Chapter 11 – Object-Oriented Design: Use Case Realizations

ASSIGNMENT

Running Cases: On the Spot Courier Services

In chapter 11 you learned about creating first-cut sequence diagrams and multi-layer sequence diagrams. Below are some of the sequence diagrams for the On the Spot Courier Services case study. Review them carefully and use them to help fill in the details of the final class diagram in the #1 assignment below.

First-cut sequence diagram (domain classes and controller classes only).

![Sequence Diagram](image-url)
Pickup a package

- [no label] requestInfo := findPickupRequests (customerName)
  - [no label] aR := findPickupRequest {customerName}
  - aPR := showRequest
  - [no label] aP := processPackagePickup ()
    - aP := createNewPackage {aPR}
    - aE := getEmployee ()
    - recordEvent [aP, aE]
    - updateAccount {amt}
  - [no label] aP := processNewPackageInfo (name, address, ...)
A multilayer sequence diagram includes the domain classes and the view layer classes. (This drawing did not combine view and data access layers on the same drawing. It makes the drawing too complex.)

In Chapter 10, you developed a first-cut design class diagram and CRC card solutions for two use cases: Request a package pickup and Pickup a package. Now you will extend your solution from that chapter by developing the following:

1. A final design class diagram that includes the classes from both use cases. Include elaborated attributes, navigation arrows, and all the method signatures from both use cases. Refer back to the chapter 10 solution document in the Resources folder. Essentially you are going to combine the two class diagrams in to one diagram with the elaborated attributes, navigation arrows, and all of the methods. Refer to figure 11-18 for an example and the level of detail required for final design class diagram.

2. In Chapter 8, we defined four subsystems:
   - Customer account
   - Pickup request
   - Package delivery
   - Routing and scheduling

Even though these subsystems are somewhat arbitrary, we can treat each one as a separate package. Develop a package diagram for each of the four subsystems by assigning domain model classes to each package. A domain model class should belong to only one subsystem package. Normally, it is the subsystem that instantiates objects from that class. Also, show dependency relationships among the various packages and classes.