PriceComparable

Item

Entity objects are data store objects and have no advanced functions to them

Fields that have no setter are final

Item()

getPrice()

getId()

getStoreId()

getName()

setPrice()

setName()

Store

Ratable

Store

Entity objects are data store objects and have no advanced functions to them

Fields that have no setter are final

getId()

getName()

getLocation()

getMenu()

addItem()

getAvgRating()

getRatings()

addRating()

removeRating()

Item

RatingPoint

User

Entity objects are data store objects and have no advanced functions to them

Fields that have no setter is final

Since user can create Item and Store, we may add reference to Store and Item objects in User in the future. The specific design choices are currently under discussion.

getName()

getId()

getEmail()

getPassword()

setEmail()

setPassword()

setName()

getRatingList()

addRating()

removeRating()

RatingPoint

Ratable

RatableItem

Item

RatableItem inherits all fields and methods from Item class, which are not shown in the child class. However, Interface methods will be shown in its implemented classes. This format will be consistent throughout the CRC model.

getAvgRating()

getRatings()

addRating()

removeRating()

RatingPoint

<Interface> Ratable

getAvgRating() returns a float representing the average rating across all RatingPoint objects in the list stored in ratable.

getRatings() returns a List<RatingPoint> type, enforcing the field to exist in implementing classes

addRating() and removeRating mutates the above mentioned list.

getAvgRating()

getRatings()

addRating()

removeRating()

<Interface> PriceComparable

Enforces that a float that represent price exist in implementing classes

getPrice()

RatingPoint

getRating() within rating point returns an int, either 0 or 1, to represent thumb down or thump up.

A RatingPoint object will have its associated user and ratable object stored in them.

getRating()

getUser()

getId()

getRatable()

User

Ratable

ItemFacade

This is a usecase class that stores all usecase subclasses related to Item entity.

Dependencies for database and presenter are injected through IDb, IPresenter interface with constructor.

FindItem usecase can perform search by different matching conditions e.g. id, storeId, name, or simply return all Items in the system.

UpdateItem usecase is responsible for mutating non-final fields in Item object and reflect changes to the database.

CreateItem

FindItem

RemoveItem

UpdateItem

UpdateRatableIte
m

Item

RatingPoint

Store

Ratable

IDb

IPresenter

InputPort

StoreFacade

FindStore usecase can perform search by different matching conditions, similar to FindItem.

UpdateStore usecase is responsible for mutating Store objects and reflect those changes to the database. Update operations to all mutable fields within Store object is delegated to this class.

CreateStore

FindStore

RemoveStore

UpdateStore

Store

Item

RatingPoint

Ratable

IDb

IPresenter

InputPort

UserFacade

LoginUser usecase matches the password stored in User object with input from the actual user. If the password matches the user is granted a Logged in state.

CreateUser

FindUser

LoginUser

RemoveUser

UpdateUser

User

RatingPoint

Stores

Item

IDb

IPresenter

InputPort

Soter

Usecase interactor that sorts PriceCOmparable objects and Ratable objects.

sortByPrice()

sortByRating()

Ratable

PriceComparable

EntityHashmap

A in-memory data store within the usecase layer.

Usecases for different entity can fetch other collaborating entity types through this memory object.

Changes made in this layer are reflected in the persistent database in real time.

add()

remove()

findById()

findAll()

Item

PriceComparable

Ratable

RatableItem

RatingPoint

Store

User

IDb

<Interface> IPresenter

Presenter interface for usecase to carry respond to the outer view element.

The coupling ResponseModel interface that is injected into presenter implementations is not shown in this CRC model.

show()

<Interface> InputPort

InputPort for controller to know what operations are supported by the usecase.

The coupling RequestModel object for this interface is not shown.

Each Usecase subclass currently have its own InputPort interface e.g. ICreateItem. This design is tentative and is subjected to change.

All usecase methods are here

<Interface> IDb

Database interface for usecase to communicate with database implementation.

Each usecase have its separate IDb interface e.g. IItemDb, to accommodate for each usecase interactors particular need. Note this is a tentative design and is subjected to change.

add()

remove()

findById()

findByName()

findAll()

GenericController

The controller that houses the main logic for choosing which usecase interaction to perform.

The run method will issue prompts and get input from the view object’s methods and choose the appropriate usecase to initiate.

Usecases are initiated as their corresponding InputPort interface type.

Dependencies required for usecase classes and the controller itself is passed into during object construction, which happens in the main method.

run()

IDb

IPresenter

IView

InputPort

<UsecaseInteractor>Controlle r

The controllers for handing specific usecase.

Once the GenericController have determined what usecase to invoke, the specific usecase controller (this object) is initiated and dependencies required are passed down from the GenericController.

The usecase specific controller will then prompt the use for necessary inputs to perform the usecase operation and initiate said operation through the input port.

Note that the output of usecase methods are handled by presenter, which also interacts with view object.

<nameOfUsecaseMethod>()

IDb

IPresenter

IView

InputPort

IPresenter

GenericPresenter

show()

IView

Presenter implementation that is injected into usecase to carry outputs to the view object.

The show method takes in a RespondModel object geneated by usecase that carries the output data and correctly format them and present it to the view object by generating a ViewModel object (not implemented in the skeleton project)

<Interface> IView

View Interface that outlines the required functions for a view(ui) implementation.

View implementation is injected into Controller which uses it to issue prompts to UI and get correct inputs from user.

This structure represents a Model-View-Controller design pattern.

getOpreation()

getAnotherQuery()

getName()

getPrice()

getId()

IView

View

Implementation of a IView object.

This exists in the skeleton project as a commandline ui.

getOpreation()

getAnotherQuery()

getName()

getPrice()

getId()

Database

IDb

Implementation of a IDb object.

This exists in the skeleton project as a non-persistent hashmap.

add()

remove()

findById()

findByName()

findAll()

Main

Main method of the program

All required dependencies are initiated here and passed into the GenericController.

```
psvm(){

GenericController.run()}
```

View

Database

GenericController

GenericPresenter