Implementation Descriptions

"Legacy System" Application will provide an interface to access old system via new system with an option to still use old system.

Application Description

A company has two systems to access customer data. One is the new developed system and the other is the old legacy system. The interfaces of these two systems are different. The old system accesses customer data using a USB connection to read data from a database (or binary files) stored on an external disk. The new system accesses the customer data using REST API over HTTPS connection to an external cloud storage server.

Design Pattern

Used ***Adapter Method pattern*** to design as we want to convert or create a support for old API to new API.

One exception: To add a feature or easy access of old system from new system, once in new system class old customer variable is used. It could be removed easily if we want totally independent system, in that case remove *void getCustomer_WithHttpsConnect(OldCustomer o);* from the interface and class in *newsystem* package.

Assumptions

- **Customer ID :** I aimed to continue the ID in old and new system, so for this application I have **assumed** that **old system have 2 entries** and new system have entries starting from 3. No more new entries will be added to old as it is no longer in use so I can keep that value constant. This is done because in this application I am using Array list to save and access data whose indexing starts from 0, so in code logic for new system to get data for any ID I am subtracting the total old entry value that is 2. In case to test if we need **to add more old entries make changes in**:
    - totalOldData variable in NewCustomerData file in customer package
    - Test files
- New Customer ID should be +1 of the previous. Its like the primary key in db.
- We don't need to delete customers.
- Data is not imported from disk to cloud. Both places have different data.
- To access old from new system, we **pass whole old objects**. Currently it don't provide access by ID as the data is not stored anywhere but in volatile variables which go out of scope.
- **New system** have **customer type** as **a new variable** which was not in old so through adapter its value will be set as **NA** (difference in new and old systems)

Packages

- **customer** : Contains all the classes related to customer and their database.
- **legacyadapteroldtonew** : defines the adapter to convert old syte, object to objects supported by new system.
- **newsystem** : Defines the new system

- **oldsystem**: Defines the old system

General Working

- Client can print data by old system functions.
- To print from new function he needs to pass the old system object to new function **OR** through adapter class.

Classes/Interfaces/Variables

customer Package

Customer

Class defining general structure of customers.

id : int - Primary key in DB, each customer have unique id

name : String - Customer name

Customer() - constuctor

Customer(int, String) - Create an object by passing parameters

Getters & Setters

getId()

setId(int)

getName()

setName(String)

NewCustomer

Class defining general structure of customers for NEW system.

custType : String - Type of customer, its defined only in new system

NewCustomer() - Constructor

NewCustomer(int, String, String) - Create an object by passing parameters

Getters & Setters

getCustType()

setCustType(String)

NewCustomerData

Class saving and managing data of new system customers.

totalOldData : int - Final or constant variable, keeps track of the entries in old system

customerlist : List<NewCustomer> - List of all the customers in new system

addCustomer(NewCustomer) - Add new customer to new customer database

getCustomer(int) - get customer by their unique ID

Getters & Setters

getCustomerlist()

setCustomerlist(List<NewCustomer>)

OldCustomer

Class defining general structure of old customers.

OldCustomer() - Constructor

OldCustomer(int, String) - Create an object by passing parameters

OldCustomerData

Class saving and managing data of old system customers.

customerlist : List<OldCustomer> - List of all the customers in old system

addCustomer(OldCustomer) - Add new customer to new customer database

getCustomer(int) - get customer by their unique ID

Getters & Setters

getCustomerlist()

setCustomerlist(List<OldCustomer>)

legacyadapteroldtonew Package

AdapterOldToNew

Class implementing logic to convert old object to new.

newCustomer : NewCustomer - New system object that will be created by this class

AdapterOldToNew() - Constructor

setCustType(String) - Set customer type for old system to NA as it's the feature of new system

adapt(OldCustomer) - Implementing adapter pattern, converting old system object to new system

printUsingAdapter() - Print the new customer or system object created

Getters & Setters

getNewCustomer()

newsystem Package

CustomerDataOverHttps

Interface defining functionalities of new system.

printCustomer(NewCustomer) - Print new system customer data when new system object passed

printCustomer() - Print customer last retrieved using get function

getCustomer_WithHttpsConnect(int) - Get new system customer by their ID

getCustomer_WithHttpsConnect(OldCustomer) - Get old customer by using adapter

UseNewSystem

Class implementing CustomerDataOverHttps to create New System. It is the NEW SYSTEM.

customer : NewCustomer - Current customer data

data : NewCustomerData - All customers

UseNewSystem() - Constructor

UseNewSystem(NewCustomerData) - To update the data or list of new customers in new system

printCustomer()- Print customer last retrieved using get function

getCustomer_WithHttpsConnect(int) - Get new system customer by their ID

printCustomer(NewCustomer) - Print new system customer when new system object passed

getCustomer_WithHttpsConnect(OldCustomer) - Get old customer by using adapter

oldsystem Package

CustomerData

Interface defining functionalities of old system.

printCustomer(OldCustomer) - Print old system customer data when old system object passed

printCustomer()- Print customer last retrieved using get function

getCustomerWithUsbConnect(int) - Get old system customer by their ID

UseOldSystem

Class implementing CustomerData to create Old System. It is the OLD SYSTEM.

customer : OldCustomer - Current customer data

data : OldCustomerData - All customers

UseOldSystem(OldCustomerData) - Constructor and update the data or list of new customers in new system

printCustomer()- Print customer last retrieved using get function

getCustomerWithUsbConnect(int) - Get old system customer by their ID

printCustomer(OldCustomer) - Print old system customer data when old system object passed

Goals of Software Design

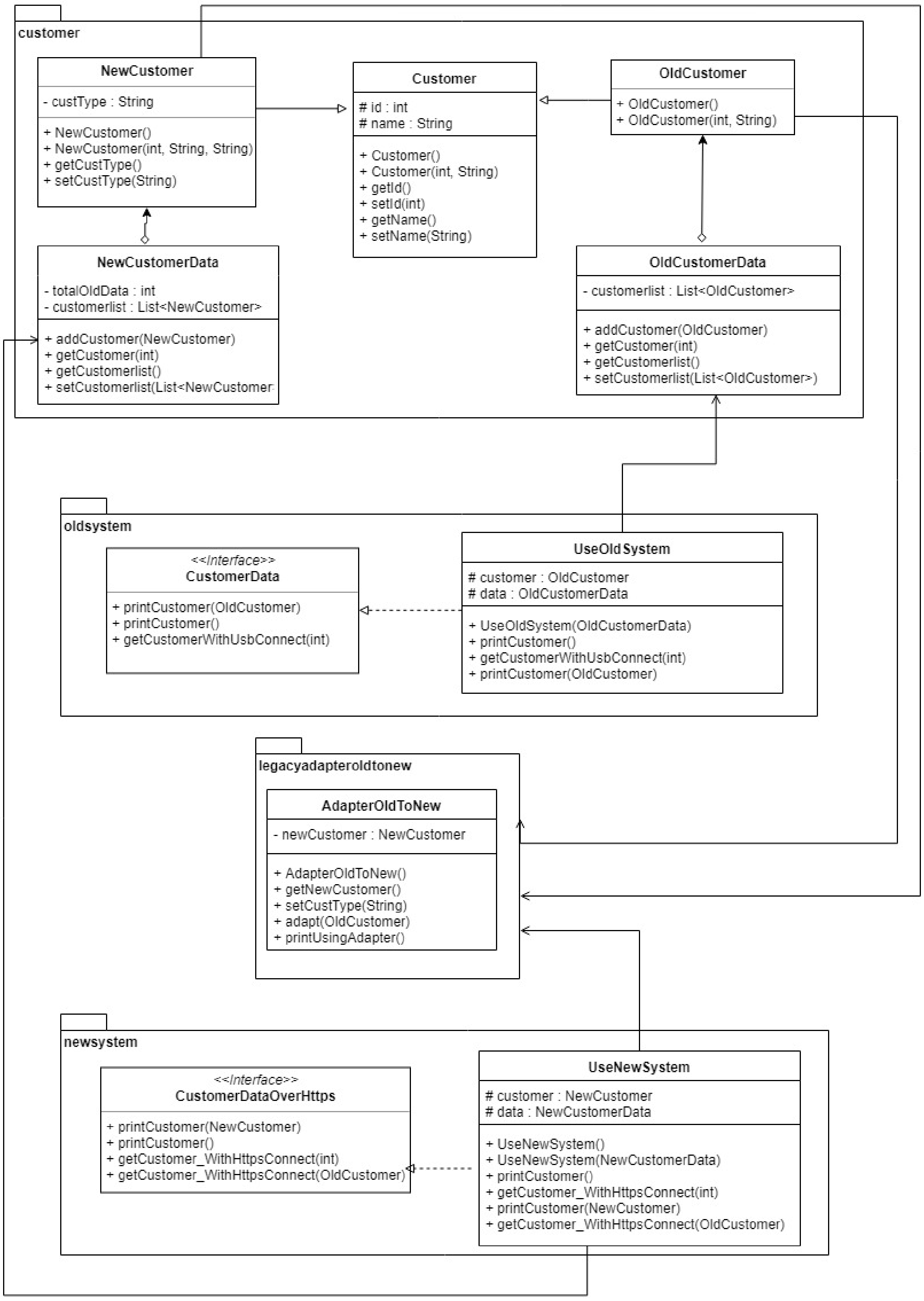
Names used in this application explains what the entity is doing, thus code is easily under stable.

There is no duplicity in the code, proper use of inheritance and implementation is done.

Every system have their own implementation independent of other thus code is easily extensible. In case adapter needs to be changed or another new system is introduced one can create packages for them and increase the functionality of the application. Thus code is modular, reusable and loosely coupled.

Code suffice all the requirements stated in the problem statement.

UML Class Diagram.


customer
NewCustomer
- custType : String
+ NewCustomer()
+ NewCustomer(int, String, String)
+ getCustType()
+ setCustType(String)
Customer
# id : int
# name : String
+ Customer()
+ Customer(int, String)
+ getId()
+ setId(int)
+ getName()
+ setName(String)
OldCustomer
+ OldCustomer()
+ OldCustomer(int, String)
NewCustomerData
- totalOldData : int
- customerlist : List<NewCustomer>
+ addCustomer(NewCustomer)
+ getCustomer(int)
+ getCustomerlist()
+ setCustomerlist(List<NewCustomer
OldCustomerData
- customerlist : List<OldCustomer>
+ addCustomer(OldCustomer)
+ getCustomer(int)
+ getCustomerlist()
+ setCustomerlist(List<OldCustomer>)
oldsystem
<<Interface>>
CustomerData
+ printCustomer(OldCustomer)
+ printCustomer()
+ getCustomerWithUsbConnect(int)
UseOld System
# customer : OldCustomer
# data : OldCustomerData
+ UseOldSystem(OldCustomerData)
+ printCustomer()
+ getCustomerWithUsbConnect(int)
+ printCustomer(OldCustomer)
legacyadapteroldtonew
AdapterOldToNew
- newCustomer : NewCustomer
+ AdapterOldToNew()
+ getNewCustomer()
+ setCustType(String)
+ adapt(OldCustomer)
+ printUsingAdapter()
newsystem
<<Interface>>
CustomerDataOverHttps
+ printCustomer(NewCustomer)
+ printCustomer()
+ getCustomer_WithHttpsConnect(int)
+ getCustomer_WithHttpsConnect(OldCustomer)
UseNewSystem
# customer : NewCustomer
# data : NewCustomerData
+ UseNewSystem()
+ UseNewSystem(NewCustomerData)
+ printCustomer()
+ getCustomer_WithHttpsConnect(int)
+ printCustomer(NewCustomer)
+ getCustomer_WithHttpsConnect(OldCustomer)