**Fruit Basket Component**

**== Overview ==**

System under development is a thread safe basket component that receives basket items from a parent system, such as an e-commerce. Basket items can be fruits i.e. Apples, Bananas and so on. After every basket item is received the basket recalculates. For example, it recalculates the total cost of the basket.

**== Assumption ==**

It is assumed that the parent system e-commerce that uses the basket

- is designed as a pub/sub system, therefore the basket implements the Consumer class.

- will maintain a separate basket for each shopper, therefore the basket uses a key and getInstance method to get or create a basket for each shopper.

- will call the accept method of the basket to add items to the basket.

- will run in a multithreaded environment, therefore the basket is designed to be thread safe.

**== Design ==**

A pub/sub approach was used to design the basket component. The publishing is provided by the parent e-commerce system. This basket component provides the subscription part of the pub/sub. The basket therefore implements the Consumer class and expects new items to be added by calling the accept method.

Questions to be answered by the system are based on all the items in the basket. As such the system segregates and accumulates the statistical data per basket. Any locks and concurrency are handled on the basket level to ensure that data is not corrupted at that level.

Program structure consists of Basket class, BasketItem class, api and model packages.

- Basket accepts each item that is added to the basket and recalculates the total cost.

- BasketItem represents each item that is added to the basket. It contains fields price and item. Item is a parameterized variable that is marked as a Fruit.

- Fruit class represents the fruit name and id. It does not have any information about pricing. That is left to the basket item and basket classes.

- api package contains the interfaces and abstract classes.

- model package contains the domain classes, i.e. the different kinds of Fruits

A test driven approach was used. Unit tests were created for the basket functions followed by implementation the logic. At each build cycle the unit tests are evaluated automatically. If any of the tests fail, the build fails.

Java 8 and maven 3 were used to develop, manage dependencies and build the program.

**== Future Development ==**

In order to extend the program, additional basket statistics will be created.

Further extension can be achieved by separating the api package into a standalone project that produces distributable jar file(s)