DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF NEBRASKA—LINCOLN

Sales System Database Design Document

Computer Science II Project

Authors: Olwen Nguyen & Linh K. Truong
12/3/21
Version 6.0

This project focuses on designing and implementing a new database backed sales system for a chain of Computer stores, The Matrix.

Revision History

Version	Description of Change(s)	Author(s)	Date
1.0	Initial draft of this design document	Olwen Nguyen Linh K. Truong	2021/19/09
2.0	Second revised draft, modify again the introduction which includes purpose and scope of the document, and add some information to overall design description part.	Olwen Nguyen Linh K. Truong	2021/01/10
3.0	Third revised draft, modify the introduction and overall design description, and add some information to database design description and class entity/model.	Olwen Nguyen Linh K. Truong	2021/15/10
4.0	Fourth revised draft modify the introduction, overall design description, and database design, and add some information to database interface. Write detailed information into Changes & Refactoring part.	Olwen Nguyen Linh K. Truong	2021/29/10
5.0	Fifth revised draft, write again the introduction, scope of the document, overall design description, and database design, and add some abbreviations and acronyms; and add some information to Database & Integration of Data Structures.	Olwen Nguyen Linh K. Truong	2021/14/11
6.0	Sixth revised draft, overview all the parts again, and write again database design and class entity/model. Besides, add more information about tables and classes into UML diagram and ER database diagram, and some more info into Database & Integration of Data Structures.	Olwen Nguyen Linh K. Truong	2021/03/12

Contents

1. Introduction

In order to update their business practices and to compete in the market, the client wants to modernize The Matrix (a chain of stores) by designing a new database backed sales system. This document describes the design of this new system. The project uses Java classes to model the problem and follows object – oriented programming principles.

1.1 Purpose of this Document

The goal of this article is to lay out the technical design of a new database backed sales system and provide general design and implement description of the new system.

1.2 Scope of the Project

The Matrix is a chain of stores selling and renting PCs, laptops, and systems as well as accessories, gift cards, subscriptions, and related products. The new system retrieves and updates data by using SQL database, while the old one used Excel sheets, which has some limitations when updating data. The new system was completed in phases. The first phase of the project created a layout of the system by designing java classes that modeled objects and entities needed in the Matrix system. In the second phase summary sale reports were incorporated. The final stages of the program included designing a SQL database and storing the data. And lastly, connecting java classes and SQL database using JDBC. The database interface provides an easy way to access and modify the data if needed.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

1.3.2 Abbreviations & Acronyms

JAR - Java ARchive.
JDBC - Java™ database connectivity
SQL - Structured Query Language

2. Overall Design Description

This project uses two main techniques: Java and SQL. To read data from file, create summary reports, and update data, creating Java classes following object-oriented programming and java parsers, which use encapsulation such as Person class, Item class, Store class and Sale class which have private fields being prevented from being accessed by code outside, inheritance such as Person which is super class has four subclasses including Customer, Employee, GoldCustomer and SilverCustomer so that they can inherit all the methods of Person class, abstraction such as Item class which is an abstract class contains some general methods which are specific just in classes implement Item like NewProduct and UsedProduct, polymorphism methods such as Person class and Item class which have many different forms of constructors. To make SQL database that be used extensively and flexibly, this database should have well-defined column types, primary and foreign keys. To create classes properly map to database tables using JDBC.

2.1 Alternative Design Options

Using interfaces and constructors containing all fields needed was the first solution for this project. When using that solution, have to put *null* value into fields that not used in some cases each time calling constructors. However, that solution had many disadvantages in a long process. It means whenever a new variable appears, constructor has to add a new field related to that variable. Therefore, the main constructors after all will contain many fields in that. Instead of that solution, creating many constructors with different fields in super classes is more convenient when needed because when constructors are called, there are no *null* value appear anymore and subclasses do not have to extend or implement many super classes or interfaces.

3. Detailed Component Description

Detailed design and testing strategy of components.

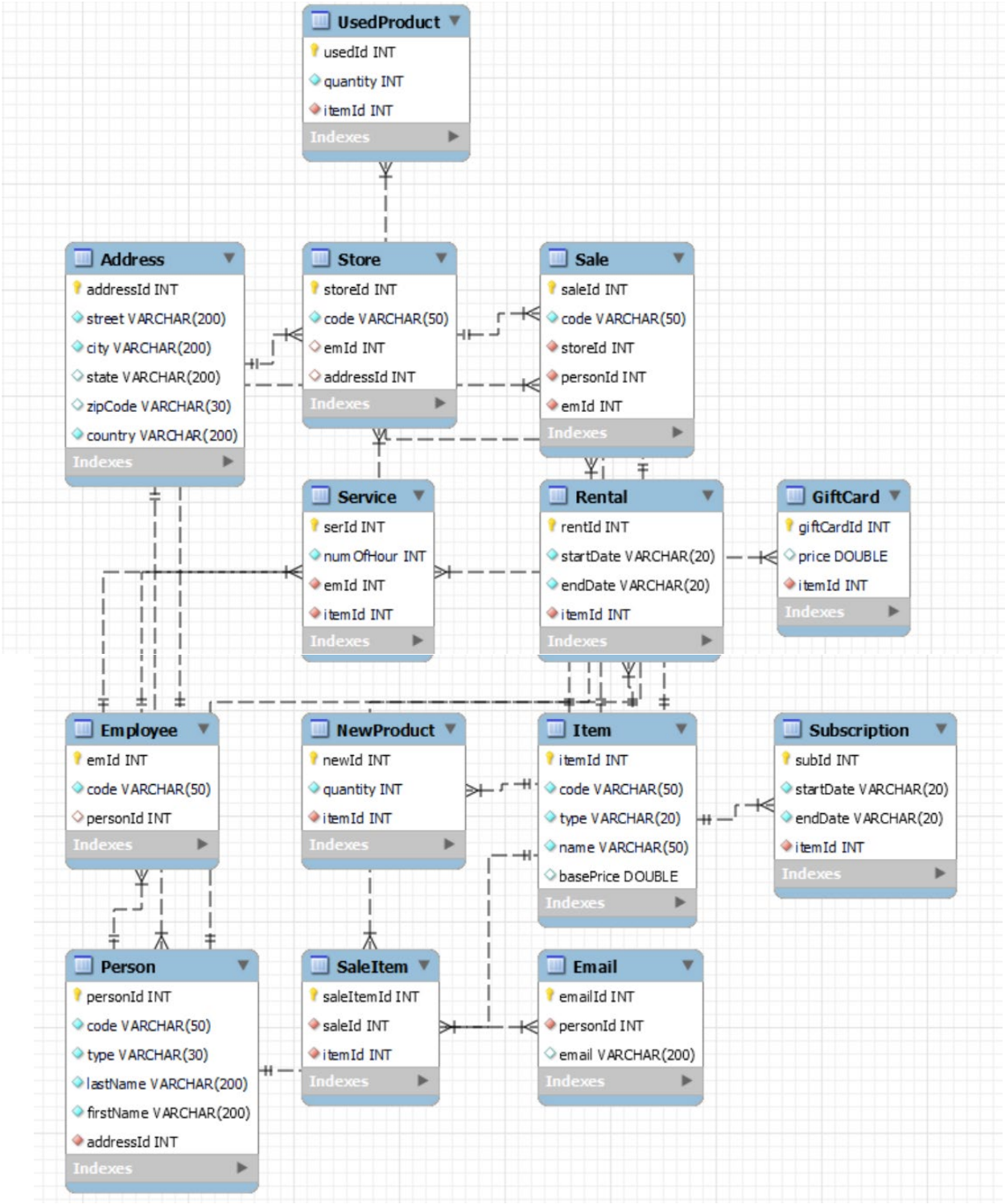
3.1 Database Design

This section needs to design a (MySQL) database to store data that models the problem in Phase II. This database will contain tables and fields which are the same with data in file csv from the previous phases. Decide how tables are related by foreign key constraints. Besides, database also includes an Entity -Relation diagram.

The Address table contains general information of addresses. Person table contains general information of people and connects with Address table to get addresses of relative people. Email table contains emails of people of Person table. Employee table is generated to contains all specific characteristics of employees of The Matrix stores by connecting to Person, Address, and Email table.

The Item table is created to include primary details of items of The Matrix stores. Rental, NewProduct, UsedProduct, Subscription, Service, GiftCard tables are made for specifying items in stores into different types with some specific fields, that's why they all have connection with Item table.

The Store table contains general information of stores, which are connected to Employee and Address tables. Sale table is used for including information about sales of stores, which are connected with Store, Person, and Employee tables. SaleItem table is built to contain information about items included in each sale, which are connected with Item, and Sale tables.


UsedProduct
usedId INT
quantity INT
itemId INT
Indexes
Address
addressId INT
street VARCHAR(200)
city VARCHAR(200)
state VARCHAR(200)
zipCode VARCHAR(30)
country VARCHAR(200)
Indexes
Store
storeId INT
code VARCHAR(50)
emId INT
addressId INT
Indexes
Sale
saleId INT
code VARCHAR(50)
storeId INT
personId INT
emId INT
Indexes
Service
serId INT
numOfHour INT
emId INT
itemId INT
Indexes
Rental
rentId INT
startDate VARCHAR(20)
endDate VARCHAR(20)
itemId INT
Indexes
GiftCard
giftCardId INT
price DOUBLE
itemId INT
Indexes
Employee
emId INT
code VARCHAR(50)
personId INT
Indexes
NewProduct
newId INT
quantity INT
itemId INT
Indexes
Item
itemId INT
code VARCHAR(50)
type VARCHAR(20)
name VARCHAR(50)
basePrice DOUBLE
Indexes
Subscription
subId INT
startDate VARCHAR(20)
endDate VARCHAR(20)
itemId INT
Indexes
Person
personId INT
code VARCHAR(50)
type VARCHAR(30)
lastName VARCHAR(200)
firstName VARCHAR(200)
addressId INT
Indexes
SaleItem
saleItemId INT
saleId INT
itemId INT
Indexes
Email
emailId INT
personId INT
email VARCHAR(200)
Indexes

3.1.1 Component Testing Strategy

Using SQL queries to develop the tests for database. For example, using query “select p.lastName, p.firstName, e.email from Person p left join Email e on p.personId = e.personId” to check whether the person’s emails are right or not.

3.2 Class/Entity Model

The Item abstract class represents for all items included in The Matrix stores.

The Address super class represents for all addresses of people and stores in the real life.

The Person abstract class represents for all characteristic of customers, managers, and employees of The Matrix stores in real life.

The Store class represents for stores of The Matrix chains in real life.

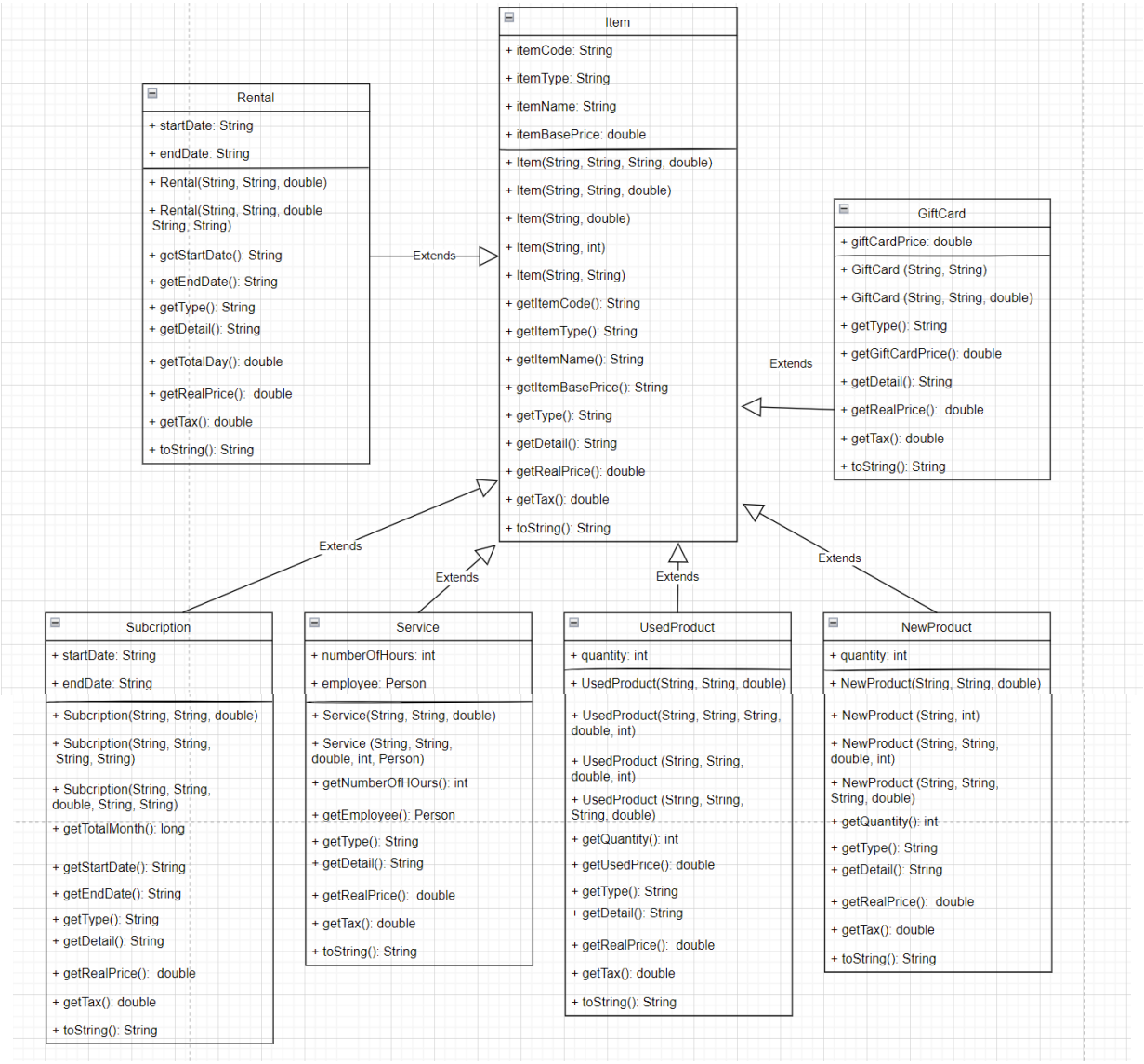


Figure 2: Item class

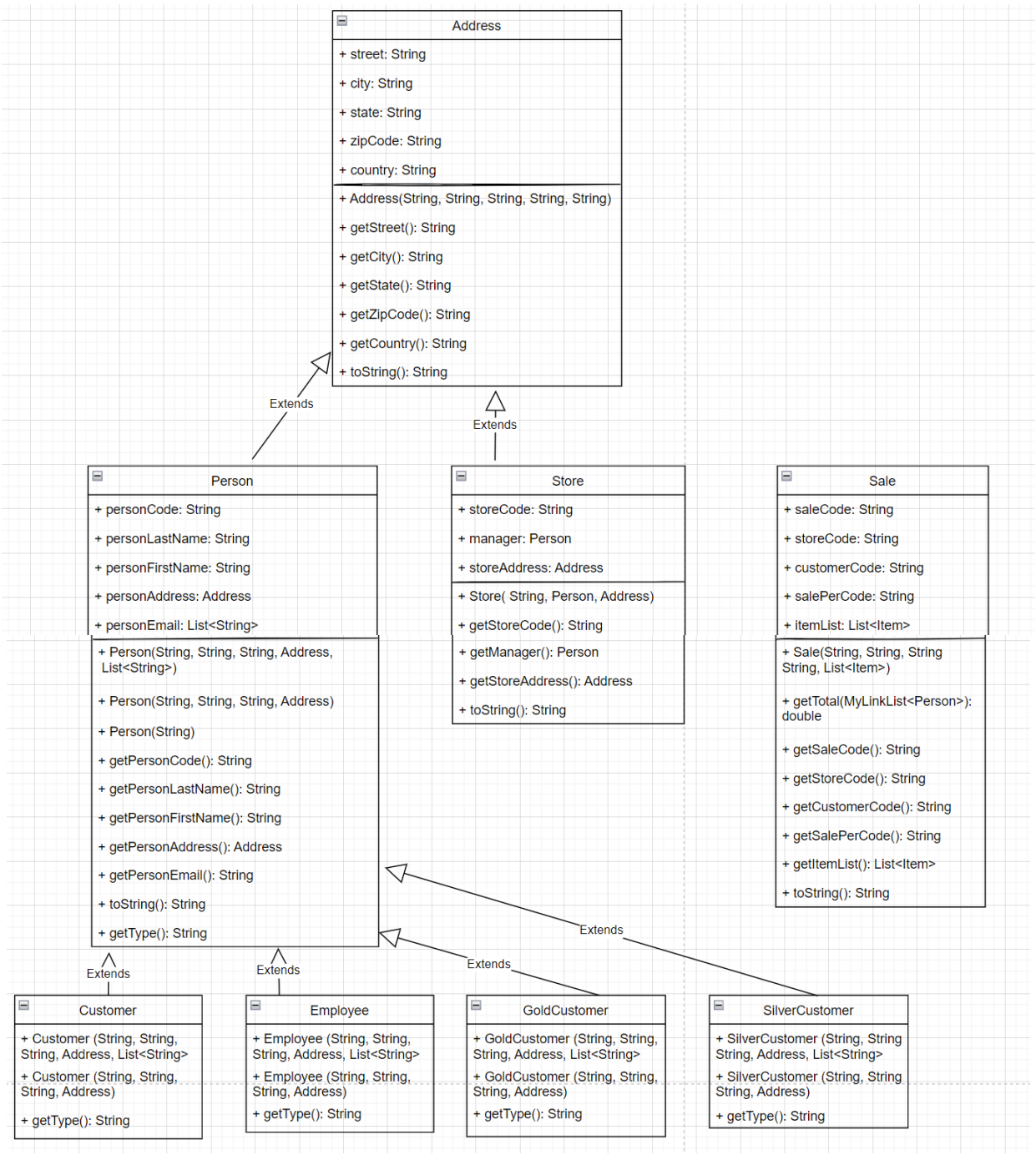


Figure 3: Address, Person, Store, Sale class

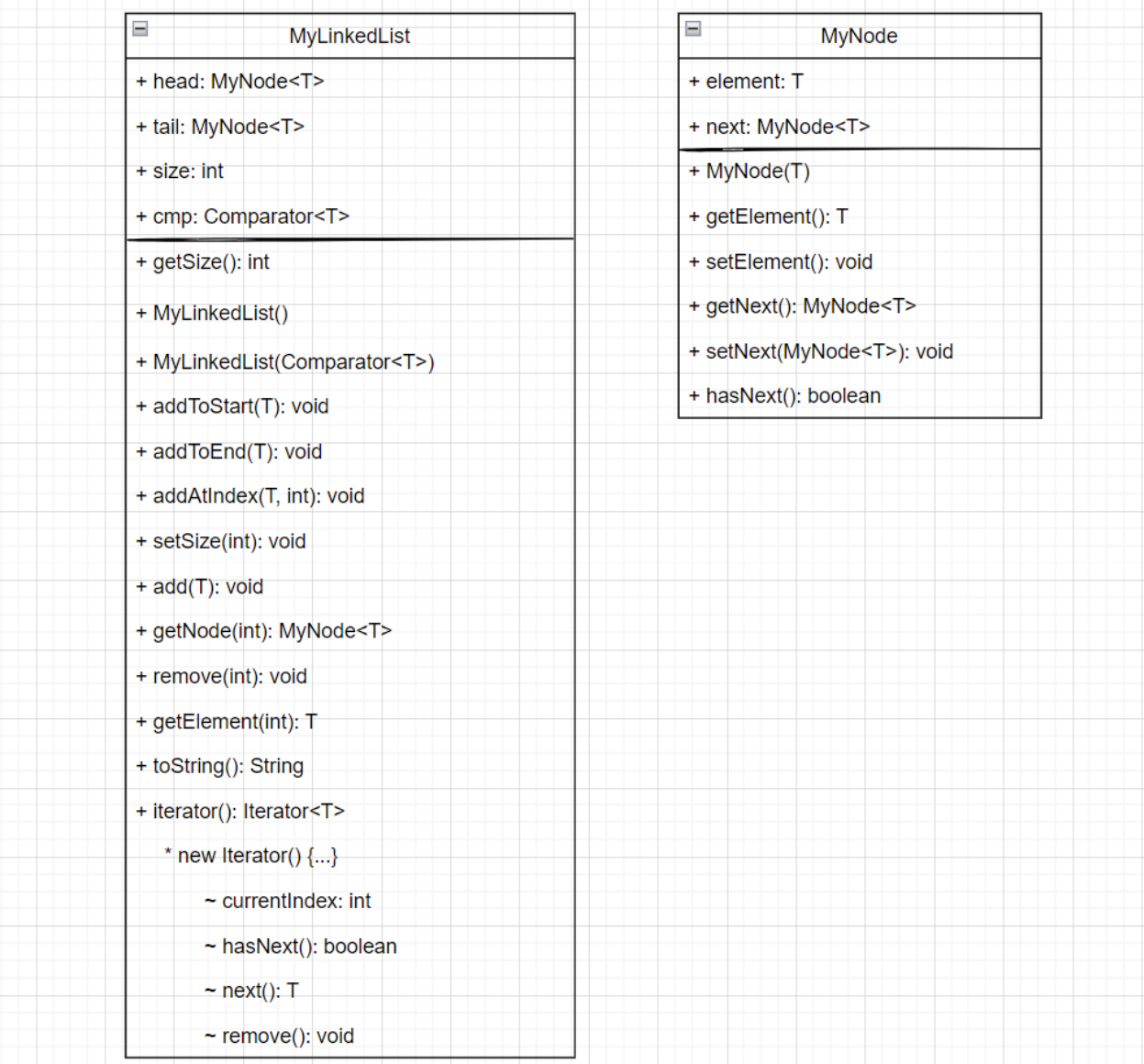


Figure 4: MyLinkList, MyNode Class

3.2.1 Component Testing Strategy

To test result from Java classes, create some specific testcases which are expected results and then compare it with result of the Java classes by JUnitTest.

Using Java JDBC API to access virtually to MySQL to read information from a database rather than from flat files. In this phase, Java code needs to be configured to connect and interact with 9 the database. Specifically, methods will be implemented to connect to database using JDBC, load the data into relative Java objects and produce the same summary and detailed reports.

3.3.1 Component Testing Strategy

To test how JDBC works, adding, removing, and modifying data to the database as needed. For example, adding one sale into database using JDBC and considering whether it is successful or not. If it succeeds, then reading information from database by Java JDBC is completely done.

3.4 Design & Integration of Data Structures

Using the API which has the driver class that generates the summary and detail reports will retain it functionality. In that class, a collection of methods to interact with database to persist and delete records. Moreover, designing, implementing, and incorporating a sorted list Abstract Data Type are to extend the functionality of the application. Specifically, designing and implementing a list class (or classes) are to hold an arbitrary number of sales objects which will maintain an ordering of each of the sales according to some criteria.

3.4.1 Component Testing Strategy

To test how the collection of methods work, using JUnit to modify test cases which compares the result of using methods and the result of using SQL queries.

3.5 Changes & Refactoring

When doing phase I, a general constructor had been used for all objects of Item class. If there were some values which did not belong to those types of items, they would be null. However, when doing phase II, there were many problems appeared that could not be fixed with that general constructor, so that more flexible constructors were created to avoid null values in constructors and to be able to solve other problems. Also, in phase I, configuring code to read information from flat files and change that information into XML had problems. The result in XML file was true, but it was not accepted by testcase due to lack of some packages of library. Therefore, instead of changing information into XML, JSON was used after that, and it was accepted.

4. Bibliography

J Jacqueem Content Writer I Technical writer focused on cybersecurity and musicianship. More Articles by Jacqueem, et al. “Create a Database Diagram in Mysql Workbench.” *InMotion Hosting Support Center*, 17 Aug. 2021, www.inmotionhosting.com/support/website/create-diagram-in-mysql-workbench/.

“Java: Implementing Iterator and Iterable Interface.” *GeeksforGeeks*, 17 July 2018, www.geeksforgeeks.org/java-implementing-iterator-and-iterable-interface/.