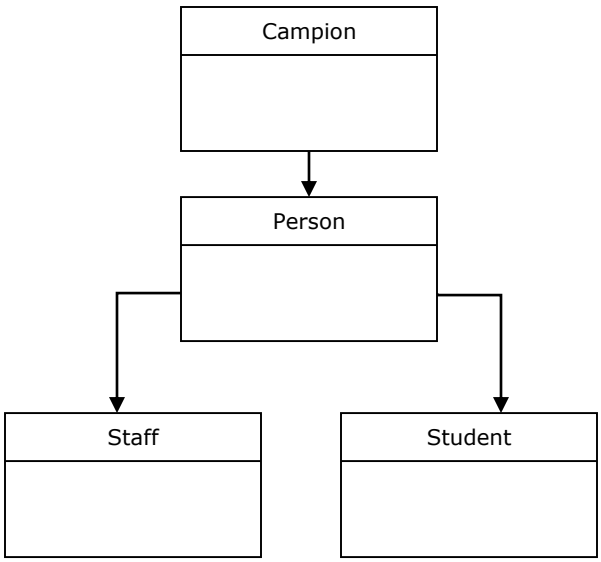
DIAGRAMMING CLASSES WITH UML

Programmers will often draw class diagrams to illustrate the relationship among the classes that make up their program(s). The following diagram, for example, is a simple class diagram for a program that has four classes. The rectangles represent the classes, and the arrows represent the relationships among the classes



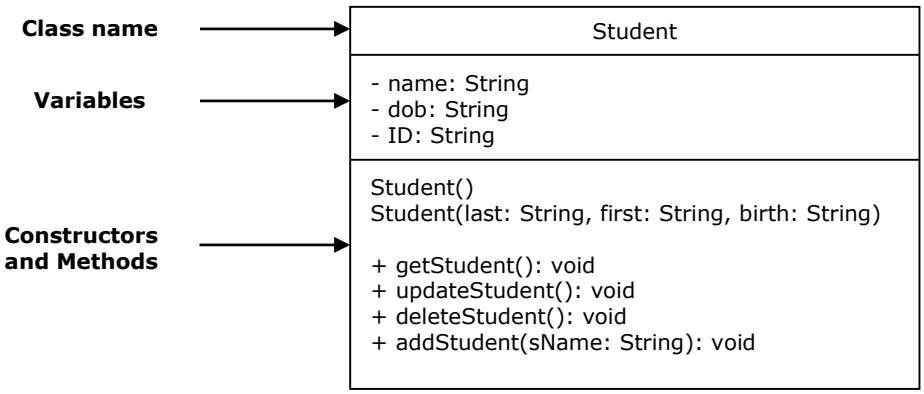
Programmers use a diagramming standard called **UML**, which stands for **Unified Modeling Language**. A UML diagram helps programmers organize the programming design process in a way that clients, analysts and other programmers can understand and agree on. The following are a list of just some of the benefits of using UML:

- Your software system is professionally designed and documented before it is coded. You will know exactly what you are getting, in advance.
- Since system design comes first, reusable code is easily spotted and coded with the highest efficiency. You will have lower development costs.
- Logic 'holes' can be spotted in the design drawings. Your software will behave as you expect it to. There are fewer surprises.
- The overall system design will dictate the way the software is developed. The right decisions are made before you are married to poorly written code. Again, your overall costs will be less.
- UML lets us see the big picture. We can develop more memory and processor efficient systems.
- When we come back to make modifications to your system, it is much easier to work on a system that has UML documentation. Much less 'relearning' takes place. Your system maintenance costs will be lower.
- If you should find the need to work with another developer, the UML diagrams will allow them to get up to speed quickly in your custom system. Think of it as a schematic to a radio. How could a tech fix it without it?
- If we need to communicate with outside contractors or even your own programmers, it is much more efficient.

DRAWING CLASSES

The basic element in a class diagram is a class. In UML, each class is drawn as a rectangle. At the very least, the rectangle must include the class name. It can also be subdivided into two or three sections that can contain additional information about the class (i.e. variables, constructors, methods, etc.).

The following class diagram includes the class name, variables and methods:



In class diagrams, the fields are denoted as:

```
variableName: variableType
```

The constructors are denoted as:

```
ClassName(parameterName: parameterType)
```

The methods are denoted as:

```
methodName(parameterName: parameterType): returnType
```

The name of each variable or method can be preceded by a visibility indicator. The following are a list of the visibility indicators for class variables and methods:

INDICATOR	VISIBILITY	DESCRIPTION
+	Public	Makes classes, methods and data fields accessible from any class.
-	Private	Makes methods and data fields accessible only from within its own class.
#	Protected	Can be accessed by any class in the same package or its subclasses.