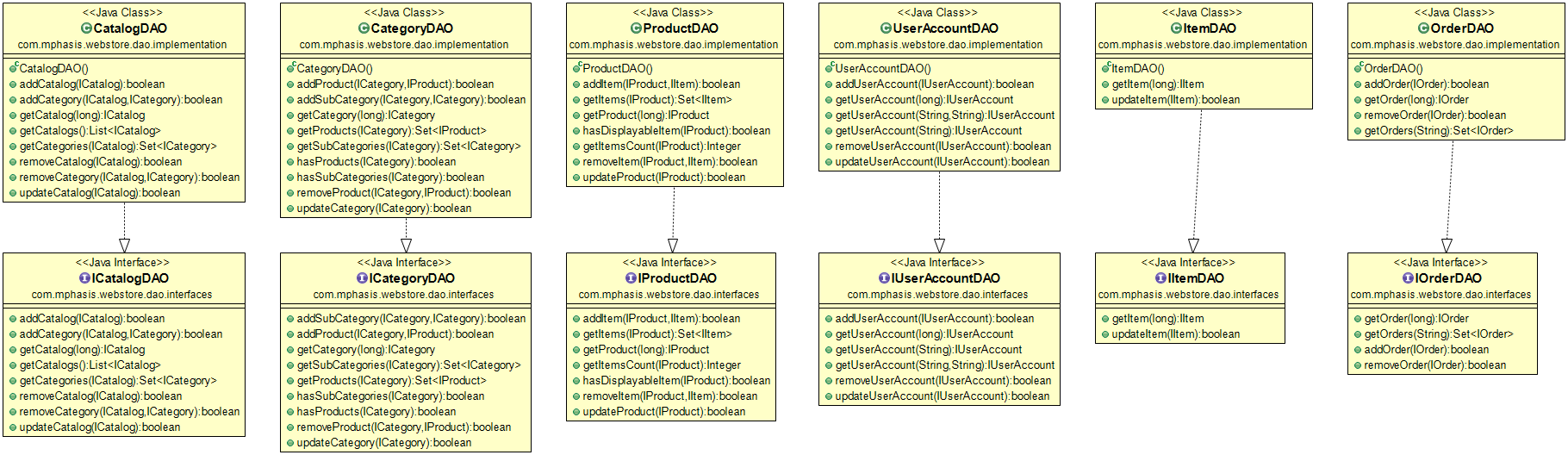
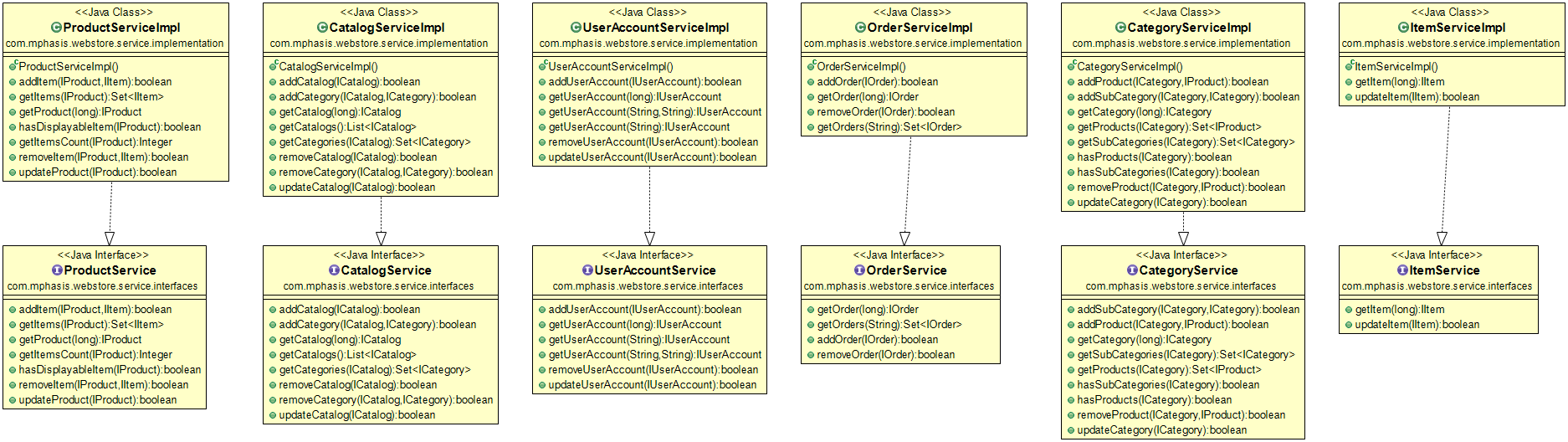
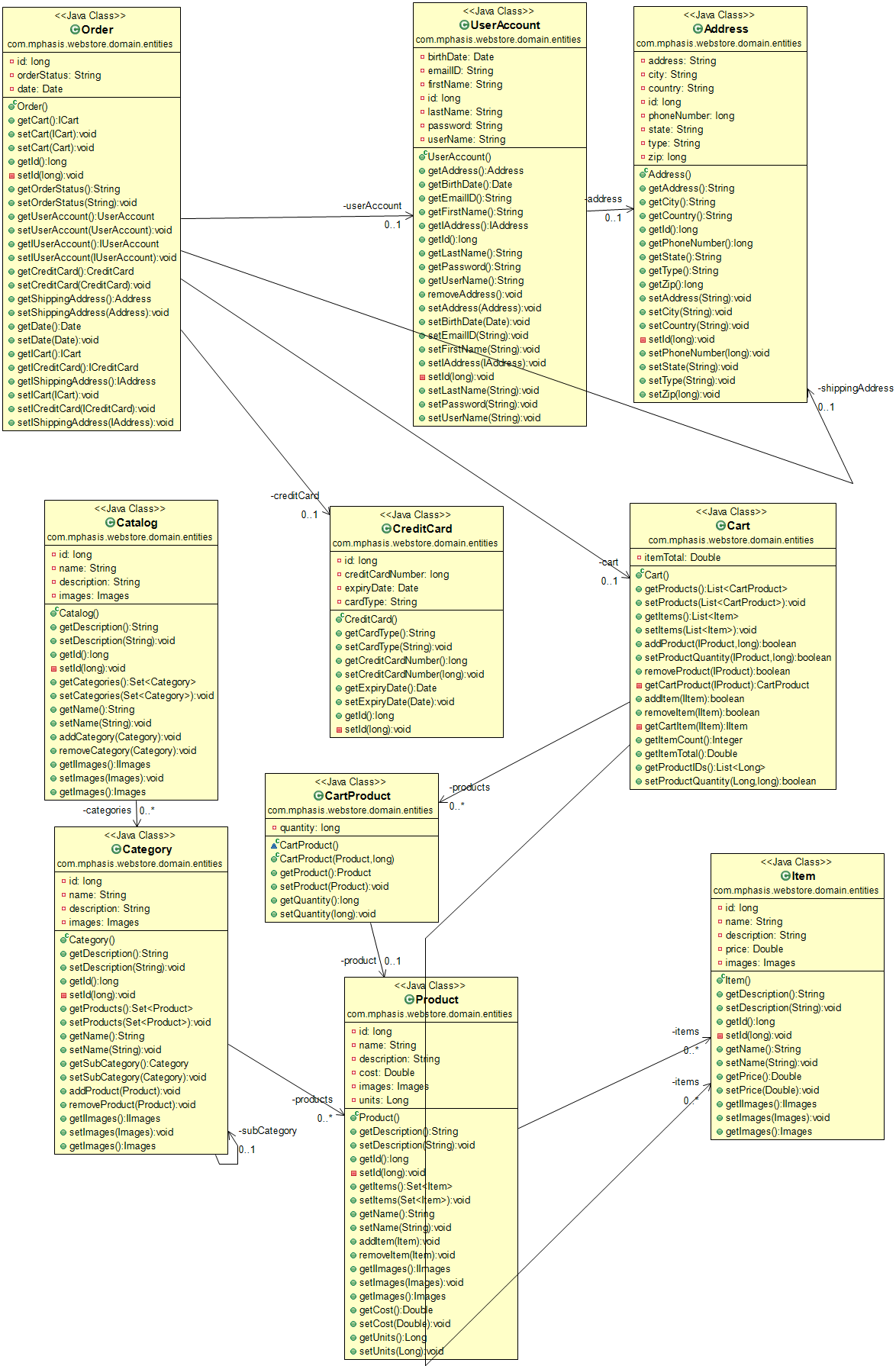
Dao classes



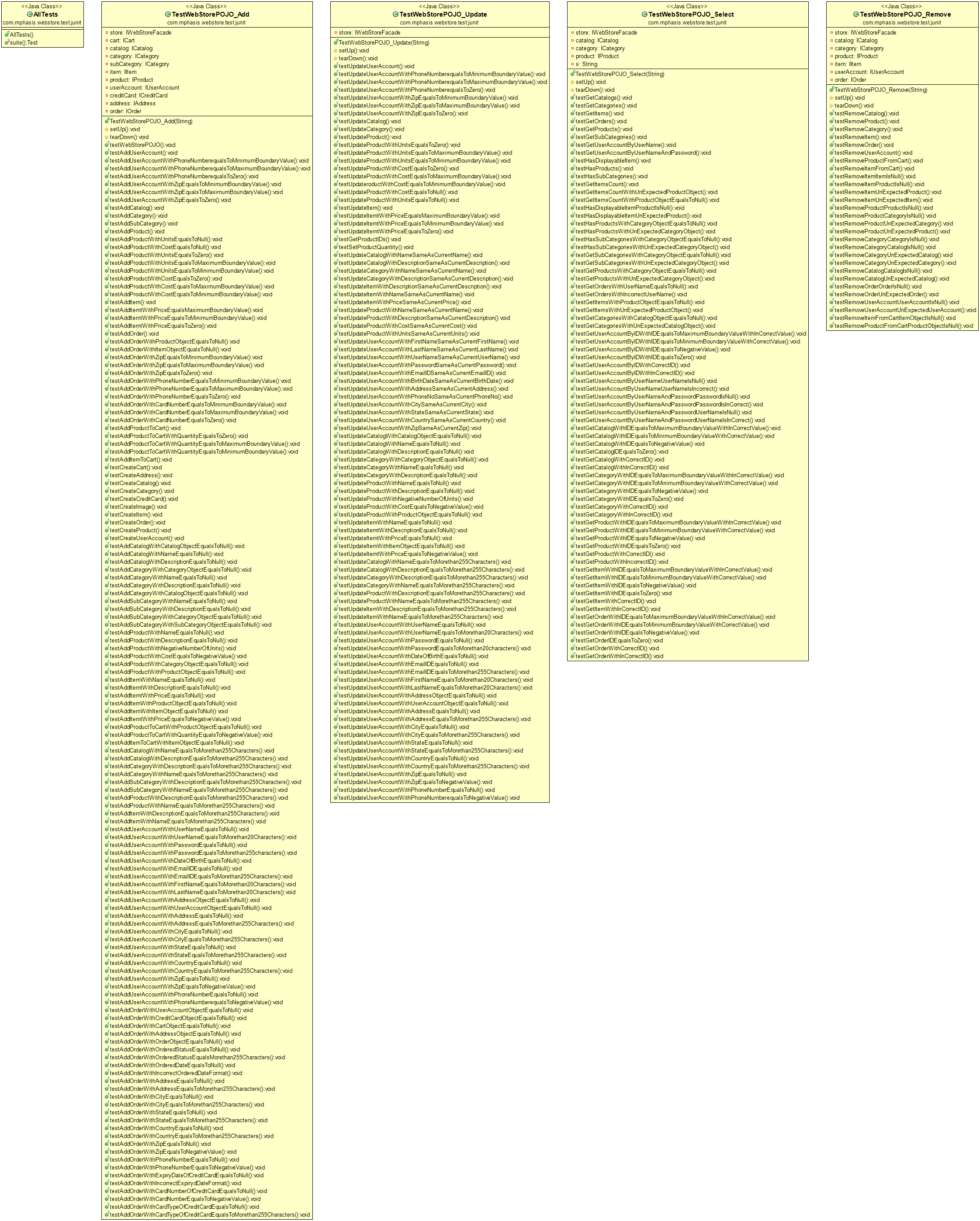
Service Classes

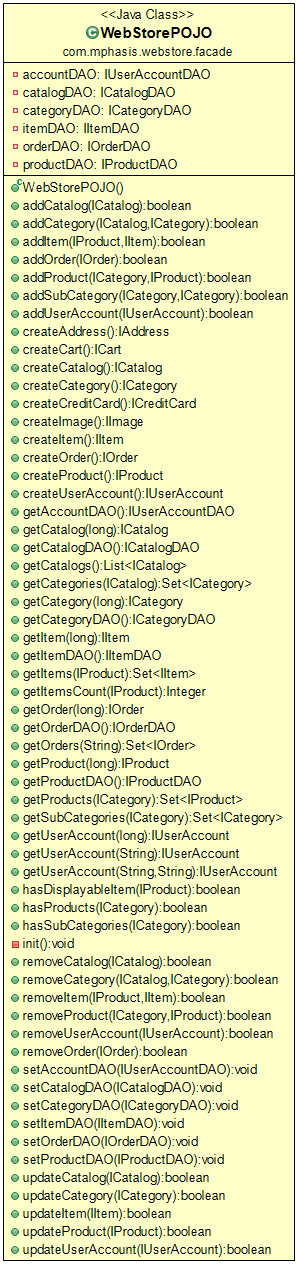


Domain Entity classes



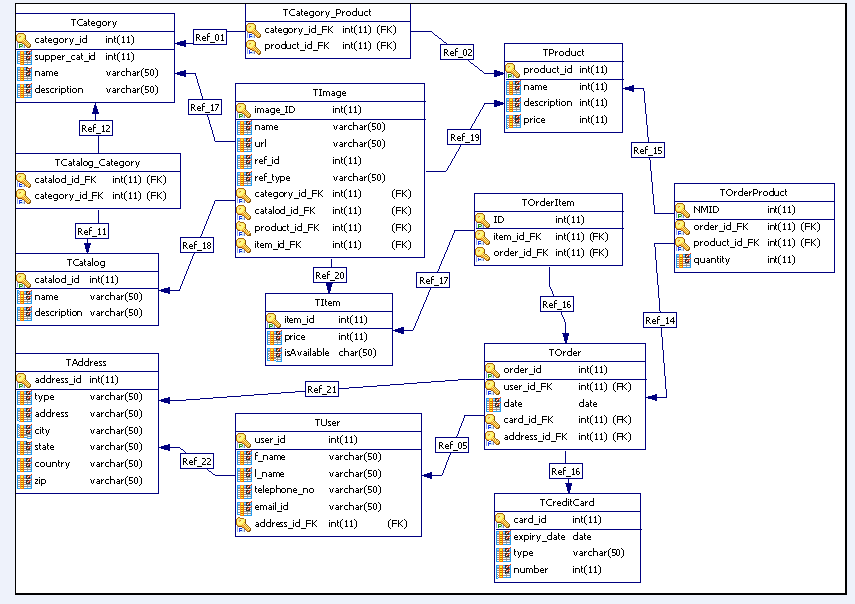
Junit Test case classes





## Logical Database Requirements

### E-R Diagram



## Global Functions Description

The global functions which are used in this application are described as following.

## 2.5.1 addCatalog (ICatalog, catalog)

## 2.5.1.1 Function Description

This function is used to add catalogs to database. It takes an object of type ICatalog as parameter and the expected output is of Boolean data type. The expected output is, if the specified catalog is added to database, otherwise the expected output will be false.

## 2.5.1.2 Global data structure usage

## No data structures are used in this function.

## 2.5.1.3 Processing Logic

This function takes an object of type ICatalog as argument and saves that object in hibernate session and adds it to the database. This function throws ExceptionInitializerError, if hibernate session factory is not created. To handle this, create hibernate session factory.

## 2.5.2 addCategory (ICatalog catalog, ICategory category)

## 2.5.2.1 Function Description

This function is used to add categories under the existing catalogs in the database. It takes two objects, one of type ICatalog and second of type ICategory as arguments and the expected output is of Boolean data type. The expected output is true, if the specified category is added to database under the specified catalog, otherwise the expected output will be false.

## 2.5.2.2 Global data structure usage

## No data structures are used in this function.

## 2.5.2.3 Processing Logic

This function saves the object of type ICatalog in hibernate session and adds it to the database and updates the catalog table in the database.

This function throws hibernate transaction exception, if the hibernate session is not able to start/begin the transaction. To handle this begin hibernate session’s transaction.

## 2.5.3 getCatalogs ()

## 2.5.3.1 Function Description

This function is used to get all the catalogs from database. This function doesn’t take an input and the expected output is of type List. The expected output is the list of catalogs, if this function gets all the available categories from database, otherwise the expected output will be null.

## 2.5.3.2 Global data structure usage

## The data structure used in this function is List.

## 2.5.3.3 Processing Logic

This function creates a select query to get all the catalogs from database and returns the catalogs in the form of List. This function throws hibernate transaction exception, if the hibernate session is not able to start/begin the transaction. To handle this begin hibernate session’s transaction.

## 2.5.4 getCatalog (long id)

## 2.5.4.1 Function Description

This function provides the logic to get the catalog from database, if the catalog with specified ID exists. This function takes the input of type long. The expected output is the catalog with specified ID, if the catalog with specified ID is available in the database, otherwise the expected output will be null.

## 2.5.4.2 Global data structure usage

## No data structures are used in this function.

## 2.5.4.3 Processing Logic

This function gets the catalog with specified ID from database by calling get method on hibernates session.

The get method takes two arguments, one is catalog class name and the other is id. This function throws hibernate transaction exception, if the hibernate session is not able to start/begin the transaction. To handle this begin hibernate session’s transaction.

## 2.5.5 getCategories(ICatalog catalog)

## 2.5.5.1 Function Description

This function gets the available categories under the specified catalog from database, if any categories under the provided catalog exist. This function takes the input of type ICatalog. The expected output the set of categories, if any categories are available under specified catalog, otherwise the expected output will be null.

## 2.5.5.2 Global data structure usage

## The data structure used in this function is Set

## 2.5.5.3 Processing Logic

This function initializes the catalog object and uses the getter method of categories to get all the categories under the specified catalog. This function throws two types of exceptions

(i) Hibernate transaction exception, if the hibernate session is not able to start/begin the transaction. To handle this begin hibernate session’s transaction.

(ii)Lazy Initialization Exception, if the function cannot load the categories under the catalog from database. To handle this initialize the catalog object, before getting the categories.

## 2.5.6 removeCatalog(ICatalog catalog)

## 2.5.6.1 Function Description

This function removes the specified catalog row/object from catalogs table in the database, if the specified catalog exists. This function takes an object of type ICatalog as input. The expected output is of Boolean type and it will be true if the specified catalog is successfully removed from database, otherwise it will be false.

## 2.5.6.2 Global data structure usage

## No data structures are used in this function.

## 2.5.6.3 Processing Logic

This function performs delete operation on hibernates session to remove the specified catalog object. The delete method takes catalog object as argument. This function throws Hibernate transaction exception, if the hibernate session is not able to start/begin the transaction. To handle this begin hibernate session’s transaction.

## 2.5.1.7 removeCategory(ICategory category)

## 2.5.7.1 Function Description

This function removes the specified category row/object from categories table in the database, if the specified category exists. This function takes an object of type ICategory as input. The expected output is of Boolean type and it will be true if the specified category is successfully removed from database, otherwise it will be false.

## 2.5.7.2 Global data structure usage

## No data structures are used in this function.

## 2.5.7.3 Processing Logic

This function performs delete operation on hibernates session to remove the specified category object. The delete method takes category object as argument. This function throws Hibernate transaction exception, if the hibernate session is not able to start/begin the transaction. To handle this begin hibernate session’s transaction.

## 2.5.8 updateCatalog (ICatalog catalog)

## 2.5.8.1 Function Description

This function updates the specified category row/object in categories table in the database, if there are any changes made to the specified category. This function takes an object of type ICatalog as input. The expected output is of Boolean type and it will be true if the specified catalog is successfully updated in the database, otherwise it will be false.

## 2.5.8.2 Global data structure usage

## No data structures are used in this function.

## 2.5.8.3 Processing Logic

This function performs update operation on hibernates session to update the specified catalog object. The update method takes catalog object as argument. This function throws Hibernate transaction exception, if the hibernate session is not able to start/begin the transaction. To handle this, begin new transaction on hibernates session.

## 2.5.9 addProduct (ICategory category, IProduct product)

## 2.5.9.1 Function Description

This function adds the specified product to products table under the specified category in the database and updates the category object in categories table in the database. This function takes two objects, one of type ICategory and the other of type IProduct as inputs. The expected output is of type Boolean and the output will be true if the specified product is added to database successfully, else the output will be false.

## 2.5.9.2. Global data structure usage

## This function uses no data structures.

## 2.5.9.3 Processing Logic

This function performs hibernate insert query to add the product object into products table in the database, by calling save method on hibernate session and adds the specified product object under the specified category object by calling addProduct method on category object. The save and addProduct methods take product object as argument. And this function updates the category by performing hibernate update query by calling update method on hibernate session. The update method takes category object as an argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.10 addSubCategory (ICategory category, ICategory subCategory)

## 2.5.10.1 Function Description

This function adds the specified subcategory to categories table under the specified category in the database and updates the category object in categories table in the database. This function takes two objects of type ICategory as inputs. The expected output is of type Boolean and the output will be true if the specified subcategory is added to database successfully, else the output will be false.

## 2.5.10.2 Global data structure usage

## No data structures are used in this function.

## 2.5.10.3 Processing Logic

This function performs hibernate insert query to add the subcategory object into categories table in the database, by calling save method on hibernate session and adds the specified subcategory object under the specified category object by calling setSubCategory method on subcategory object. The save method takes subcategory object as argument and setSubCategory method takes category object as argument. And this function updates the category by performing hibernate update query, by calling update method on hibernate session. The update method takes category object as an argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.11 getCategory (long id)

## 2.5.11.1 Function Description

This function provides the logic to get a category from database, if the category with specified id exists. This function takes id of type long as input. The expected output is a category object with, if the category with specified id is available in the database, else the expected output will be null.

## 2.5.11.2 Global data structure usage

## This function uses no data structure.

## 2.5.11.3 Processing Logic

This function calls the get method on hibernate session to get the category with specified id from database. The get method takes two arguments, one is the class name of category and the other is id. This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.12 getSubCategories(ICategory category)

## 2.5.12.1 Function Description

This function gets all the available subcategories under the specified category from categories table in database, if any subcategories under the provided category exist. This function takes an object of type ICategory as input. The expected output is of type set of objects and the output will be Set of subcategory objects, if there are any subcategories available under the specified category, else the output will be null.

## 2.5.12.2 Global data structure usage

## The data structures used in this function are Set and List.

## 2.5.12.3 Processing Logic

This function creates hibernate select query to get all the subcategories present under the specified category from categories table in database, by calling createQuery method on hibernate session. The createQuery method takes the input of type String. This String contains the query to select the subcategories from categories table, by specifying the id. The subcategories are obtained in the form of list of objects, by calling list method on query object. And if the list is not null, it will be casted to set by creating a new HashSet object of type ICategory, by passing the list as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.13 getProducts(ICategory category)

## 2.5.13.1 Function Description

This function gets all the available products under the specified category from products table in database, if any products are available under the provided category. This function takes an object of type ICategory as input. The expected output is of type set of objects and the output will be Set of product objects, if there are any products available under the specified category, else the output will be null.

## 2.5.13.2 Global data structure usage

## The data structure used in this function is Set.

## 2.5.13.3 Processing Logic

This function first initializes the specified category object, by calling initialize method on hibernate session and gets all the products that are available under the specified category in the form of set of product objects, by calling getProducts method on category object. If the obtained set is not null, then the function will return a new HashSet of objects of type IProduct, by passing the set as argument.

This function throws two types of exceptions

(i) Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

(ii)Lazy Initialization Exception, if the function cannot load the products present under the category from database. To handle this initialize the category object, by calling initialize method hibernate session before calling getProducts method.

## 2.5.14 hasProducts (ICategory category)

## 2.5.14.1 Function Description

This function checks whether there are any products available or not under the specified category in database. This function takes an object of type ICategory as input. The expected output is of type Boolean and the output will be true, if there are any products present under the specified category, else the output will be false.

## 2.5.14.2 Global data structure usage

## This data structure used in this function is Set.

## 2.5.14.3 Processing Logic

This function first initializes the specified category object, by calling initialize method on hibernate session and gets all the products that are available under the specified category in the form of set of product objects, by calling getProducts method on category object. If the obtained set is not null, then the function finds the size of the set by calling size method on the obtained set. If the size is greater than zero, then this function returns true. Otherwise it returns false.

This function throws two types of exceptions

(i) Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

(ii)Lazy Initialization Exception, if the function cannot load the products present under the category from database. To handle this initialize the category object, by calling initialize method hibernate session before calling getProducts method.

## 2.5.15 hasSubcategories (ICategory category)

## 2.5.15.1 Function Description

This function checks whether there are any subcategories available or not under the specified category in database. This function takes an object of type ICategory as input. The expected output is of type Boolean and the output will be true, if there are any subcategories present under the specified category, else the output will be false.

## 2.5.15.2 Global data structure usage

## This function uses no data structure.

## 2.5.15.3 Processing Logic

This function creates an SQL select query to get the count (number) of subcategories that are present under the specified category in database, by providing the subcategory id

. And the count will be returned, by calling unique Result method on query object.

If the count obtained is not null, then it will be casted to big integer and the size of count is found by calling intValue method on count variable. If the size is less than zero, this function will output false, else it will return true.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.16 removeProduct(ICategory category, IProduct product)

## 2.5.16.1 Function Description

This function removes the specified product from specified category from categories table and also from database, if the specified category & product exist. This function takes two objects, one of type ICategory and the other of type IProduct as inputs. The expected output is of type Boolean and the output will be true if the specified product is removed successfully from database, else it will be false.

## 2.5.16.2 Global data structure usage

## This function uses no data structures.

## 2.5.16.3 Processing Logic

This function performs hibernate delete query to remove the specified product object from database, by calling delete method on hibernates session and removes that product from specified category by calling removeProduct method on category object. The delete and removeProduct methods take product object as argument. This function updates (hibernate update query) the specified category by calling update method on hibernate session. The update method takes category object as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.17 updateCategory(ICategory category)

## 2.5.17.1 Function Description

This function updates the specified category object in categories table in the database, if there are any changes made to the specified category. This function takes an object of type ICategory as input. The expected output is of type Boolean and the output will be true, if the specified category is updated successfully in the database, otherwise it will be false.

## 2.5.17.2 Global data structure usage

## No data structures are used in this function.

## 2.5.17.3 Processing Logic

This function performs hibernate update query to update the specified category object, by calling update method on hibernates session. The update method takes category object as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.18 addItem (IProduct product, IItem item)

## 2.5.18.1 Function Description

This function adds the specified item to items table under the specified product in the database and updates that product object in products table in the database. This function takes two objects, one of type IProduct and the other of type IItem as inputs. The expected output is of type Boolean and the output will be true if the specified item is added to database successfully, else the output will be false.

## 2.5.18.2 Global data structure usage

## This function uses no data structures.

## 2.5.18.3 Processing Logic

This function performs hibernate insert query to add the item object into items table in the database, by calling save method on hibernate session and adds the specified item object under the specified product object by calling addItem method on product object. The save and addItem methods take item object as argument. And this function updates that product by performing hibernate update query, by calling update method on hibernate session. The update method takes product object as an argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.19 getItems(IProduct product)

## 2.5.19.1 Function Description

This function gets all the available items under the specified product from items table in database, if any items are available under the specified product. This function takes an object of type IProduct as input. The expected output is of type set of objects and the output will be Set of product objects, if there are any products available under the specified category, else the output will be null.

## 2.5.19.2 Global data structure usage

## The data structure used in this function is Set.

## 2.5.19.3 Processing Logic

This function first initializes the specified category object, by calling initialize method on hibernate session and gets all the products that are available under the specified category in the form of set of product objects, by calling getProducts method on category object. If the obtained set is not null, then the function will return a new HashSet of objects of type IProduct, by passing the set as argument.

This function throws two types of exceptions

(i) Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

(ii)Lazy Initialization Exception, if the function cannot load the products present under the category from database. To handle this initialize the category object, by calling initialize method hibernate session before calling getProducts method.

## 2.5.20 getProduct (long id)

## 2.5.20.1 Function Description

This function provides the logic to get a product from database, if any product with specified id exists. This function takes id of type long as input. The expected output is of type object. This function returns a product object, if the product with specified id is available in the database, else the expected output will be null.

## 2.5.20.2 Global data structure usage

## This function uses no data structure.

## 2.5.20.3 Processing Logic

This function calls the load method on hibernate session to get the product with specified id from database. The load method takes two arguments, one is the class name of product and the other is id.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.21 hasDisplayableItems (IProduct product)

## 2.5.21.1 Function Description

This function checks whether there are any items available or not under the specified product in database. This function takes an object of type IProduct as input. The expected output is of type Boolean and the output will be true, if there are any items present under the specified category, else the output will be false.

## 2.5.21.2 Global data structure usage

## This function uses no data structure.

## 2.5.21.3 Processing Logic

This function creates an SQL select query to get the count (number) of items that are present under the specified product in database, by providing the product id. And the count will be returned, by calling uniqueResult method on query object. If the count obtained is not null, then it will be casted to big integer and the size of count is found by calling intValue method on count variable. If the size is less than zero, this function will output false, else it will return true.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.22 getItemsCount (IProduct product)

## 2.5.22.1 Function Description

This function gets the number of items that are available under the specified product in database. This function takes an object of type IProduct as input. The expected output is of type Integer and the output will be the number of items present under the specified product.

## 2.5.22.2 Global data structure usage

## This function uses no data structure.

## 2.5.22.3 Processing Logic

This function creates an SQL select query to get the count (number) of items that are present under the specified product in database, by providing the product id. And the count will be returned, by calling uniqueResult method on query object. If the count obtained is not null, then it will be casted to big integer and the size of count is found by calling intValue method on count variable.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.23 removeItem(IProduct product, IItem item)

## 2.5.23.1 Function Description

This function removes the specified item under specified product from items table in database, if the specified product & item exist. This function takes two objects, one of type IProduct and the other of type IItem as inputs. The expected output is of type Boolean and the output will be true if the specified item is removed successfully from database, else it will be false.

## 2.5.23.2 Global data structure usage

## This function uses no data structures.

## 2.5.23.3 Processing Logic

This function performs hibernate delete query to remove the specified item object from database, by calling delete method on hibernates session and removes that item under specified category by calling removeItem method on product object. The delete and removeItem methods take item object as argument. This function also updates (hibernate update query) the specified product by calling update method on hibernate session. The update method takes product object as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.24 updateProduct(IProduct product)

## 2.5.24.1 Function Description

This function updates the specified product object in products table in the database, if there are any changes made to the specified product. This function takes an object of type IProduct as input. The expected output is of type Boolean and the output will be true, if the specified product is updated successfully in the database, otherwise it will be false.

## 2.5.24.2 Global data structure usage

## No data structures are used in this function.

## 2.5.24.3 Processing Logic

This function performs hibernate update query to update the specified product object, by calling update method on hibernates session. The update method takes product object as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.25 getItem (long id)

## 2.5.25.1 Function Description

This function provides the logic to get an item from database. This function takes id of type long as input. The expected output is of type object. This function returns an item object, if the item with specified id is available in the database, else the expected output will be null.

## 2.5.25.2 Global data structure usage

## This function uses no data structure.

## 2.5.25.3 Processing Logic

This function calls the load method on hibernate session to get the item with specified id from database. The load method takes two arguments, one is the class name of item and the other is id.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.26 updateItem(IItem item)

## 2.5.26.1 Function Description

This function updates the specified item object in items table in the database, if there are any changes made to the specified item. This function takes an object of type IItem as input. The expected output is of type Boolean and the output will be true, if the specified item object is updated successfully in the database, otherwise it will be false.

## 2.5.26.2 Global data structure usage

## No data structures are used in this function.

## 2.5.26.3 Processing Logic

This function performs hibernate update query to update the specified item object, by calling update method on hibernates session. The update method takes an object of type IItem as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.27 addOrder (IOrder order)

## 2.5.27.1 Function Description

This function adds the specified order to orders table in the database. This function takes an object of type IOrder as input. The expected output is of type Boolean and the output will be true if the specified order is added to database successfully, else the output will be false.

## 2.5.27.2 Global data structure usage

## This function uses no data structures.

## 2.5.27.3 Processing Logic

This function performs hibernate insert query to add the order object into orders table in the database, by calling save method on hibernate session. The save method takes order object as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.28 getOrders (String userName)

## 2.5.28.1 Function Description

This function gets all the orders from orders table in the database. This function takes userName of type String as input. The expected output is of type set of objects and the output will be Set of order objects, if there are any orders associated with specified user, else the output will be null.

## 2.5.28.2 Global data structure usage

## The data structures used in this function are Set and List.

## 2.5.28.3 Processing Logic

This function creates hibernate select query to get all the orders present in orders table in the database, by calling createQuery method on hibernate session. The createQuery method takes an input of type String. This String contains the query to select the orders from orders table, by specifying the userName. The orders are obtained in the form of list of objects, by calling list method on query object. And if the list is not null, it will be casted to set by creating a new Set objects of type IProduct, by passing the list as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.29 getOrder (long id)

## 2.5.29.1 Function Description

This function provides the logic to get an order from database. This function takes id of type long as input. The expected output is of type object. This function returns an order object, if the order with specified id is available in the database, else the expected output will be null.

## 2.5.29.2 Global data structure usage

## This function uses no data structure.

## 2.5.29.3 Processing Logic

This function calls the load method on hibernate session to get the order with specified id from database. The load method takes two arguments, one is the class name of order and the other is id.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.30 addUserAccount (IUserAccount account)

## 2.5.30.1 Function Description

This function adds the specified user account object to useraccount table in the database. This function takes an object of type IUserAccount as input. The expected output is of type Boolean and the output will be true if the specified user account is added to database successfully, else the output will be false.

## 2.5.30.2 Global data structure usage

## This function uses no data structures.

## 2.5.30.3 Processing Logic

This function performs hibernate insert query to add the user account object into useraccount table in the database, by calling save method on hibernate session. The save method takes user account object as argument.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.31 getUserAccount (long id)

## 2.5.31.1 Function Description

This function provides the logic to get a user account object from useraccount table in database. This function takes id of type long as input. The expected output is of type object. This function returns a user account object, if any user account with specified id is available in the database, else the expected output will be null.

## 2.5.31.2 Global data structure usage

## This function uses no data structure.

## 2.5.31.3 Processing Logic

This function calls the get method on hibernate session to get the user account with specified id from database. The get method takes two arguments, one is the class name of user account and the other is id.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.32 getuserAccount (String userName, String password)

## 2.5.32.1 Function Description

This function gets a user account row/object from user account table in the database. This function takes userName and password of type String as inputs. The expected output is of type object and the output will be user account object, if there is any user account present in the database with specified userName and password, else the output will be null.

## 2.5.32.2 Global data structure usage

## This function uses no data structure.

## 2.5.32.3 Processing Logic

This function creates hibernate select query to get a user account present in useraccount table in the database, by calling createQuery method on hibernate session. The createQuery method takes a String value as input. This String contains the query to select the user account from useraccount table, by specifying the userName and password. The user account is obtained in the form of object, by calling uniqueResult method on query object.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.33 getuserAccount (String userName)

## 2.5.33.1 Function Description

This function gets a user account row/object from user account table in the database. This function takes userName of type String as input. The expected output is of type object and the output will be user account object, if there is any user account present in database with specified userName, else the output will be null.

## 2.5.33.2 Global data structure usage

## This function uses no data structure.

## 2.5.33.3 Processing Logic

This function creates hibernate select query to get a user account present in useraccount table in the database, by calling createQuery method on hibernate session. The createQuery method takes a String value as input. This String contains the query to select the user account from useraccount table, by specifying the userName. The user account is obtained in the form of object, by calling uniqueResult method on query object.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.34 removeUserAccount(IUserAccount account)

## 2.5.34.1 Function Description

This function removes the specified user account row/object from user account table in the database, if the specified user account exists. This function takes an object of type IUserAccount as input. The expected output is of Boolean type and it will be true if the specified user account is successfully removed from database, otherwise it will be false.

## 2.5.34.2 Global data structure usage

## No data structures are used in this function.

## 2.5.34.3 Processing Logic

This function performs delete operation on hibernates session to remove the specified user account. The delete method takes user account object as input.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.35 updateUserAccount (IUserAccount account)

## 2.5.35.1 Function Description

This function updates the specified user account row/object in user account table in the database, if there are any changes made to the specified user account. This function takes an object of type IUserAccount as input. The expected output is of Boolean type and it will be true if the specified user account is successfully updated in the database, otherwise it will be false.

## 2.5.35.2 Global data structure usage

## No data structures are used in this function.

## 2.5.35.3 Processing Logic

This function performs update operation on hibernates session to update the specified user account. The update method takes user account object as input.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.36 getCreditCard (long id)

## 2.5.36.1 Function Description

This function provides the logic to get a credit card object from creditcard table in database. This function takes id of type long as input. The expected output is of type object. This function returns a credit card object, if any credit card with specified id is available in the database, else the expected output will be null.

## 2.5.36.2 Global data structure usage

## This function uses no data structure.

## 2.5.36.3 Processing Logic

This function calls the get method on hibernate session to get the credit card with specified id from database. The get method takes two arguments, one is the class name of credit card and the other is id.

This function throws Hibernate transaction exception, if no transaction instance is created. To handle this, begin a new transaction by calling beginTransaction method on hibernate session which returns a transaction instance.

## 2.5.37 addCategory (Category category)

## 2.5.37.1 Function Description

This function adds the specified category to the set of categories. This function takes an object of type Category as input. The expected output is of type void.

## 2.5.37.2 Global data structure usage

## The data structure used in this function is Set.

## 2.5.37.3 Processing Logic

This function adds the specified category object to the set of category objects, by calling add method on the set. The set method takes category object as input.

## 2.5.38 removeCategory (Category category)

## 2.5.38.1 Function Description

This function removes the specified category object from set of category objects. This function takes an object of type Category as input. The expected output is of void type and this function removes the specified category object from the set, only if the category object is present in the set.

## 2.5.38.2 Global data structure usage

## The data structure used in this function is Set.

## 2.5.38.3 Processing Logic

This function removes the specified category from the set of category objects, by calling remove method on the set. The remove method takes category object as input.

## 2.5.39.1 addProduct (Product product)

## 2.5.39 Function Description

This function adds the specified product to the set of products. This function takes an object of type Product as input. The expected output is of type void.

## 2.5.39.2 Global data structure usage

## The data structure used in this function is Set.

## 2.5.39.3 Processing Logic

This function adds the specified product object to the set of product objects, by calling add method on the set. The add method takes product object as input.

## 2.5.40 removeProduct(Product product)

## 2.5.40.1 Function Description

This function removes the specified product object from set of products. This function takes an object of type Product as input. The expected output is of void type and this function removes the specified product object from the set, only if it is present in the set.

## 2.5.40.2 Global data structure usage

## The data structure used in this function is Set.

## 2.5.40.3 Processing Logic

This function removes the specified product from the set of product objects, by calling remove method on the set. The remove method takes product object as input.

## 2.5.41 addImage (IImage image)

## 2.5.41.1 Function Description

This function adds the specified image to the set of images. This function takes an object of type IImage as input. The expected output is of type void.

## 2.5.41.2 Global data structure usage

## The data structure used in this function is Set.

## 2.5.41.3 Processing Logic

This function creates a new image object of type IImage and sets the name, path of the image and adds this object to the set of image objects, by calling add method on the set. The add method takes image object of type IImage as input.

## 2.5.42 removeProduct (Product product)

## 2.5.42.1 Function Description

This function removes the specified image from set of images. This function takes an object of type IImage as input. The expected output is of void type and this function removes the specified image object from the set, only if it is present in the set.

## 2.5.42.2 Global data structure usage

## The data structure used in this function is Set.

## 2.5.42.3 Processing Logic

This function removes the specified image from the set of image objects, by calling remove method on the set. The remove method takes image object of type IImage as input.

# Software Organization

The software used to develop this application is organized in different folders. The folder structure is explained as follows.

The **src** folder contains packages which include all the classes. Each package is explained as follows.

1. com.mphasis.webstore.dao.interfaces

This package has six interfaces. They are,

1. ICatalogDAO.java
2. ICAtegoryDAO.java
3. IItemDAO.java
4. IOrderDAO.java
5. IProductDAO.java
6. IUserAccounyDAO.java
7. com.mphasis.webstore.dao.implementation

This package has six classes. Each class implements the methods defined in the above interfaces.

* 1. CatalogDAO.java
  2. CAtegoryDAO.java
  3. ItemDAO.java
  4. OrderDAO.java
  5. ProductDAO.java
  6. UserAccounyDAO.java

1. com.mphasis.webstore.service.interfaces

This package has six interfaces. They are,

1. CatalogService.java
2. CAtegoryService.java
3. ItemService.java
4. OrderService.java
5. ProductService.java
6. UserAccounyService.java
7. com.mphasis.webstore.service.implementation

This package has six classes. Each class implements the methods defined in the above interfaces.

* 1. CatalogServiceImpl.java
  2. CAtegoryServiceImpl.java
  3. ItemServiceImpl.java
  4. OrderServiceImpl.java
  5. ProductServiceImpl.java
  6. UserAccounyServiceImpl.java

1. com.mphasis.webstore.domain.interfaces

This package has eleven interfaces. They are,

* 1. IAddress.java
  2. ICart.java
  3. ICatalog.java
  4. ICategory.java
  5. ICreditCard.java
  6. IImage.java
  7. IImages.java
  8. IItem.java
  9. IOrder.java
  10. IProduct.java
  11. IUserAccount.java
  12. IWebStoreFacade.java

1. com.mphasis.webstore.domain.entities

This package has twelve classes which implement the methods defined in the above interfaces except for the methods in IWebStoreFacade interface.

* 1. Address.java
  2. Cart.java
  3. CartProduct.java
  4. Catalog.java
  5. Category.java
  6. CreditCard.java
  7. Image.java
  8. Images.java
  9. Item.java
  10. Order.java
  11. Product.java
  12. UserAccount.java

1. com.mphasis.webstore.domain.constants

This package has three interfaces. Each interface has two static variables of type String.

1. IAddressType.java
2. ICreditCardType.java
3. IOrderStatus.java
4. com.mphasis.webstore.facade

This package has one class which implements all the methods in IWebStoreFacade interface.

* 1. WebStorePOJO.java

1. com.mphasis.webstore.test.junit

This package has five (Junit test) classes which are used to test the methods defined in DAO classes.

1. AllTests.java
2. TestWebStorePOJO\_Add.java
3. TestWebStorePOJO\_Remove.java
4. TestWebStorePOJO\_Select.java
5. TestWebStorePOJO\_Update.java

## Introduction

**Database Module:-**

This module describes the implementation of database (backend) for the application. All the data related to this application is stored in database tables. If any operation is performed by user on the application, the changes will be reflected in the database.

**Business Module:-**

This module describes the business logic to be implemented in order to meet the requirements of the application. The implementation of this module follows MVC (Model-View-Control) architecture. This module acts as Model. The model is a collection of Java classes that form the application.

**Web Module:-**

This module describes the view & controller part of MVC architecture. The view is represented by a Spring MVC, with data being transported to the page in the Request or Session.

The Controller communicates with the front end of the model and loads the Spring Controller classes or Spring beans with appropriate data, before forwarding the Spring controller and Response to the JSP using a MoodelAndView.

## Design Constraints and Considerations

### Design Constraints

* More than 10000 users cannot access the Web store at a time.
* Customer cannot order more than 1000 items of one type at a time.

### Design Considerations

There are several design considerations which are to be taken into account while developing this product. They are:

* + - What are the goals of the web-site?
    - Who are the target audience?
    - What do you want your site to look like?
    - Where is the site to be hosted?
    - What are the success criteria for your site?

You will have to design web-pages that can be displayed by many different browsers. Common browsers include Microsoft Internet Explorer, Mozilla Firefox, Apple Safari and Google Chrome.

Each page should contain a title, an author, copyright information and a link to home page of the site.

You should test your web-pages in as many browsers and different operating systems as possible during the development process to ensure that they will display properly. You should also ensure that your web-pages work in previous versions of the browsers, as some of your audience may not have updated their software.