CS 278

Code Review

Name:

Review of the solution to Asgn \_\_\_\_ implemented by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Git Repo:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Path:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Provide a 1-3 paragraph overview of the architecture of the code and the design rationale.

Example: *This code uses the reactor-pattern to dispatch service events to a series of service handlers. The service handler interface is specified as a Java interface allowing users of the application to extend the available services by creating a Java object that implements this interface. An event pre-processor is used to extract the “service type” header from events and then lookup the appropriate service handler using the type name. Service handlers are stored in a central HandlerRegistry and bound to service types, which are strings.*

*The code’s unit testing methodology is to create one unit test per concrete class. No unit tests are provided for classes that are primarily simple data structures with getters/setters. The unit tests are designed to call each of the methods on the test subject. No apparent boundary value analysis or other approach was used to devise the unit tests.*

1. Refactoring 1:
   1. Provide the assumption that the code is making that may not always be correct and that your refactoring will correct.

Example: *The application assumes that there is only a single event dispatcher implementation needed.*

* 1. Provide a 1-paragraph overview of why you think the code structure, functionality, etc. that you are suggesting to refactor is not ideal in all cases.

Example: *The application currently has a single event dispatcher implementation based on the reactor pattern. There are multiple ways to implement the event dispatcher, such as a leader/followers implementation. However, there is no interface for an “event dispatcher” that can be used to decouple the code from the specific event dispatcher implementation that is currently used. This lack of an event dispatcher interface limits the extensibility and reusability of the code base.*

* 1. Provide a 1-paragraph overview of how your proposed refactoring will provide better functional/non-functional characteristics.

Example: *I propose to refactor the code base to add an event dispatcher interface with methods for: 1) setting the HandlerRegistry to lookup service handler implementations, 2) starting/stopping the dispatcher, and 3) adding pre/post event dispatching filters. This proposed refactoring will improve extensibility by allowing other event dispatcher implementations to easily be swapped in and also allow event dispatch filters to be applied without changing the application’s core logic. For example, an event dispatch filter to log all events before dispatching could be added.*

* 1. Provide a step-by-step description of the suggested refactoring.

Example:

1. *Create an EventDispatcher interface in the org.foo.bar package*
2. *Add methods to the EventDispatcher interface as follows:*
   1. *setServiceRegistry(HandlerRegistry reg)*
   2. *addDispatchFilter(When whentoapply, Filter<Event> f)*
   3. *removeDispatchFilter(Filter<Event> f)*
   4. *start()*
   5. *stop()*
3. *Refactor the org.foo.bar.ReactorDispatcher to implement the EventDispatcher interface*
4. Refactoring 2:
   1. Provide the assumption that the code is making that may not always be correct and that your refactoring will correct.
   2. Provide a 1-paragraph overview of why you think the code structure, functionality, etc. that you are suggesting to refactor is not ideal in all cases.
   3. Provide a 1-paragraph overview of how your proposed refactoring will provide better functional/non-functional characteristics.
   4. Provide a step-by-step description of the suggested refactoring.
5. Refactoring 3:
   1. Provide the assumption that the code is making that may not always be correct and that your refactoring will correct.
   2. Provide a 1-paragraph overview of why you think the code structure, functionality, etc. that you are suggesting to refactor is not ideal in all cases.
   3. Provide a 1-paragraph overview of how your proposed refactoring will provide better functional/non-functional characteristics.
   4. Provide a step-by-step description of the suggested refactoring.