LINGI2251
Software Engineering: Development Methods
Assignment 2
Spring 2016

Charles Pecheur

April 12, 2016

Due Apr 25, 2016

This assignment can be performed in **groups of two students**. Individual submissions are also accepted. Interaction between students and groups is allowed but plagiarism will not be tolerated.

Please post any question regarding this assignment on the corresponding forum on the course website. For personal issues you may contact the assistant, *Hossein Haeri* (`hossein.haeri@uclouvain.be`).

Subject: the Dinoco GSCS

Following the work done in Assignment 1, You are now in charge of the design of the Gas Station Control System for Dinoco. From the requirements analysis, the following data flow diagram (Fig. 1) and class diagram (Fig. 2) have been produced for the system. The description of the Dinoco GSCS system is available as a separate document.

1 Architectural Design

The data flow diagram shows three main processes, respectively in charge of processing a refueling at the pump, processing payment for refueling at the cashier's, and processing bill payments mailed by account holders. The following additional considerations have been identified:

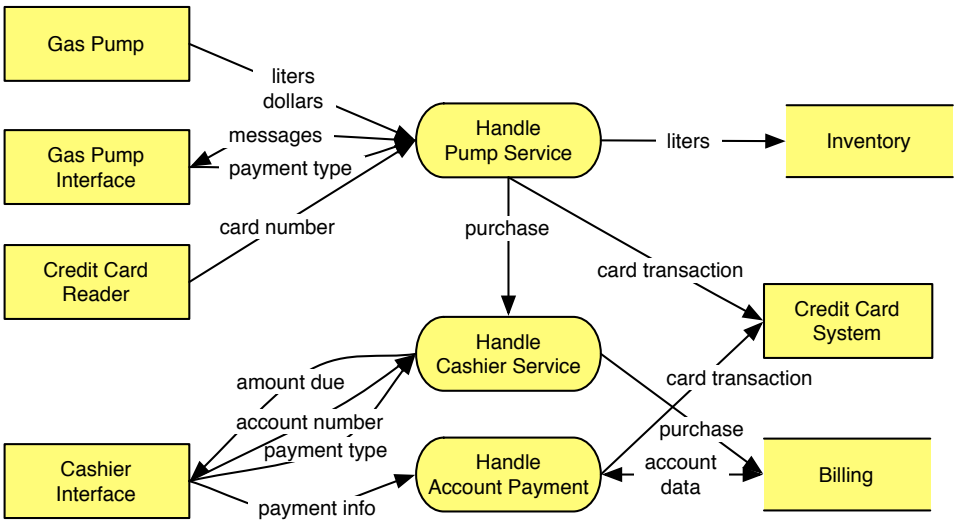

Figure 1: Data flow diagram for GSCS

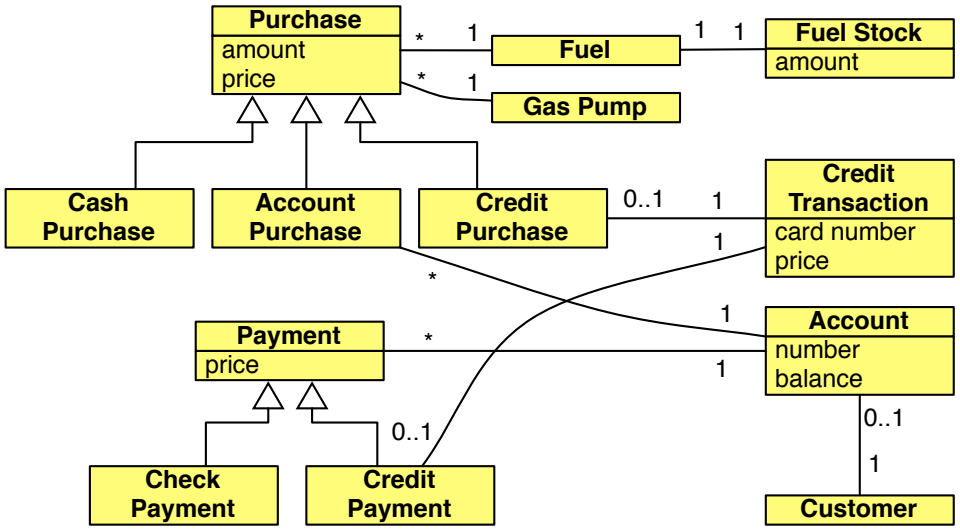

Figure 2: Class diagram for GSCS

- Communication with the pump, the pump interface and the credit card reader will use specific communication protocols implemented in available software drivers and libraries.
- The cashier's interface will be a graphical user interface running on the same computer as the GSCS. Communication will be by messages through a socket interface.
- Fuel stock inventory and accounting will be handled by separate databases.
- Communication with the credit card system will use a secure data communication over a phone line.
- Multiple pumps must be served concurrently, while there is only a single cashier's interface.

Propose an architectural design for the GSCS system. Interactions with external elements (devices and repositories) should be encapsulated so as to minimize the impact of a change of these elements on the overall system. Your design will include:

1. A **hierarchical decomposition** of the system into components.
2. A description of the **roles and interactions** of all components.

2 Detailed Design

The software will be built using object-oriented design. Your design will include:

1. A detailed **class diagram** for the software. Show attributes and particular operations (not generic getters/setters). Your diagram may refer to some or all of the classes of Fig. 2, refined as needed.
2. A **short description** of the role of each class.
3. A **USES diagram** of the classes. Report any loop and discuss whether and how it could be eliminated. Identify possible sets of classes for an incremental development (e.g. by framing groups within the graph).

Design Patterns

You will apply design patterns to refine and enrich specific elements of your object-oriented design. For each application, give a class diagram of the relevant classes and write skeletons of the relevant methods. (Use Java or a similar OO programming language). Apply the following patterns:

1. The class `Payment` represents a mailed payment received for an account bill. The subclasses correspond to a payment with a check or charged on a credit card. Using the **Template Method** pattern, add a method `getReceipt()` to `Payment` that produces a printable receipt for the payment. The receipt will contain a description of the payment that will depend on the payment type.

2. Different reports will be generated from the account database. Using the **Strategy** pattern, add a `Reporting` interface that provides two methods for generating printouts: one for instances of `Account` and one for instances of `Payment`. Illustrate how this interface could be used.

3. Using the **Decorator** pattern, add decorator capabilities to the `Purchase` class. Show how this could be used to add processing fees to the price of credit purchases.

4. Once the customer hangs up the nozzle, the gas pump reports the amount and price of gas dispensed. Use the **Observer** pattern to allow both the purchase processing component and the inventory update to be notified of this event.

5. Companies can open collective accounts at the gas station for their employees. These accounts are grouped together and billed to the company, but the accounts are still recorded individually to track each employee's purchases. Apply the **Composite** pattern to enrich the `Account` class to allow a hierarchy of accounts. Discuss at which level your existing methods for this class will be implemented.

2.1 Deliverables

Assignment results will be returned as an archive file (`.zip` or `.gz`) whose base name is the last names of the students (e.g. `Pecheur_Cailliau.zip`, no accents please). Submit your deliverables using the *Assignments* tool of iCampus, before the deadline stated on the first page. Late submissions will

not be accepted. Make sure to *include proper identification* (course, year, assignment number, student names) at the beginning of all documents.

The archive will at least contain a report covering all task items above. If more files are provided, their use will be explained at the beginning of the report.

You can use any drawing tool to construct diagrams asked in this assignment. We recommend *yEd* from yWorks, a general-purpose graphing tool with excellent graph layout capabilities. yEd is freely available and runs on all platforms, see `http://www.yworks.com`.