Credit Card Processing System Class Diagram

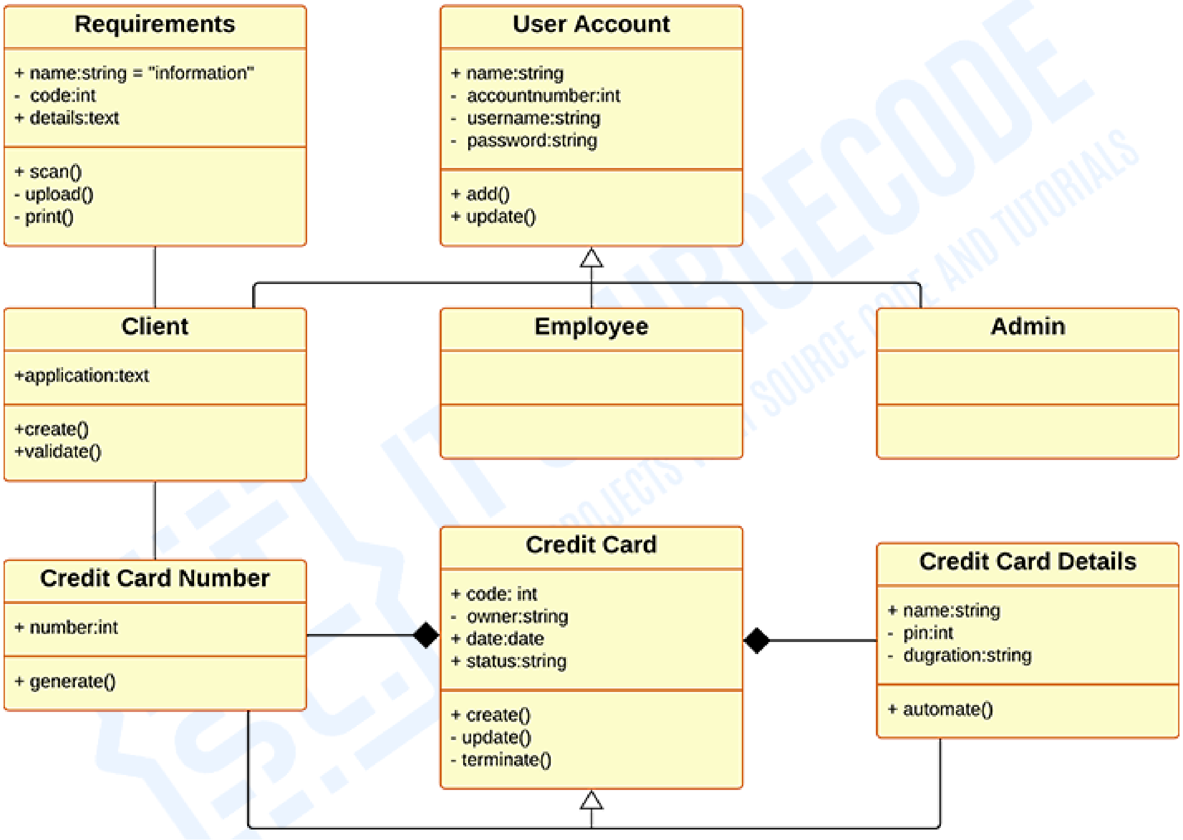
The **credit card processing system class diagram** is an illustration of the Unified Modeling Language's (UML) class relationships and source code dependencies. Its class describes the methods and variables of an object, which is a specific thing in a program or a piece of code that represents that object.

UML Class Diagram for Credit Card Processing System

The UML class diagram is similar to a flowchart in which classes are represented by boxes with three rows inside. The top rectangle holds the class's name; the middle rectangle contains the class's properties, and the bottom row contains the class's operation (methods). The class diagram is one of the most useful forms of UML diagrams because it elaborates the system's structure by representing its classes, characteristics, processes, and object relationships.

Credit Card Processing System Class Diagram

Here's the constructed Credit Card Processing System Class Diagram. It was provided with its attributes with matching methods to show the structure of each class. This is constructed with the simple idea derived from the common function of a Credit Card Processing System.



Class Diagram of Credit Card Processing System in UML

The illustration of a class diagram informs the project proponents on what are the characteristic (structure) of the software by showing the classes that will be included in the project. It resembles a chart in which classes are represented as boxes with three rectangles inside each box. The top rectangle has the class's name; the middle rectangle contains the class's properties; and the bottom rectangle contains the class's methods, commonly known as operations.

UML Class Diagram for Credit Card Processing System (Explanation)

As you can see through the illustration, the classes were determined which is symbolized by boxes. They were designated with their corresponding attributes and shows the class' methods. Their relationships are also plotted to show the connections between classes and their multiplicity.

You should also look into the visibility symbols displayed in the diagram. These are important because it declares the status of attributes in your Class Diagram. Some of the Class' attributes are for public (+) which means that they can be accessed by the classes connected to them. While the protected (#) symbols, means that the attributes of the data can be accessed by the same classes or subclass and the (-) symbol means it cannot be accessed by other class.

Just bear in mind that when you create your class diagram, you have to be specific. Because it will affect your project development. Do not worry because you can use the sample given as your project reference or you may also create your own.