Description of Project

Goal

This project required the ability to have prices for cans of Power Juice, and discounts for specific compositions of food products, e.g., popcorn, peanuts, pizza. This implementation deals with Power Juice only, but is extensible to include other types of food product. Discounts are applied based on the composite of all food products purchased.

Components

The classes contained in our project are as follows (please reference the UML class diagrams included in this submission for the methods and variables contained in each class):

- **FoodProduct** interface - represents a general food product
- **PowerJuiceCan** class - implements FoodProduct, represents a single can of Power Juice
- **FoodProductComposite** class that implements FoodProduct; composed of multiple FoodProducts

Design Decisions

The Composite design pattern was used to create sets of food products, and apply the correct discount based on the total number of items.

The FoodProduct interface was created so that other types of food can be added to the entire purchase. A PowerJuiceCan implements the getPrice() and getCount() methods from FoodProduct.

The FoodProductComposite keeps track of all the items in a purchase. It calculates the total price by counting its constituents and adding their prices. It then applies the discount for a 24-pack for every 24 items, and the discount for a six-pack for every remaining six items.

Testing

JUnit tests were written and performed to check the functioning of the model. All tests were successful, so we concluded that the model does a good job of expressing the Observer pattern.