Homework 2: Command Pattern

Group: SNAL (Sammy Edwards, Andi Fuerst, Nathan Kawula, Logan Thompson)

1. Describe which two operations to undo and redo

**The first operation to undo and redo is for the creation of a new transaction**

**The second operation to undo and redo is for the creation of a new category**

2. identify your client, invoker, receiver, and concrete commands.

**Client: MoneyBeeForm**

**Invoker: User (MoneyBeeForm)**

**Receiver: Handler**

**Concrete Commands: AddCategoryCommand, AddTransactionCommand**

3. Include code segments of the implementation of your concrete commands' execute and unexecute methods.

// Inserts a new transaction for the user and sends all transaction

// information to different parts of the system i.e the database,

// U.I forms, and the command stacks(undo/redo).

// params: (in, in)

//---------------------------------------------------------------------

public void insertTransaction(Transaction transaction, int primaryKey)

{

databaseManager.addTransaction(transaction);

int transactionKey = databaseManager.getTransactionPrimaryKey(transaction);

if (transaction.getCategory() != null) // if the category is associated with a category

{

int categoryKey = databaseManager.getCategoryPrimaryKey(transaction.getCategory());

databaseManager.linkCategoryToTransaction(categoryKey, transactionKey);

transaction.getCategory().redoTransaction(transaction.getPrice());

databaseManager.updateCategoryToDatabase(transaction.getCategory(), categoryKey);

}

databaseManager.linkTransactionToUser(primaryKey, transactionKey);

}

//---------------------------------------------------------------------

// Removes a transction from the user and send all transaction information

// to different part of the system i.e the database, U.I forms, and the

// command stacks(undo/redo).

// params: (in, in)

//---------------------------------------------------------------------

public void deleteTransaction(Transaction transaction, int primaryKey)

{

if (transaction.getCategory() == null) // if the transaction is not associated with a category

{

int transactionKey = databaseManager.getTransactionPrimaryKey(transaction);

int transactionLinkKey = databaseManager.getTransactionLinkKey(transactionKey);

databaseManager.deleteTransactionUserLink(transactionLinkKey);

databaseManager.deleteTransaction(transactionKey);

}

else // if the transation is associated with a category

{

int transactionKey = databaseManager.getTransactionPrimaryKey(transaction);

int transactionLinkKey = databaseManager.getTransactionLinkKey(transactionKey);

int categoryTransactionLinkKey = databaseManager.getCategoryTransactionLinkKey(transactionKey);

int categoryKey = databaseManager.getCategoryPrimaryKey(transaction.getCategory());

transaction.getCategory().undoTransaction(transaction.getPrice());

databaseManager.updateCategoryToDatabase(transaction.getCategory(), categoryKey);

databaseManager.deleteCategoryTransactionLink(categoryTransactionLinkKey);

databaseManager.deleteTransactionUserLink(transactionLinkKey);

databaseManager.deleteTransaction(transactionKey);

}

}

//---------------------------------------------------------------------

// Inserts a new cateogry for the user and sends all category

// information to different parts of the system i.e the database,

// U.I forms, and the command stacks(undo/redo).

// params: (in, in)

//---------------------------------------------------------------------

public void insertCategory(Category category, int primaryKey)

{

databaseManager.addCategory(category);

int categoryKey = databaseManager.getCategoryPrimaryKey(category);

databaseManager.linkCategoryToUser(primaryKey, categoryKey);

}

//---------------------------------------------------------------------

// Removes a category from the user and send all category information

// to different part of the system i.e the database, U.I forms, and the

// command stacks(undo/redo).

// params: (in, in)

//---------------------------------------------------------------------

public void deleteCategory(Category category, int primaryKey)

{

int categoryKey = databaseManager.getCategoryPrimaryKey(category);

int categoryLinkKey = databaseManager.getCategoryUserLinkKey(categoryKey);

databaseManager.deleteUserCategoryLink(categoryLinkKey);

databaseManager.deleteCategory(categoryKey);

}

//---------------------------------------------------------------------

// Executes the removal of a transaction for the user

// params: ()

//---------------------------------------------------------------------

public override void execute()

{

base.transactionManager.deleteTransaction(transactionToAdd, primaryKey);

}

//---------------------------------------------------------------------

// Unexecutes the removal of a transaction for the user

// params: ()

//---------------------------------------------------------------------

public override void unexecute()

{

base.transactionManager.insertTransaction(transactionToAdd, primaryKey);

}

// Executes the removal of a category for the user

// params: ()

//---------------------------------------------------------------------

public override void execute()

{

base.transactionManager.deleteCategory(categoryToRemove, primaryKey);

}

//---------------------------------------------------------------------

// Unexecutes the removal of a category for the user

// params: ()

//---------------------------------------------------------------------

public override void unexecute()

{

base.transactionManager.insertCategory(categoryToRemove, primaryKey);

}

// Executes the creation of a transaction for the user

// params: ()

//---------------------------------------------------------------------

public override void execute()

{

base.transactionManager.insertTransaction(transactionToAdd, primaryKey);

}

//---------------------------------------------------------------------

// Unexecutes, undoes the creation of a transaction for the user

// params: ()

//---------------------------------------------------------------------

public override void unexecute()

{

base.transactionManager.deleteTransaction(transactionToAdd, primaryKey);

}

//---------------------------------------------------------------------

// Executes the creation of a category for the user

// params: ()

//---------------------------------------------------------------------

public override void execute()

{

base.transactionManager.insertCategory(categoryToAdd, primaryKey);

}

//---------------------------------------------------------------------

// Unexecutes, undoes the creation of a category for the user

// params: ()

//---------------------------------------------------------------------

public override void unexecute()

{

base.transactionManager.deleteCategory(categoryToAdd, primaryKey);

}

4. Include a design level class diagram of your system clearly demonstrating command pattern.

