

Komar University of Science and Technology

College of Engineering

Subject: Term Report (Final Assessment)– Spring 2020

Department Name: Computer Science

Report Title: DVD Movies Rental Shop Application

Course Code: CPE2315

Course Name: Object Oriented Programming

Student ID: F180292

Student Name: Abdulrahman Tawffeq Jalal

Submission Date: 26-Jun-2020

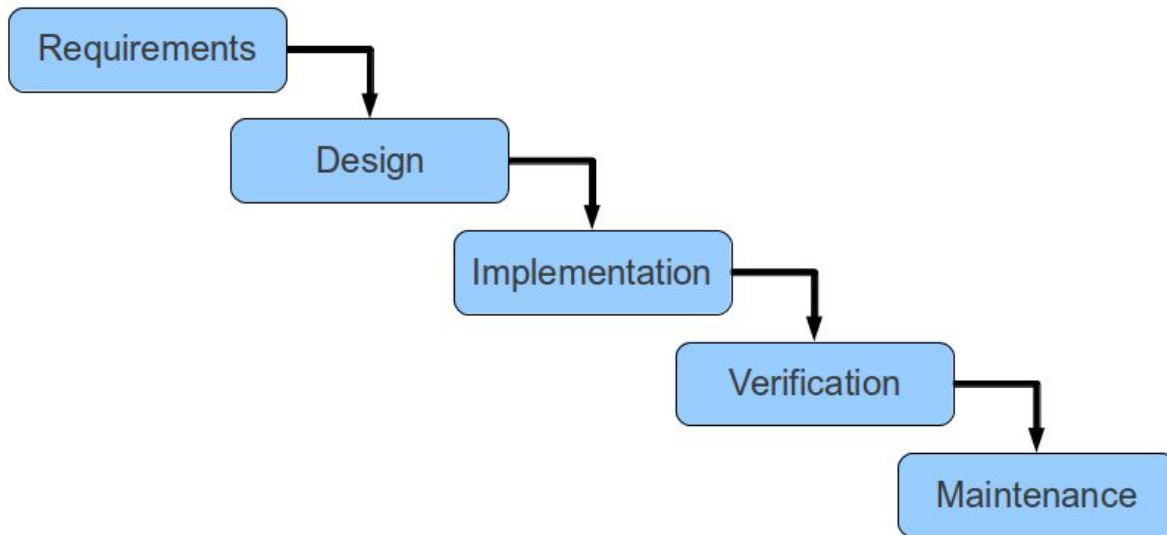
Introduction	3
Software Development Model	4
Requirements	4
Non-functional Requirements	4
Functional Requirements	5
The problem	5
Design	6
Class UML Diagram	6
Use Case Diagram	7
Person (Any User) Use Case	7
Manager Use Case	7
Employee Use Case	8
Customer Use Case	8
Database	9
A. MySQL DBMS	9
B. Tables	9
C. Data Dictionary	11
ER Diagram	12
Relationships	12
Implementation	13
Java Programming Language	13
Eclipse IDE	14
DVD Movies Rental Shop Application	14
Services Class	14
DBconnection Class	16
Properties	16
Methods	16
DataBaseCreation Class	17
Person Class	18
Signup	18
Session	18
Logout	18
Customer Class	19
Manager Class	20
Conclusion	20
References	21

Introduction

The science of design and programming is one of the most important and most popular sciences these days. Most businessmen go to programmers to request the provision of programs or electronic websites that serve their trade and business. And the best evidence for that was what was requested from us as a team to program an application for a movie rental store. We will draw a solid plan based on the requirements previously presented, make a specific design, and implement the work based on a specific strategy. After completing these steps, we will ensure that the program is working properly and thoroughly examining it. In the event of any error, we will repair it in the maintenance phase that is already in the work plan.

Software Development Model

The waterfall model



Requirements

Non-functional Requirements

1. The system must achieve the maximum benefit expected from it.
2. The operation performed by the user must be carried out at a high speed.
3. The rate of occurrence of errors and malfunctions in the system should be low.
4. The program should be with high efficiency in terms of space and performance.
5. The program should be easy to use.
6. The program must be secured enough.

Functional Requirements

Any user should be able to do:

1. Sign up, login, logout
2. Searching through the store selection of movies.

Manager user type should be able to:

1. Get a financial report.
2. Get a report of the top trending movies.
3. Hire, fire employees.
4. Get customers' issues and complaints. In the form of a report.
5. Add DVD movies to the database and Update movie details.

Employee user type should be able to:

1. Get a report of all the movies in store.
2. See all customers balances, and to update customers' balances.
3. Add DVD movies to the database and Update movie details.
4. Return DVD movies back from the customer.

Customer user type should be able to:

1. Rent movies.
2. Check his balance.
3. Give feedback about the employee performance.
4. See all his rentings in the form of a report.

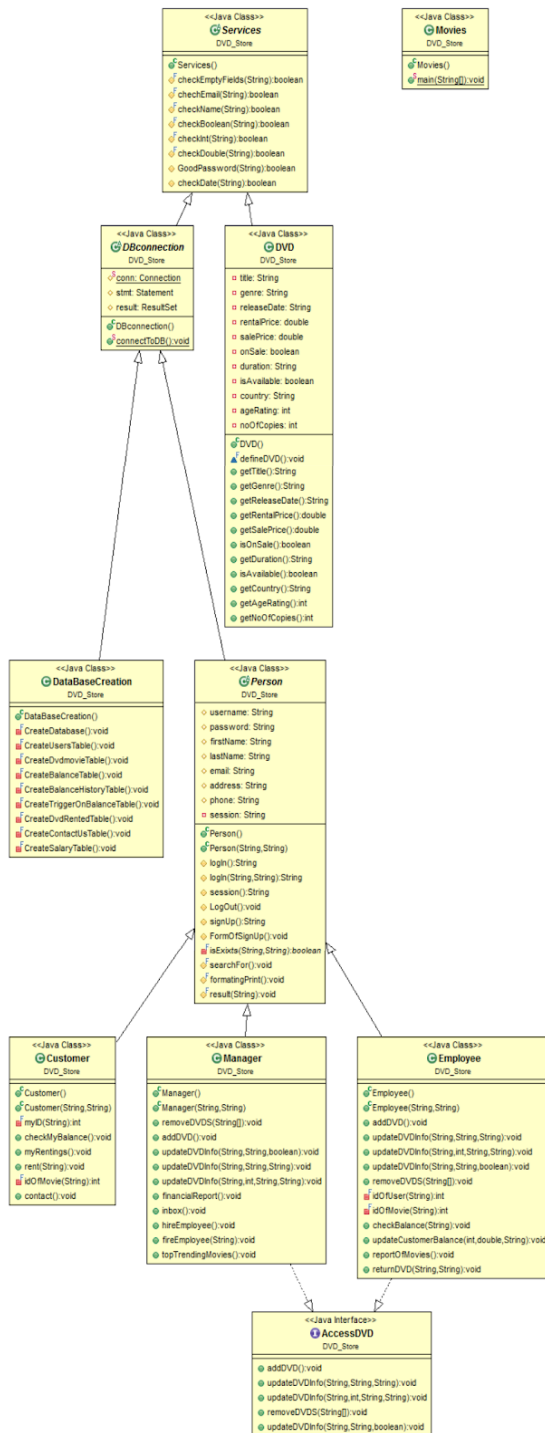
The problem

We need a system for a DVD movie rental shop that allows each user's type to do specific tasks. For example, it lets the Manager manage the program and be able to block or activate any user. As well as to let the Employee and Customer to do other different things in the program. That means each user will be able to do things and disable to do other things within the same program.

There are many other solutions for this problem out there, but those are too comprehensive that does not satisfy the current requirements. That's why we want to implement it ourselves to get a small program that can accomplish the task without any overhead. As well as, the program should be implementable and easy to use. That's why we decided to implement it to be easier than the others in terms of usability.

Design

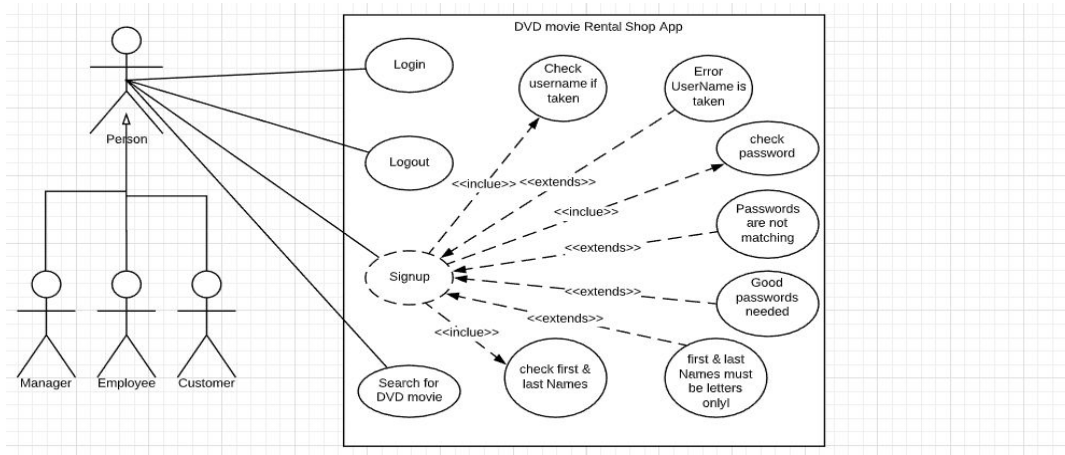
Class UML Diagram



Use Case Diagram

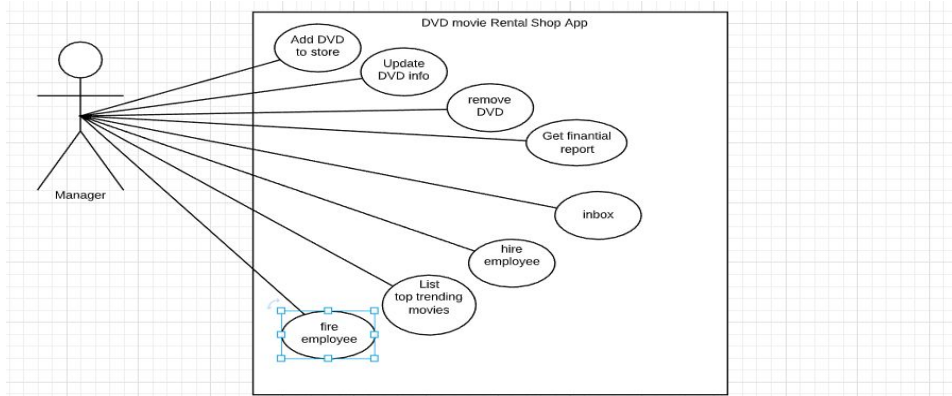
Person (Any User) Use Case

All Manager, Employee, and Customer classes are childs to the Person parent class. In this case, any user in the system will be able to login, logout, signup, and search for DVD movies in the application. In other words, those four methods are common, so that they are all in the parent class with protected visibility. While signing up, the user will select a username and during that the system will check if it is taken before or not (because it is a unique value). That's why the relationship is “include” because this check will be each time the new user tries to sign up. After the check, an error (Username is taken before) may occur. That's why this error extends this method as a relationship.



Manager Use Case

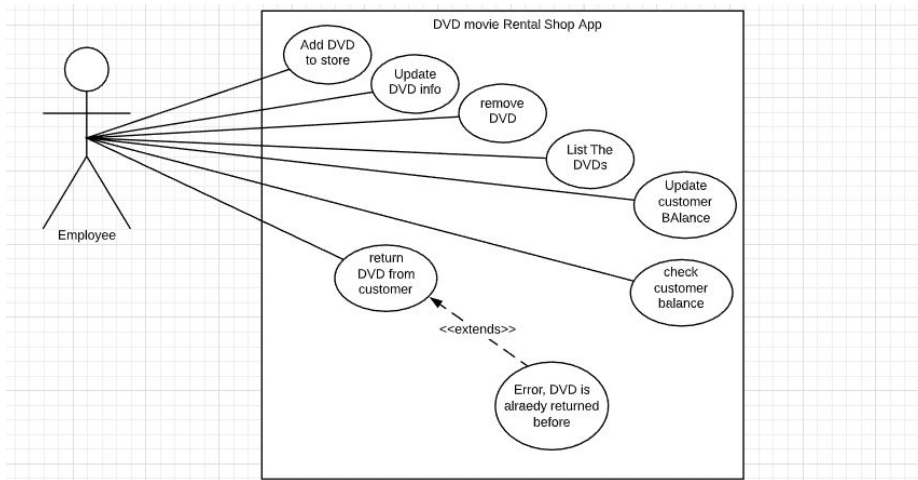
The manager will be able to do all these activities. He will be able to do the operations (with protected or public visibility) inside his parent (Person) class as well. The Manager can hire (sign employees up to the system).



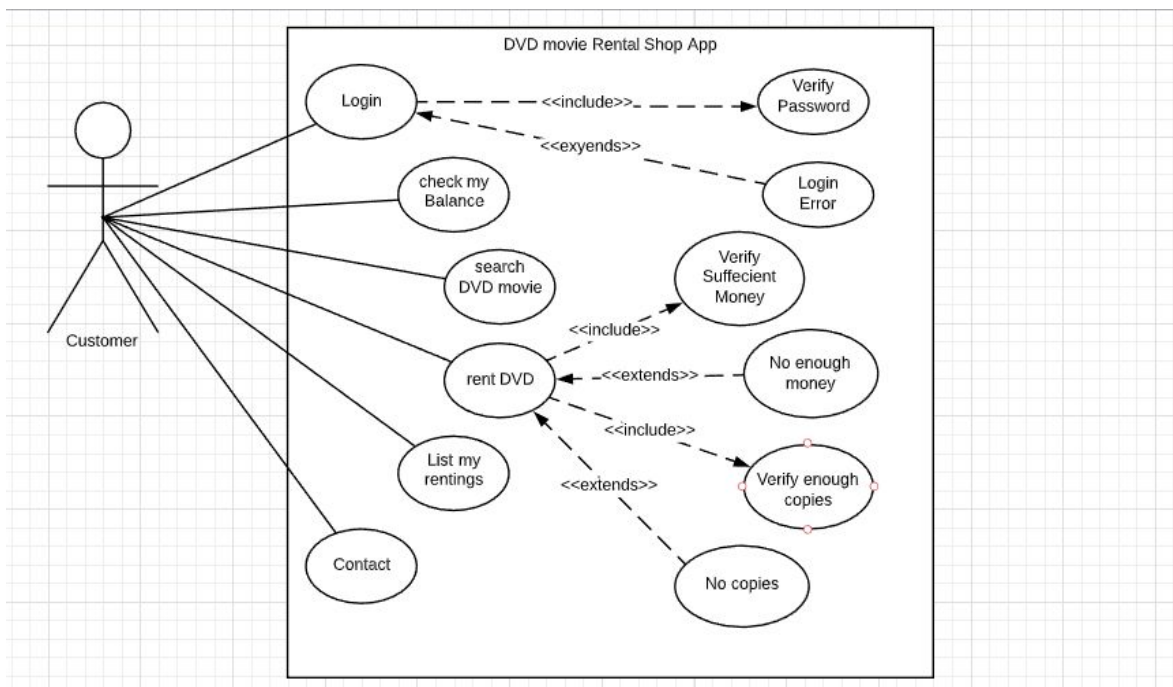
Employee Use Case

We notice that both employee and manager are implementing the same add, remove, and update DVD methods, that's because both are implementing the same AccessDVD interface.

When the customer returns a DVD, the employee will execute the returnDVD method to return the DVD from this specific user. If this DVD is already returned by this customer before and the status is returned in the dvdRented table, then an error message will appear. Because this message may be shown or not, the relation is extended to the return DVD operation.



Customer Use Case



Database

A. MySQL DBMS

The most important features of MySQL database systems are speed and reliability, which explains why they are frequently used by developers, administrators, and users around the world.

Here are some reasons why I chose this system to be used in the DVD Movie Rental Shop application.

1- Security in MySQL:

Security is a strong point in MySQL, as it comes with a complex access control system and an authorization system to prevent unauthorized users from accessing the database. This system is implemented in the form of five layers of powers hierarchically, as MySQL administrators can protect access to sensitive data. It can also allow users to perform operations on specific databases or only specific fields.

2- Ease of use MySQL:

Ease of use is an important point that MySQL has focused on. The MySQL development team has taken on the task of facilitating the use, management and improvement of MySQL performance. The main interface of MySQL is a simple inline interface, which consists of two graphical interfaces which are MySQL Control Center and MySQL Administrator, which were developed by MySQL AB to use and manage MySQL.

3- Support various programming languages:

MySQL provides a programming interface for various programming languages to enable you to write database applications in the language of your choice. It supports PHP, Java, c ++, Perl, Python, Tcl and others to give developers maximum freedom in designing applications that rely on MySQL.

B. Tables

Table Name	Fields' Names
Users	(<u>iduser</u> , username, password, firstname, lastname, email, address, phone, type, disactivated)
Dvdmovie	(<u>iddvd</u> , title, genre, releasedate, rentprice, onsale, saleprice, duration, noOfTimesRented, isAvailable, country, agerating, copies)
Dvdrented	(<u>id_rented</u> , idCustomer, iddvd, status, timeRent, timeReturn, rentPrice)
Salary	(<u>idSalary</u> , idEmployee, bounce, Salary)
Balance	(<u>idbalance</u> , customerID, recietNo, atime, amount)
Contactus	(<u>idContact</u> , from, to, topic, message, atime, status)

1. Users table: The user's information will be saved in this table once he or she signs up. The sign up form will ask the user for specific information related to him like his full name, username, password, address, etc.

Note: Username, email, and phone number are unique values here. The disactivated column will be zero by default and once the admin makes it one, the user will be disactivated.

iduser	username	password	firstname	lastname	email	address	phone	type	disactivated
1	Sara	1111aaaa	Sara	Muhammed	sm@gmail.com	Sulaimaniyah	07711116557	Customer	0
2	ADMIN	1111aaaa	Ali	Omar	ao@gmail.com	Sulaimaniyah	07711114545	Manager	0
3	Khalid	1111aaaa	Khalid	Omar	ko@gmail.com	Sulaimaniyah	0771111451	Employee	0
4	Omar	1111aaaa	Omar	Muhammed	oaaa@gmail.com	Suli	07711114747	Employee	0

2. Dvdmovie table: This table will contain all the information related to the DVD movie that the employee will define when he or she wants to add DVD movie to the database.

Note: The title of the movie is a unique value here and the number of times this dvd rented will be zero by default once the DVD is being added.

iddvd	title	genre	releasedate	rentprice	onsale	saleprice	duration	noOfTimesRented	isAvailable	country	agerating	copies
1	Superwoman	Action	2009	5	true	10	02:00	10	true	America	12	11
2	Spiderman	Action	2012	6	true	12	02:45	40	true	German	12	3

3. Dvdrented table: Once the customer rents a DVD movie, his id with the id of the DVD movie will be saved here as foreign keys. The date of the renting will be saved as well with the status which will be "currently rented". Once the customer returns the DVD back, the status will be returned and the return date will be saved as well.

id_rented	idCustomer	iddvd	status	timeRent	timeReturn	rentPrice
1	1	2	returned	2020-06-11 22:04:37	2020-06-11 23:12:15	6
2	1	2	returned	2020-06-11 22:06:29	2020-06-11 23:12:15	6

4. Balance table: Once the customer signs up to the system, his id will be saved in this table as foreign key with zero amount of money. If the customer contacts the employee to deposit some amount of money, his balance will be updated according to his id and the receipt number and date of deposit will be saved as well.

idbalance	customerID	recietNo	atime	amount
1	1	1564	2020-06-12 01:04:25	58

Note: A trigger is created for this table to save the old values once any changes done in this table. If the customer deposits, withdrawal, or rents, his balance will change and the old value will be saved in balancehistory table because it is a sensitive data (It is read only).

idbalance	customerID	recietNo	atime	amount
1	1	0	2020-06-11 18:41:55	0
1	1	12345	2020-06-11 18:43:34	45
1	1	0	2020-06-11 18:59:19	39

5. Salary table: The id of the employee will be saved here as foreign key with his salary and bounce if any.

idSalary	idEmployee	bounce	Salary
1	3	15	300
2	4	25	300

6. Contactus table: The customer is able to contact the manager directly in the system to ask any questions or to provide any feedback. On the other hand, the manager have an inbox to see all those messages from the customer. So, this table will save all the information of the messages and the date will be saved as well.

Note: The message will have an unread status by default till the manager marks it as read. Then, it will be erased from the inbox of the manager.

idContact	from	to	topic	message	atime	status
1	Sara	Manager	Problem	Hello, How are you? I have a problem with the..	2020-06-12 01:04:48	unread

C. Data Dictionary

Users Table

Column Name	Datatype
iduser	INT(11)
username	VARCHAR(45)
password	VARCHAR(255)
firstname	VARCHAR(45)
lastname	VARCHAR(45)
email	VARCHAR(45)
address	VARCHAR(45)
phone	VARCHAR(20)
type	VARCHAR(20)
disactivated	TINYINT(1)

DvdMovie Table

Column Name	Datatype
iddvd	INT(11)
title	VARCHAR(45)
genre	VARCHAR(45)
releasedate	VARCHAR(20)
rentprice	DOUBLE
onsale	VARCHAR(5)
saleprice	DOUBLE
duration	VARCHAR(20)
noOfTimesRented	INT(11)
isAvailable	VARCHAR(5)
country	VARCHAR(45)

DvdRented Table

Column Name	Datatype
id_rented	INT(11)
idCustomer	INT(11)
iddvd	INT(11)
status	VARCHAR(45)
timeRent	DATETIME
timeReturn	DATETIME
rentPrice	DOUBLE

Contactus Table

Column Name	Datatype
idContact	INT(11)
from	VARCHAR(45)
to	VARCHAR(10)
topic	VARCHAR(45)
message	VARCHAR(1000)
atime	DATETIME
status	VARCHAR(10)

Salary Table

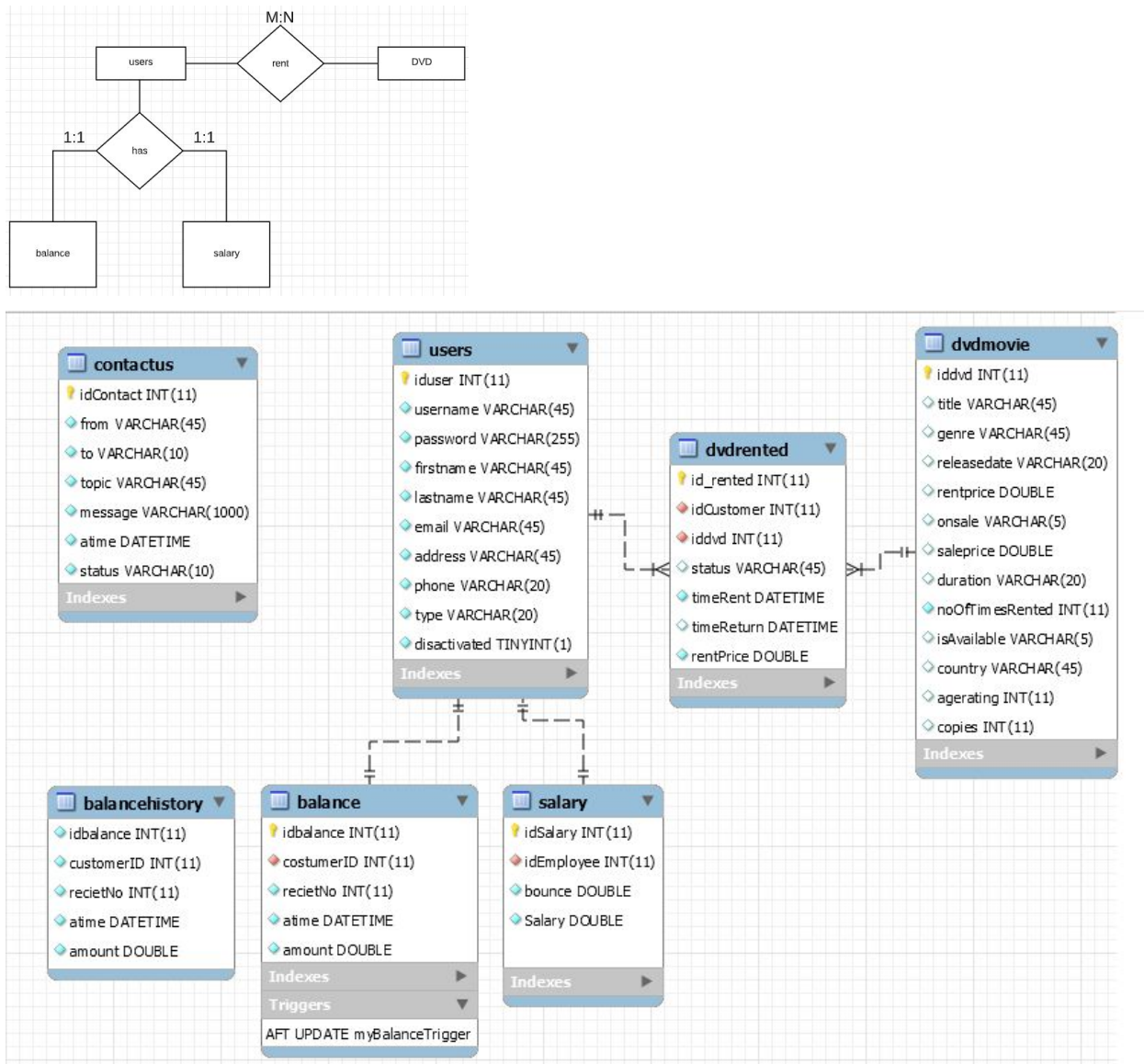
Column Name	Datatype
idSalary	INT(11)
idEmployee	INT(11)
bounce	DOUBLE
Salary	DOUBLE

Balance Table

Column Name	Datatype
idbalance	INT(11)
customerID	INT(11)
recietNo	INT(11)
atime	DATETIME
amount	DOUBLE

ER Diagram

ER diagram to database schema:



Relationships

1- Many to Many Relationship (M:N): This relationship is between the users and dvdmovie tables. The user will be able to rent several DVDs and the DVD will be rented by several users.

Note: The dvdrented will be created automatically by giving M:N relationship between those two tables. Their primary keys will be as foreign keys in this intermediate table

2- One to One Relationship (1:1): This relationship will be between the users and balance or salary tables. Each user (customer) will have one balance and the balance in that table will be related to a specific customer. The same thing with the salary table which is related to the employee.

Implementation

Java Programming Language

Java is a high-level, object-oriented and versatile programming language. A high-level programming language where it is characterized by its ease and use of understandable English terms and hides a lot of details to deal with computer hardware unlike the low-level languages.

Object-oriented (OO) programming language as everything is represented as an object, and each object has a specific type, attributes and actions that set it apart from others.

General-purpose programming language as it is used to make software in various fields, including Desktop applications, web applications, mobile applications, embedded systems, and other limited hardware devices.

Java is also a software platform that contains a Java Virtual Machine (JVM) that runs Java programs on it, regardless of the operating system (OS) or the type of CPU architecture that runs this machine. Java code is usually produced by the Java compiler which translates the Java language into the Java byte code that the virtual Java machine understands. There are also many programming languages that can be translated into Java bytecode, called JVM languages, and the most famous of them The language of Scala.

Because of the nature of the Java virtual machine in that it runs the same Java byte code on any operating environment, the Java language is characterized by the principle of "Write a Java program only once, and it will work on all operating systems", unlike other languages such as C / C ++ where it needs to write a different program for each operating system.

Java is the most widely used programming language in the world. According to Oracle, "there are three billion devices that run on Java". The TIOBE Programming Community index is updated monthly and, as usual, Java is the most popular programming language. That's why I used java to build the DVD movie rental shop application.

Eclipse IDE

We used Eclipse IDE because it has a lot of advantages. Eclipse IDE is an integrated pivotal environment for the development and modernization of IDE software, written in the "Java" language free of charge. It also remains under an open source license just like Android.

Although Eclipse is free and open source, it is full of documents that enable the programmer and help him to develop software and solve problems and difficulties that the programmer may encounter. As well as, Eclipse can work with the same degree of efficiency on various systems and devices.

Eclipse also deals with PLUGIN additions in a large and wide way where you can, as a programmer or as a user, program your additions and mix them in the Eclipse environment, then you can start programming your applications and programs.

One of the most important distinctions of the Eclipse program is that it is not large in size, as is the rest of the IDEs, as it can run smoothly and quickly even if the computer specifications are weak.

DVD Movies Rental Shop Application

Services Class

I created this class and it is at the top of the hierarchy. It is an abstract class so that it is not going to be instantiated. It is extended by DVD and DBconnection classes. So, it is a super class for all the classes directly or indirectly.

All the methods in this class are boolean and they are in protected visibility.

The methods:

1. checkEmptyFields (checks any input from the user, if it is empty or not)
2. chechEmail (checks any email if it is valid or not)
3. checkName (checks any name, does it contain letters only)
4. checkBoolean (checks if the input is true or false only, ignoring the case)
5. checkInt (checks the value if it is int or not)
6. checkDouble (checks the String if it is valid in case be parsedDouble)
7. GoodPassword (method to check if the password contains at least letters and numbers)
8. checkDate (This Method is to check the format of the entered date if using the correct formatting date. it accepts string type, the output is boolean type. Accepting dates formats eg. 2020-01-23).

Example for the checkEmptyFields method and where it is being used

```
protected final boolean checkEmptyFields(String input) {  
    boolean check = false;  
    if (input.trim().equals("")) {  
        check = true;  
    } else {  
        check = false;  
    }  
    return check;  
}
```

First In the sign up method in Person class:

```
System.out.println("Address: ");  
this.address = scan.nextLine();  
while (checkEmptyFields(this.address)) {  
    System.out.println("Fill out the empty field!");  
    System.out.println("Address: ");  
    this.address = scan.nextLine();  
}
```

Second in defineDVD method in DVD class:

```
System.out.println("Title: ");  
String title = scan.nextLine();  
while (checkEmptyFields(title)) {  
    System.out.println("Title: ");  
    title = scan.nextLine();  
}  
this.title = title;
```

Third in the Login method:

```
if (checkEmptyFields(userName) && checkEmptyFields(password)) {  
    System.out.println("Login Failed: Fill out the username & password parameters!");  
} else if (checkEmptyFields(userName)) {  
    System.out.println("Login Failed: Fill out the username parameter!");  
} else if (checkEmptyFields(password)) {  
    System.out.println("Login Failed: Fill out the password parameter!");  
}
```

Note: The same thing will be for other methods. We can use these methods anywhere in any child class for this super class (Services). It is so useful to have all these services in a place then to call them when needed. That will decrease code duplication as well.

DBconnection Class

Abstract Class to get connection to the “javadb” database. It is a parent of Person Class, so it is an indirect parent for manager, customer, and employee classes. Because this class inherits from Services class, so manager, customer, and employee classes can also access the Services class methods and use them.

Properties

It has three protected properties. So, they can be used in any child class to create or execute statements.

```
public abstract class DBconnection extends Services {  
  
    protected static Connection conn;  
    protected Statement stmt;  
    protected ResultSet result;  
}
```

Examples of its uses in different classes:

```
public void inbox() {  
    String query = "SELECT * FROM contactus WHERE status = 'unread'";  
    try {  
        stmt = conn.createStatement();  
        result = stmt.executeQuery(query);  
        System.out.println("\n-----INBOX MESSAGES-----");  
        if (!result.next()) {  
            System.out.println("No Unread Messages");  
        } else {  
            result = stmt.executeQuery(query);  
            while (result.next()) {  
                System.out.println("Name: " + result.getString(1) + "  
                Address: " + result.getString(2) + "  
                Phone: " + result.getString(3) + "  
                Email: " + result.getString(4) + "  
                Status: " + result.getString(5));  
            }  
        }  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
    }  
}
```

```
public void removeDVDs(String... title) {  
    for (int i = 0; i < title.length; i++) {  
        try {  
            connectToDB();  
            String query = "DELETE FROM dvdmovie WHERE title = '" + title[i] + "'";  
            stmt = conn.createStatement();  
            if (stmt.executeUpdate(query) == 1) {  
                System.out.printf("%s have been removed\n", title[i]);  
            } else {  
                System.out.println("removing process failed");  
            }  
        } catch (Exception e) {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

```
public void signUp() {  
    try {  
        int deactivated = 0;  
        String query2 = "insert into users(username,password,firstname,lastname,email,address,phone) values ('" + this.username + "','" + this.password + "','" + this.firstName + "','" + this.lastName + "','" + this.email + "','" + this.address + "','" + this.phone + "','" + this.type + "')";  
        if (conn.prepareStatement(query2).executeUpdate() == 1) {  
            System.out.println("You are signed up successfully");  
            this.session = this.username;  
            System.out.println("Welcome " + this.session + "(" + Type + ")");  
        }  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
    }  
}
```

Methods

connectToDB() method: To connect to “javadb” database which is already created beforehand. It is protected so it can be called anywhere to connect to the database

DataBaseCreation Class

This class is for creating the database and the required tables directly from java. Once the constructor is instantiated, the database will be built. This class inherits the DBconnection class and uses its conn and stmt properties. this class uses connectToDB() method which is inside DBconnection to connect to “javadb” in order to create tables in javadb database.

By Running this command in the main class, the database and tables will be created.

`DataBaseCreation newOne = new DataBaseCreation();` Because this class has private methods to create the database and tables. All these methods are called in the constructor.

```
public DataBaseCreation() {  
    // TODO Auto-generated constructor stub  
    try {  
        Class.forName("com.mysql.cj.jdbc.Driver");  
        conn = DriverManager.getConnection(  
            "jdbc:mysql://localhost/?useUnicode  
            "root", "");  
        CreateDatabase();  
        CreateUsersTable();  
        CreateDvdmovieTable();  
        CreateBalanceTable();  
        CreateBalanceHistoryTable();  
        CreateDvdRentedTable();  
        CreateTriggerOnBalanceTable();  
        CreateContactUsTable();  
        CreateSalaryTable();  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
    }  
}
```

CreateDatabase method creates “javadb” database if it is not existed.

All other methods call connectToDB() method which is in the DBconnection class. Those methods are about creating tables and there should be a connection to a specific database to create a table. That’s why there is a call for the connectToDB method.

Examples:

```
private final void CreateDatabase() {  
    try {  
        stmt = conn.createStatement();  
        String query = "Create Database javadb";  
        stmt.executeUpdate(query);  
        System.out.println("javadb database Created Succ  
        stmt.close();  
        conn.close();  
    } catch (Exception e) {  
        if (e.getMessage().equals("Can't create databas  
            System.out.print("");  
        } else {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

```
private final void CreateUsersTable() {  
    try {  
        connectToDB();  
        String query = "CREATE TABLE `javadb`.`users  
        stmt = conn.createStatement();  
        stmt.executeUpdate(query);  
        System.out.println("users table created succ  
        stmt.close();  
        conn.close();  
    } catch (Exception e) {  
        if (e.getMessage().equals("Table 'users' alr  
            System.out.print("");  
        } else {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

Person Class

I created signUp, FormOfSignUp, isExists, session, and logout methods in this class.

Signup

It Uses connectToDB() method from the superclass to get connection to the database. Asking the user to type specific information to sign up himself uses several private methods in the same class and from Services class to check if the inputs from the user are valid or not.

- It uses connectToDB() method because there should be a connection to the database.
- It uses FormOfSignUp method (a sign up form to let the user fill out his info) and this method uses another private boolean helper method called "isExists" (checks the username, email and phone number that the user will input when signing up, is it taken or not, because all these values are unique.)
- It uses this command `String Type = this.getClass().getSimpleName();` to get the type of the user who is signing up, to be inserted into the users table as well. If the type is Employee, "Ask the Manager to hire you!" message will appear because only the manager can hire this type of user.

Once the user be signed up, two things will be done as well:

1. His balance will be inserted as zero in the balance table (related to the customers).
2. The session property in the same class "Person" will be equal to the current user's username who just signed up.

Session

Protected method

```
/**
 * catches the user name of the user once he or she login or sign up
 *
 * @return session private property which is null or contain the user name
 */
protected String session() {
    return this.session;
}
```

Logout

```
* makes the session equal to null, means the user should login again to be able
* to react with the program
*/
protected void Logout() {
    String name = this.session;
    this.session = null;
    String Type = this.getClass().getSimpleName();
    System.out.println(Type + " " + name + ", you are logged out.");
}
```

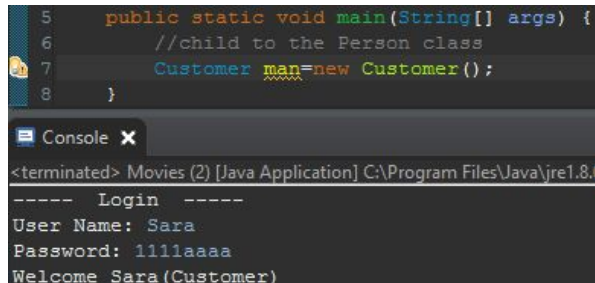
Constructors

The default constructor and constructor with parameters in the Person “Parent Class” have a logic to force the user to login or sign up to use the system because:

1. The login and sign up methods are called in Person’s constructors in a way.
2. All other children classes have constructors using super(), which means that they like to have the same constructors as Parent class.

So, by instantiating any child’s constructor, the system will ask for login or sign up automatically.

Example:



```
5 public static void main(String[] args) {
6     //child to the Person class
7     Customer man=new Customer();
8 }
```

Console X

```
<terminated> Movies (2) [Java Application] C:\Program Files\Java\jre1.8.
----- Login -----
User Name: Sara
Password: 1111aaaa
Welcome Sara(Customer)
```

Note: Sara now can search, rent, check her balance, etc because she logged in and inside this login method there is a call to connectToDb() method. So, once she logs in, there will be a connection to the database and she can interact with the system and get the results.

Customer Class

Contact

This method allows the Customer to be able to contact the Manager of the DVD Movie Rental Store and send him feedback, issues, or complaints.

This method uses prepared statements because here the user inserts data to the database and it is important to prevent the database by using this technology.

```
String query = "INSERT INTO contactus(`from`,`to`,`topic`,`message`) VALUES(?,?,?,?) ";
try {
    PreparedStatement ps = conn.prepareStatement(query);
    ps.setString(1, myUserName);
    ps.setString(2, "Manager");
    ps.setString(3, topic);
    ps.setString(4, message);
}
```

Manager Class

Inbox

This method is to let the Manager be able to read all the messages he received from the customers. He can also mark them as read to make the inbox empty.

This is done by executing this query:

`String query = "SELECT * FROM `contactus` WHERE status='unread' order by `atime` asc";` then by using a loop to get all the unread messages in the contactus table.

If the manager marked them as read, then the status will be changed in the table to be as read. For the second call, the inbox will be empty.

```
System.out.println("Mark them as read(Yes or No): ");
Scanner scan = new Scanner(System.in);
String value = scan.nextLine();
if (value.equalsIgnoreCase("Yes")) {
    String query2 = "UPDATE contactus SET status='read'";
    stmt = conn.createStatement();
    if (stmt.executeUpdate(query2) == 1) {
    }
}
```

Conclusion

Once the project is completed, examined, and verified that it is working properly and without errors, this does not mean that the programmer's work has ended. Most successful applications are periodically maintained. In the future, after using the program for a certain period by a large number of people, some errors may appear that need urgent maintenance. Other times, the owner of the program may ask to add some updates to it, in this case the same approach that was taken when building the program will be followed. The programmer analyzes, designs, then performs and checks. Personally, I strongly recommend that some updates should be made to the current program in this report. For example, the feature to send a notification to the user should be added in the event that the DVD movie is not returned after a specified time. I also recommend rewriting the method of updating movie information and making it in one method only to make it easier to use.

References

1. <https://www.techno-4u.com>
2. <https://laravel-ar.com/article>
3. <https://blog.naqrah.net/7-reasons-to-learn-java/57>
4. https://fouad.io/2017/01/java_introduction
5. <https://www.w3schools.com>
6. <https://stackoverflow.com>
7. Ramez Elmasri and Shamkant B Navathe. 2015. Fundamentals of database systems.
8. https://www.youtube.com/channel/UCs9QB-ccbNQDWvIQIu_RWlQ
9. <https://www.youtube.com/user/alxs1aa>