Problem 14.16

(Term Project) Starting with your specifications of Problem 12.20 or 13.22, design the Chocoholics Anonymous product (Appendix A). Use the design technique specified by your instructor.

Step-by-step solution

1. **Step 1** of 9

The final class diagram is shown in Figure 13.109 of this Instructor’s Solution Manual.

**Design Decisions**

1. A command line interface will be used to simulate the provider terminal as well as the operator and manager interface.

[Comment](javascript:;)

1. **Step 2** of 9

2. Instead of sending reports as email attachments, or printing reports, they will be saved as text files.

[Comment](javascript:;)

1. **Step 3** of 9

3. EFT data that are generated will also be saved as a text file.

[Comment](javascript:;)

1. **Step 4** of 9

4. The system will be implemented as one stand-alone system on a PC.  The simulations of the provider terminal, operator interface and manager interface will be able to be launched from this system, or separately.

[Comment](javascript:;)

1. **Step 5** of 9

5. The operator’s terminal simulation will allow the update of a member’s status, at any time, because there is no third party accounting system available yet.

[Comment](javascript:;)

1. **Step 6** of 9

6. The accounting procedure will be run whenever the tester chooses instead of at midnight on Fridays.

[Comment](javascript:;)

1. **Step 7** of 9

7. Persistent classes will be saved in text files between runs of the program.  During the run of the program, persistent classes will be stored in array lists.

[Comment](javascript:;)

1. **Step 8** of 9

8. Relationships between entity classes will be implemented by common values (identifiers) in attributes of the related objects, and not as references to the related objects.

[Comment](javascript:;)

1. **Step 9** of 9

9. The product will be implemented using Java and C++.

**Mapping of analysis classes to design classes**

 For most of the analysis classes there is a one-to-one mapping to design classes.

 The **ChocAnSystem** class is introduced to launch the simulations of the provider termi­nals, operator interfaces and manager interfaces.

 The **UserInterface** class is introduced to encapsulate the command line interface.  The four analysis class user interfaces (**ProviderInterface**, **OperatorInterface**, **ManagerInterface**, and **SchedulerInterface**) use the **UserInterface** design class for all input.

 Each entity class is mapped to two design classes, for example, the **Member** analysis class will be mapped to a **Member** design class and a **Members** design class.  The **Members** class handles the persistency.  There are five such classes: **Persons**, **Members**, **Providers**, **Services**and **Claims**.  These classes could be replaced with an interface to a database.

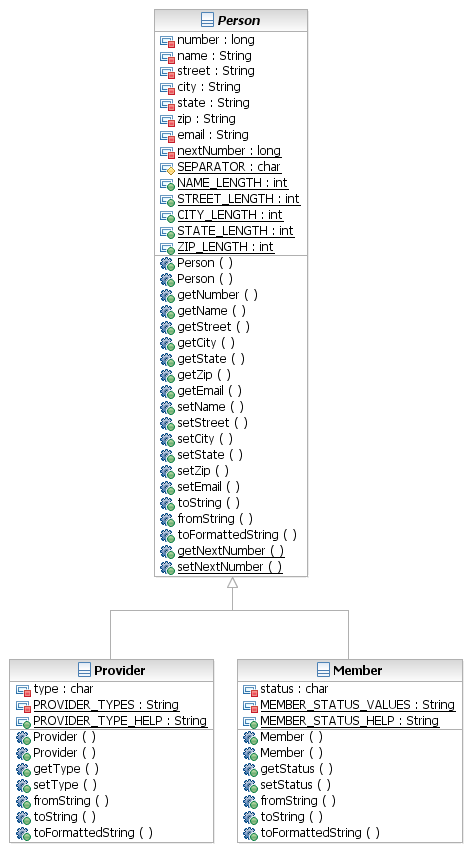
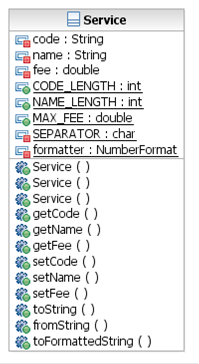
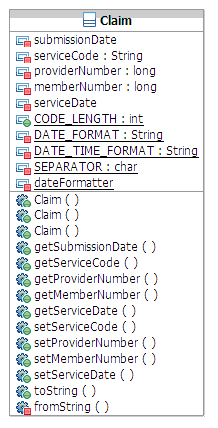
 The common behavior of **ProviderMaintainer** and **MemberMaintainer** is generalized into a new superclass **PersonMaintainer**.

 The common attributes and behavior of all the report classes are generalized into a new superclass **Report**, and the common behavior and attributes of those reports that are lim­ited by a date range are generalized into a new superclass **DateRangeReport**, a subclass of **Report**.

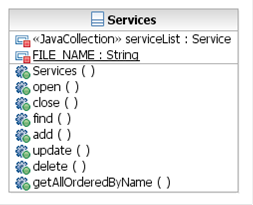
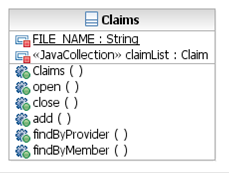
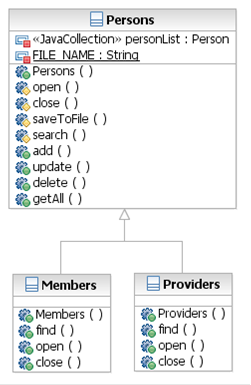
 Because text files are being used, the responsibility for “displaying,” “printing,” “emailing, and “saving” a report has been moved from the report-generating control classes (e.g., **MemberReportGenerator**) to the **Report** class hierarchy.

Design classes

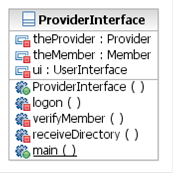
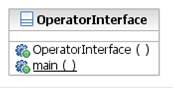
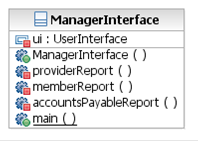
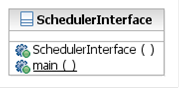
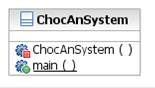
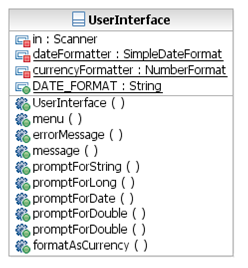
Entity classes



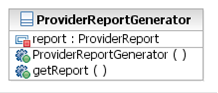
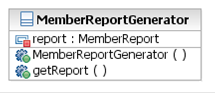
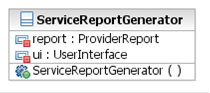
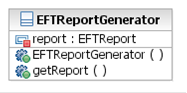
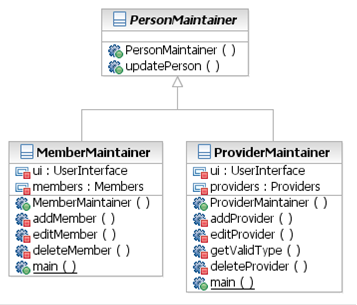
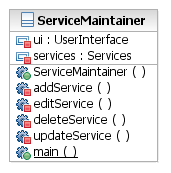
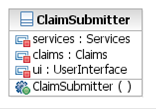
**Persistency classes**



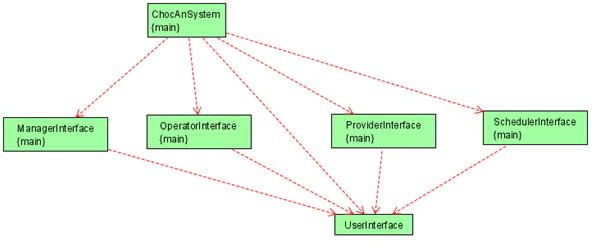
**System, subsystem, and boundary classes**



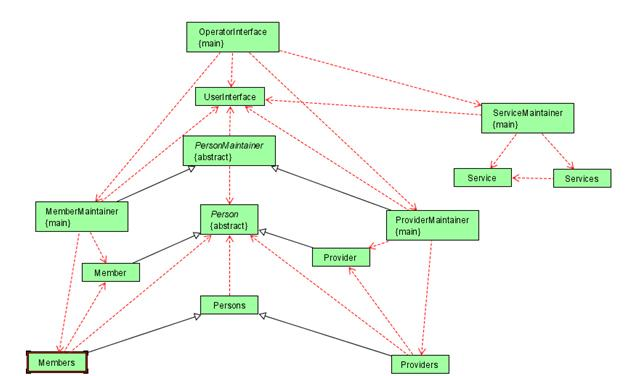
Control classes



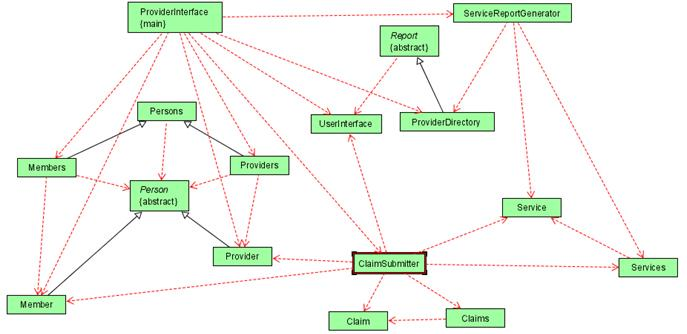
**Design dependencies ChocAnSystem dependencies**



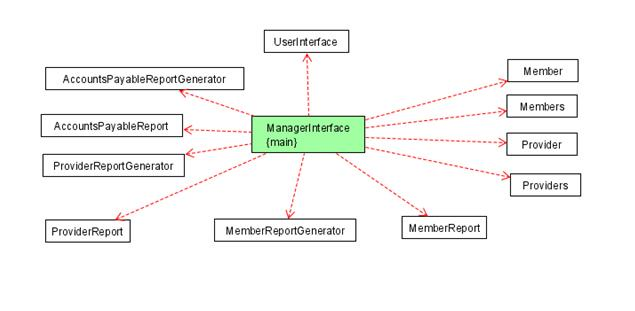
**Maintenance subsystem dependencies**



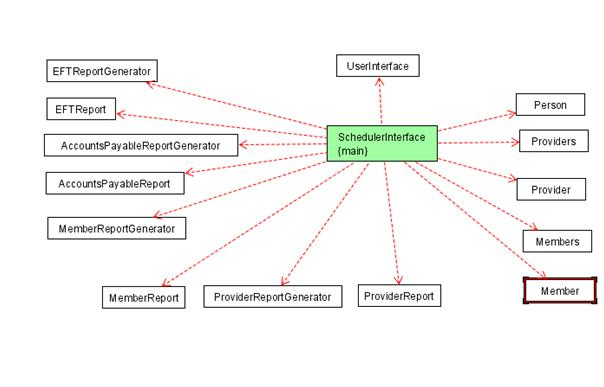
**Provider subsystem dependencies**



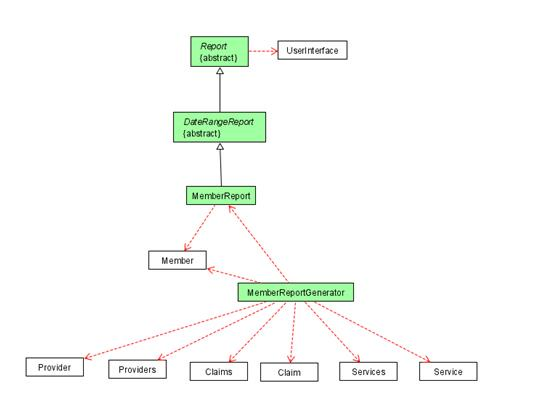
**Reporting subsystem dependencies (direct dependencies)**



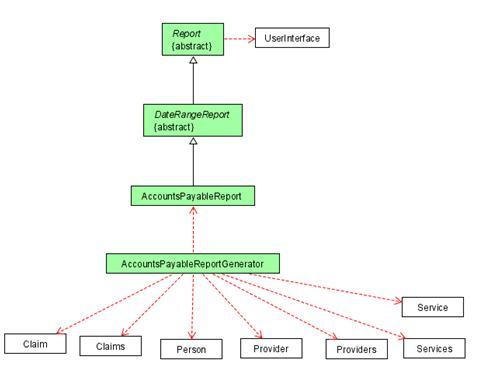
**Scheduler interface (direct dependencies)**



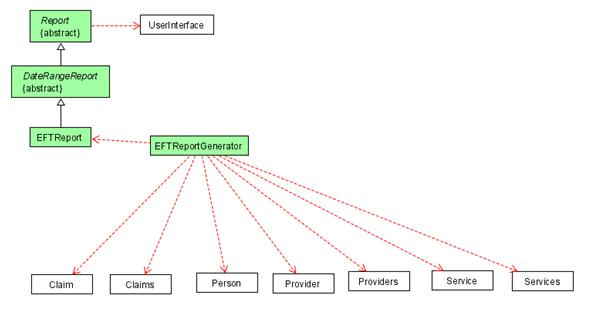
**Member report generator dependencies**



**Accounts payable report generator dependencies**



**EFT report generator dependencies**



**Report generator dependencies combined**

