CHAPTER 12

# Sequence Diagrams

MAJOR TOPICS

## OBJECTIVES

At the completion of this chapter, you will be able to:

- Interpret sequence diagrams.
- Use sequence diagrams to illustrate the interactions between classes.

## PRE-TEST QUESTIONS

The answers to these questions are in Appendix A at the end of this manual.

1. What is a sequence diagram?

   .....

   .....

2. In a sequence diagram, what does a box depict? What does a dashed line depict? What does a arrow between boxes depict?

   .....

   .....

## INTRODUCTION

In an earlier chapter, you were introduced to collaboration diagrams and you used them to illustrate the flow of messages between objects. Sequence diagrams are a second type of interaction diagram. Sequence diagrams convey the same information as collaboration diagrams, but the sequence of events is more specific. You used collaboration diagrams within the analysis workflow to illustrate the relationships between instances of analysis classes. In this chapter, you will learn to use sequence diagrams to illustrate similar relationships between instances of design classes.

Figure 12-1 is a simple sequence diagram. Objects are depicted using boxes across the top of the diagram. A vertical dashed line called a lifeline extends down from each object. Messages are passed from one object to another, depicted by an arrow. Each message is labeled. At this stage in the development process, the labels should correspond directly with class methods defined in the class diagram. In this example, a CheckOutController object calls the PatronDB object's getPatron method. The patronID variable is passed as an argument.

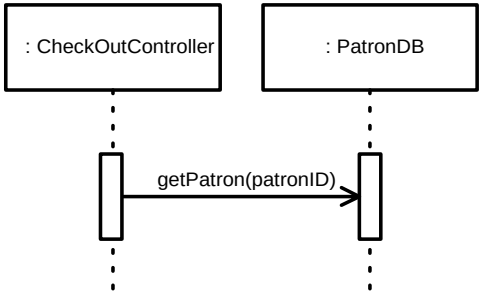


**Figure 12-1: Sequence diagram example**

## RETURN VALUES

The sequence of events proceeds down the diagram. Figure 12-2 includes a second message between the CheckOutController object and the Patron object. You can see that this message is delivered after the getPatron message because it appears lower in the diagram.

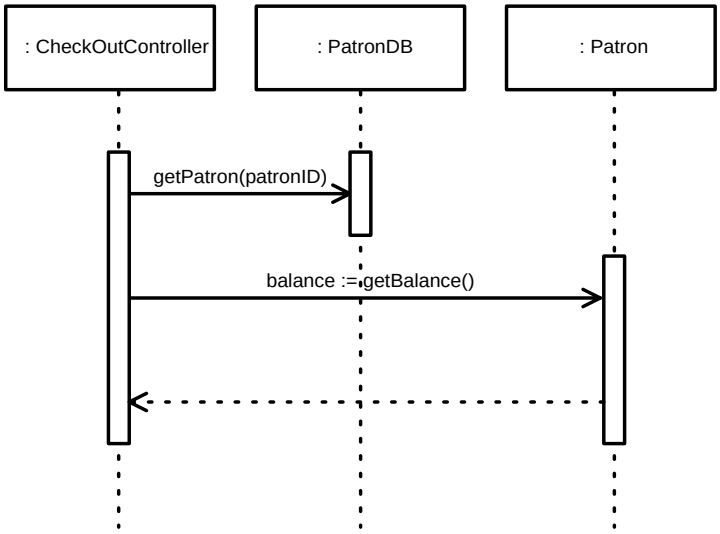


**Figure 12-2: Return values example**

As with collaboration diagrams, messages within a sequence diagram can have return values. The getBalance method returns a patron's account balance. This value is stored as a variable called balance. This variable can be used to set conditions on methods, or it can be passed as an argument to subsequent methods.

The dotted arrow between the Patron object and the CheckOutController illustrates a return from the getBalance method call. Returns could be included for all method calls (including those without return values), but they are generally added only where they contribute to the clarity of the diagram.

## MESSAGE CONDITIONS

Example 12-3 adds a third message, the payFine message. The payFine message is a call to the PayOverdueFineController object. It transfers control to the portion of the system that handles overdue fines. This message should only be passed if the patron owes an overdue fine. To illustrate this, a condition is placed on the message. The condition is indicated between brackets. In this example, the payFine message will only be passed if the patron's account balance is greater than zero (balance > 0).

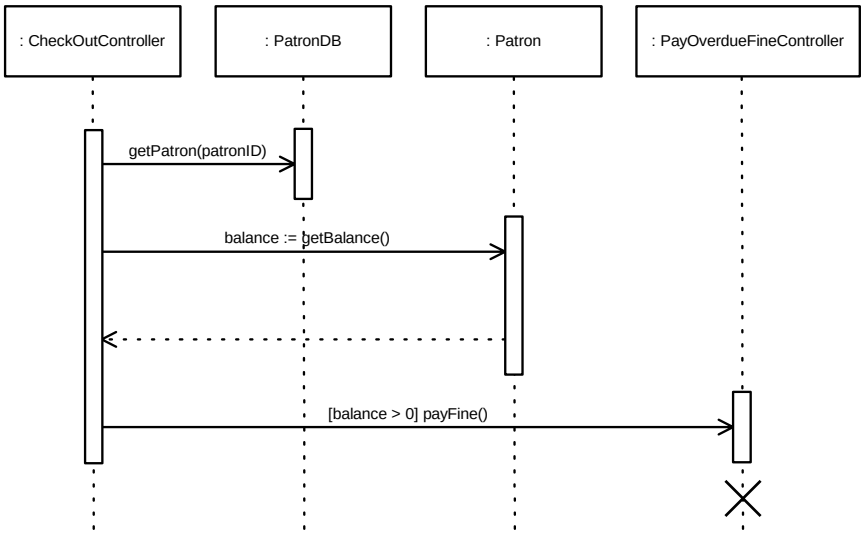


**Figure 12-3: Message condition example**

## DELETION

Notice the large X along the PayOverdueFineController lifeline in the preceding figure. This X is called a deletion. A deletion indicates the death of an object. In this example, the PayOverdueFineController is instantiated by the CheckOutController. Control is transferred to the PayOverdueFineController object through the message payFine. Once the overdue fine has been paid, the PayOverdueFineController deletes itself. The object no longer exists and no additional messages can be sent to it. Objects may also be deleted by other objects. A message arrow from one object's lifeline to a deletion on another object's lifeline indicates that the deletion is performed by the first object.

## MULTIPLICITY

Some messages may be sent multiple times. A patron may check out multiple assets simultaneously. Figure 12-4 is the complete sequence diagram for the Check Out Asset use case. The CheckOutController must add multiple assets to its AssetList object. The addAsset method is called multiple times using a flow-control structure such as a while loop. The multiplicity of this message is indicated with an iteration marker. The asterisk next to the addAsset message indicates that this message may be sent multiple times.

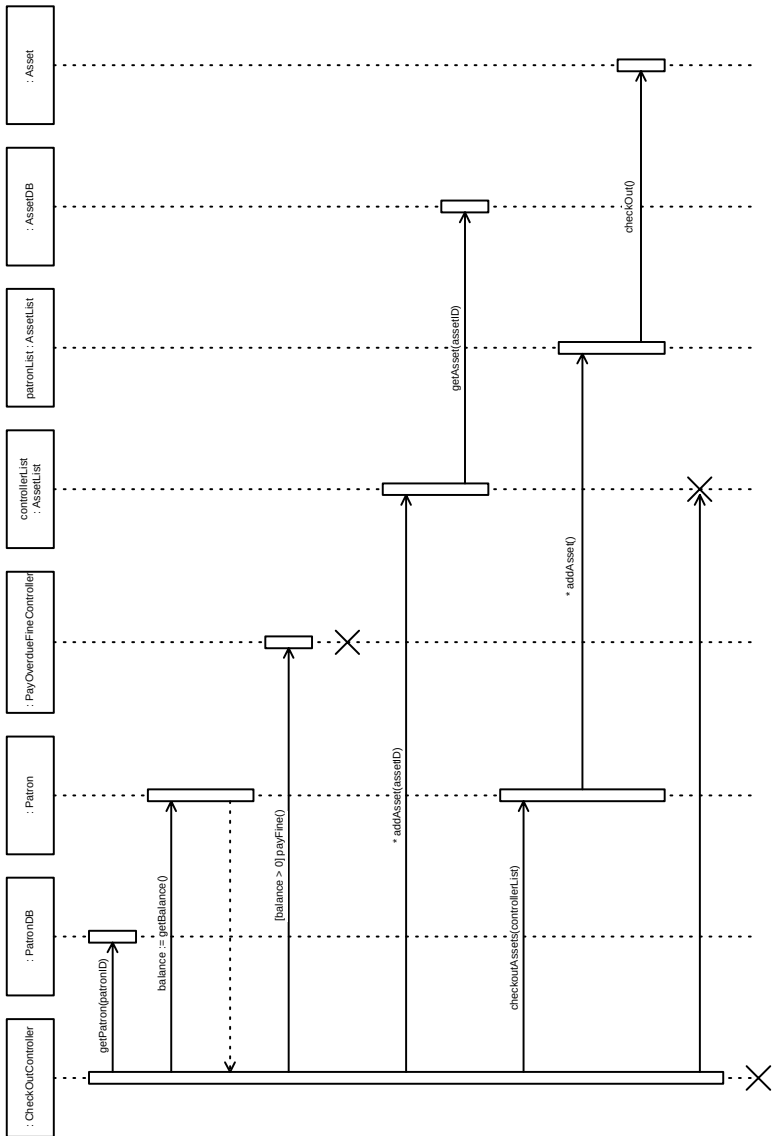


**Figure 12-4: Check Out Asset sequence diagram**

## RETURN STACK

As one class's method makes a call to another class's method, a reference is placed on the return stack so the CPU knows where to return to when the called method is complete. If the called method calls a third method, a second reference is placed on top of the first. As execution of each method completes, the references are removed from the stack in reverse order and control is passed back to calling methods in reverse order. This process is illustrated using the boxes (called activation boxes) along the lifelines of objects in a sequence diagram. An activation box indicates a reference to one of the object's methods on the return stack.

In the preceding figure, the CheckOutController makes a call to the Patron object's checkoutAssets method. Notice that the activation box along the Patron object's lifeline extends down the diagram past the checkOut message. This notation indicates that this calling method remains active and on the stack until the methods it calls have returned.

### Exercise 12-1: Creating sequence diagrams

1. Consider the class diagrams you created in the previous chapter's exercise. Identify the sequence of message passing between objects that will realize each class diagram's functionality.
2. Create sequence diagrams for each class diagram you created in the previous chapter's exercise.

## SUMMARY

Sequence diagrams are a form of interaction diagram. Like collaboration diagrams, sequence diagrams illustrate the flow of messages between objects. Sequence diagrams are used within the design workflow to illustrate the flow of messages between design classes.

## POST-TEST QUESTIONS

The answers to these questions are in Appendix A at the end of this manual.

1. What does a X over a lifeline indicate?

   .....

   .....

2. How does one go about illustrating the return stack's processes in a sequence diagram.

   .....

   .....