Model 1 Credit Card

Classes often represent objects in the real world. In this section, you will design a new class that represents a `CreditCard` like the one below:

Questions (15 min) Start time:

1. Identify two or more attributes that would be necessary for the `CreditCard` class. For each attribute, indicate what data type would be most appropriate.

Answers may include `number:long`, `expire:Date`, `name:String`, `code:int`, etc.

2. Using UML syntax, define two or more constructors for the `CreditCard` class.

```
+CreditCard()
+CreditCard(number:long, name:String)
```

3. Define two or more accessor methods for the `CreditCard` class. Include arguments and return values, using the same format as a UML diagram.

```
+getNumber(): long
+getExpire(): Date
+getName(): String
+getCode(): int
```

4. Define two or more mutator methods for the `CreditCard` class. Include arguments and return values, using the same format as a UML diagram.

```
+setNumber(number:long): void
+setExpire(expire:Date): void
+setName(name:String): void
+setCode(code:int): void
```

5. Describe how you would implement the `equals` method of the `CreditCard` class.

Two credit cards would be considered equal if they have the same account number, assuming there are no duplicates in the bank.

6. Describe how you would implement the `toString` method of the `CreditCard` class.

The `toString` would print the account number, expiration date, and cardholder's name, each separated by a comma.

7. When constructing (or updating) a `CreditCard` object, which arguments would you need to validate? What are the valid ranges of values for each attribute?

The number should have 16 digits, dates need to have valid months and days, names should be at most 22 letters and not contain digits or other characters, code should be 3–4 digits, etc.