Data Integrity Report - Group 5

— Preface —

Each of the models (Product, Item, StorageUnit, etc…) have their own singleton `Vault` class which makes sure they are valid with respect to other models of their kind, and manages relationships between models.

For example, a new StorageUnit will check to see if it's unique by calling the `validateNew` or `validateModify` methods on its pointer to the StorageUnitVault instance. The vault will check to see if the name is unique, along with other constraints, and will return a Result object indicating the success of the verification. If a model does not meet the data constraints, it cannot be successfully validated.

Each instance of a model contains a `_valid` member variable which is by defualt false, and is set to true only when the model has been successfully validated. Models can only be saved if they are first validated.

In our data organization scheme, each model instance is assigned an ID and added to a map in its associated vault. Rather than a parent node holding pointers to its children as is suggested in the functional spec, each model holds the ids of its parents. If I were desire access to a product beneath a specific StorageUnit, I'd call `find("RootParentId = foo")` which would return to me a copy of the found product.

— Product Container —

The data constraints for ProductContainer are enforced by the children of this superclass.

— Storage Unit —

CONSTRAINT	IMPLEMENTED	TESTED
Name must be non empty	StorageUnit.validate()	StorageUnitTest.testValidate()
Name must be unique	StorageUnit.validate()	StorageUnitTest.testValidate()

— Product —

CONSTRAINT	IMPLEMENTED	TESTED
creationDate must equal the earliest entry date for any item of the product. If the product holds no items, creationDate is set to today.		