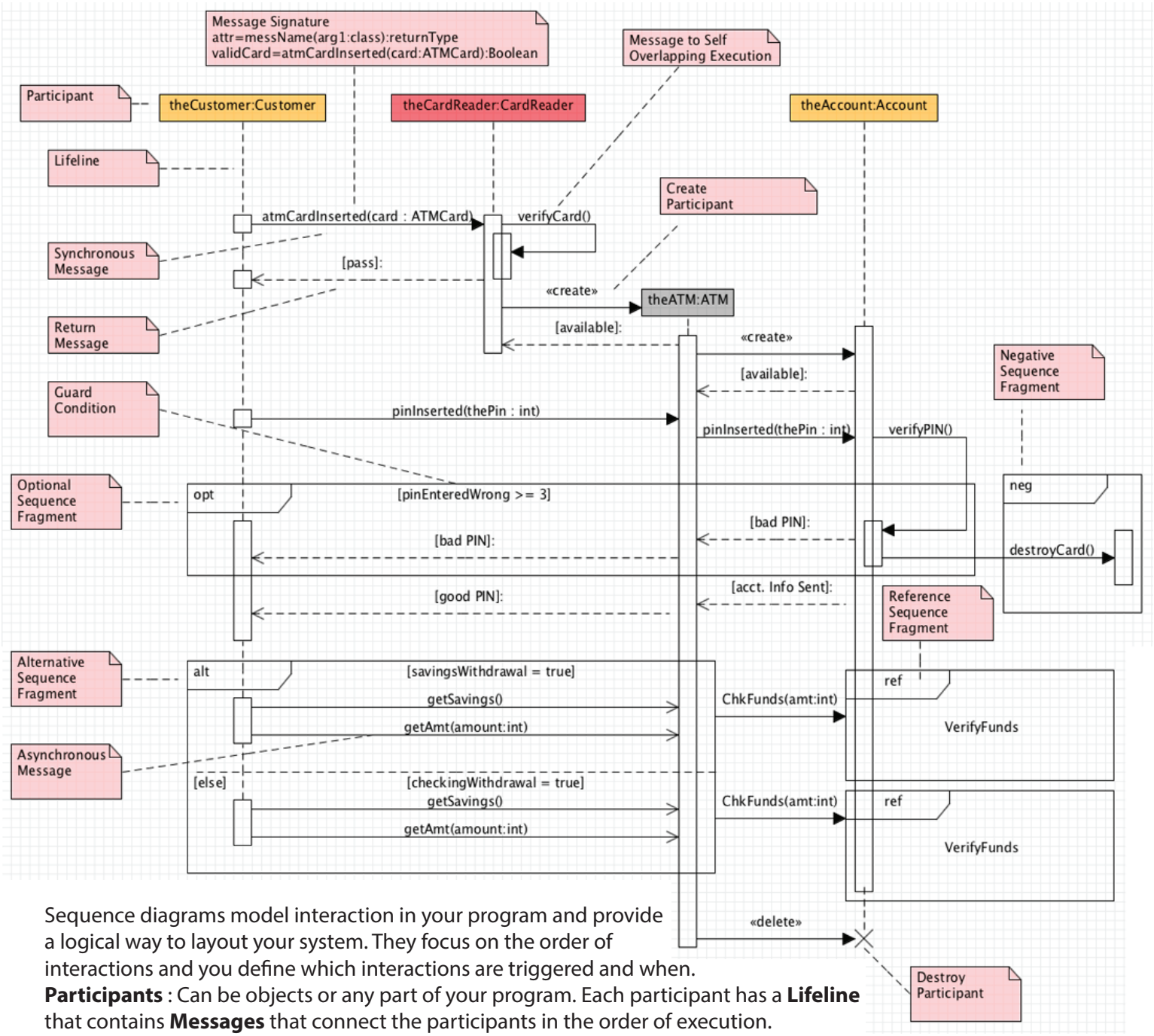

Sequence diagrams model interaction in your program and provide a logical way to layout your system. They focus on the order of interactions and you define which interactions are triggered and when. **Participants** : Can be objects or any part of your program. Each participant has a **Lifeline** that contains **Messages** that connect the participants in the order of execution.

* Here I demonstrate that the customer enters an ATM card and I send a Synchronus Message to the card reader to verify the card. **A Message Signature** defines the attribute, method name, parameters passed and the return type. A **Synchronous Message** is a message that sends and then waits for a response. **Asynchronous Messages** are messages that are sent and will not wait for a response like we do with getSavings(). You can have a participant create another with **Create**. And you can also **destroy** participants. **Sequence Fragments** are boxes that surround interactions. In the top left corner you define a fragment operator. **opt** means it is optional and it will only be used if the **guard** is true. So we are saying only if the **pinEnteredWrong** >= 3 will we destroy the card. At this time we don't want to destroy cards so we will create a **Negative Sequence Fragment**. A **Reference Sequence Fragment** is used when you have a complex interaction you want to place somewhere else in your diagram. In VerifyFunds you can see how to describe **looping**. Looping continues until the condition is false. You can also define to loop over a range. A **Parallel Sequence Fragment** is defined by using **par** and it designates that operations can occur in parallel. You can separate parallel operations with a dashed line if you want the operations to occur in order. An **Alternative Sequence Fragment** labeled with **alt** is used when you want to perform different actions depending on different conditions. Your also going to be able to define **Time Constraints** as we show we will allow 10ms after a call request and reply within 5ms. You can define what happens when a **Message is Lost** and when it is **Found**. An **Assert Sequence Fragment** specifies that interactions must work perfectly or an exception is triggered. We can **break** out of a loop if conditions are met. **Critical** can be used to make sure that 2 people can't access the account at once. **Nested Messages** occur when one message causes the receiving participant to send out 1 or more messages.