*Title: Refactoring:*

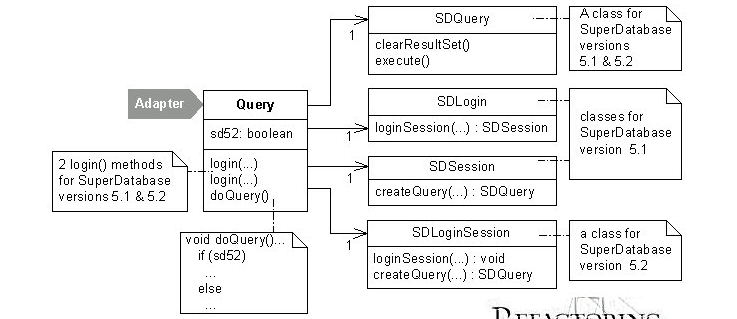
*One class adapts multiple versions of a component, library, API or other entity.*

Extract an Adapter for a single version of the component, library, API or other entity.

**Motivation to refactor:**

While software must often support multiple versions of a component, library, or API, code that handles these versions doesn't have to be a confusing mess. Yet I routinely encounter code that attempts to handle multiple versions of something by overloading classes with version-specific state variables, constructors, and methods. Accompanying such code are comments like "This is for version X—please delete this code when we move to version Y!" Sure, like that's ever going to happen. Most programmers won't delete the version X code for fear that something they don't know about still relies on it. So the comments don't get deleted, and many versions supported by the code remain in the code.

Sample:



Sample of code:

public class FilterCoffeeApp {

private Logger log = Logger.getLogger(

FilterCoffeeApp.class.getSimpleName());

private FilterCoffeeMachine coffeeMachine;

public FilterCoffeeApp(FilterCoffeeMachine coffeeMachine) {

this.coffeeMachine = coffeeMachine;

}

public Coffee prepareCoffee() {

Coffee coffee = this.coffeeMachine.brewCoffee();

log.info("Coffee is ready!");

log.info(" -> " + coffee);

return coffee;

}

}