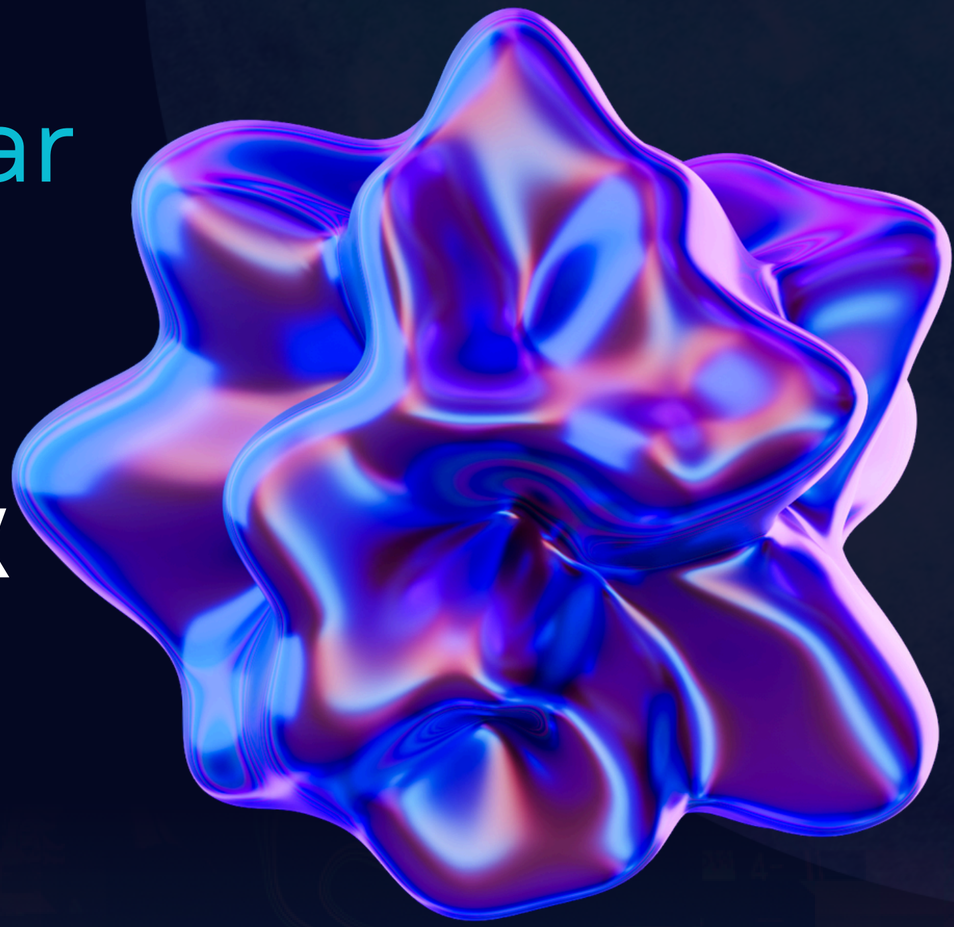
Presentasi Tugas Besar

IF-404 Rekayasa Perangkat Lunak

01
02
03



STEAM GAME

Our Team

1122001 - Nico Jonathan Setiawan

1122003 - Jonathan Marvel Sena

1122045 - Devid Laritsan Manurung

Tugas Besar RPL

Background Story

Pada Tugas Besar kali ini kami mengambil tema tentang Steam yaitu platform distribusi digital untuk permainan video yang dikembangkan oleh Valve Corporation.

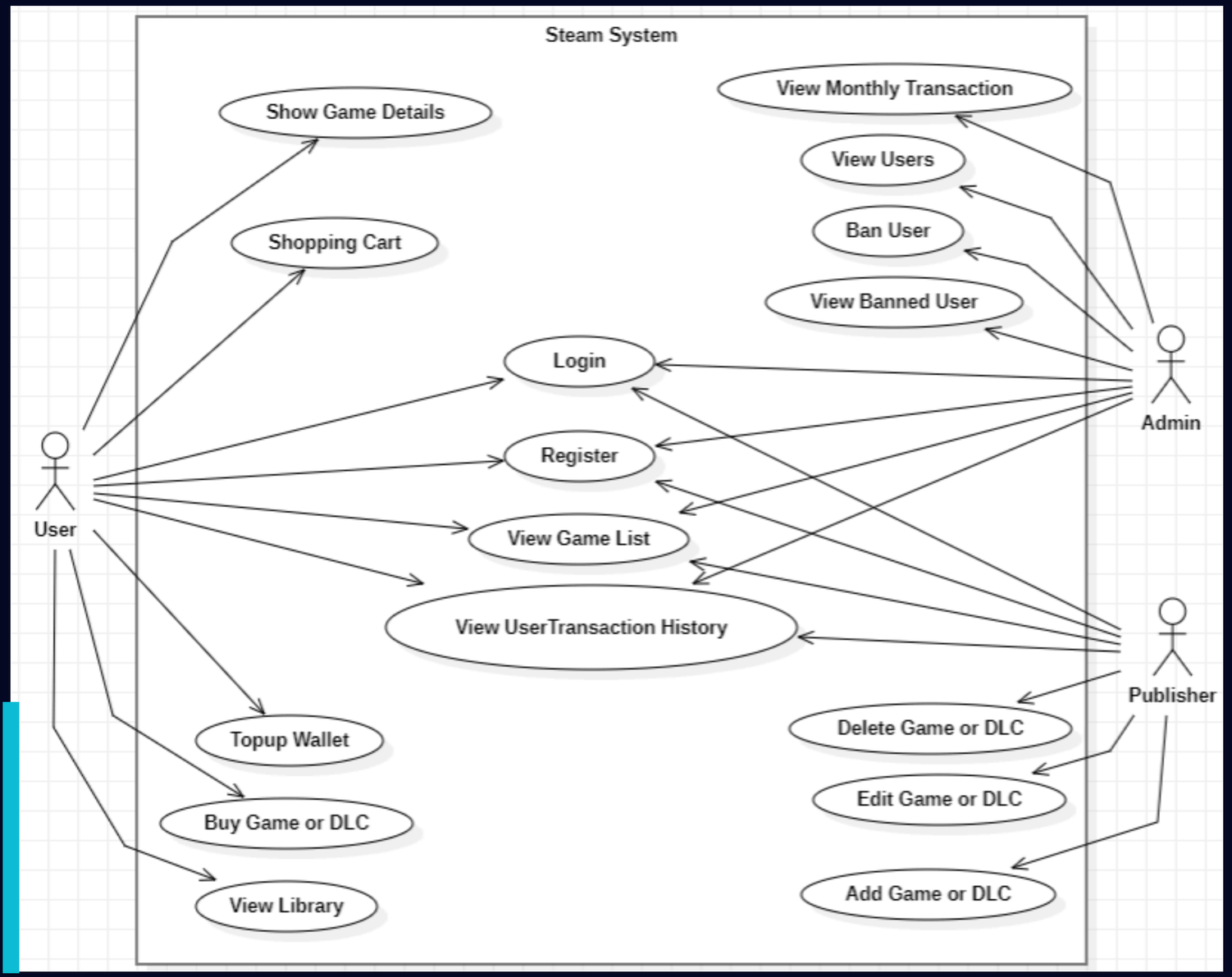
Alasan kami memilih tema Steam karena platform ini tidak hanya merupakan pemimpin dalam industri permainan digital, tetapi juga menawarkan beragam fitur yang relevan dengan konsep yang ingin kami kembangkan dalam Tugas Besar kami. Dengan menggunakan Steam sebagai inspirasi, kami dapat menerapkan Dokumentasi berupa Use Case dan Class Diagram dan juga prinsip-prinsip pembangunan Perangkat Lunak seperti Clean Code, Design Pattern dan SOLID Principles.

04
05
06

Software Documentation

05
06
07

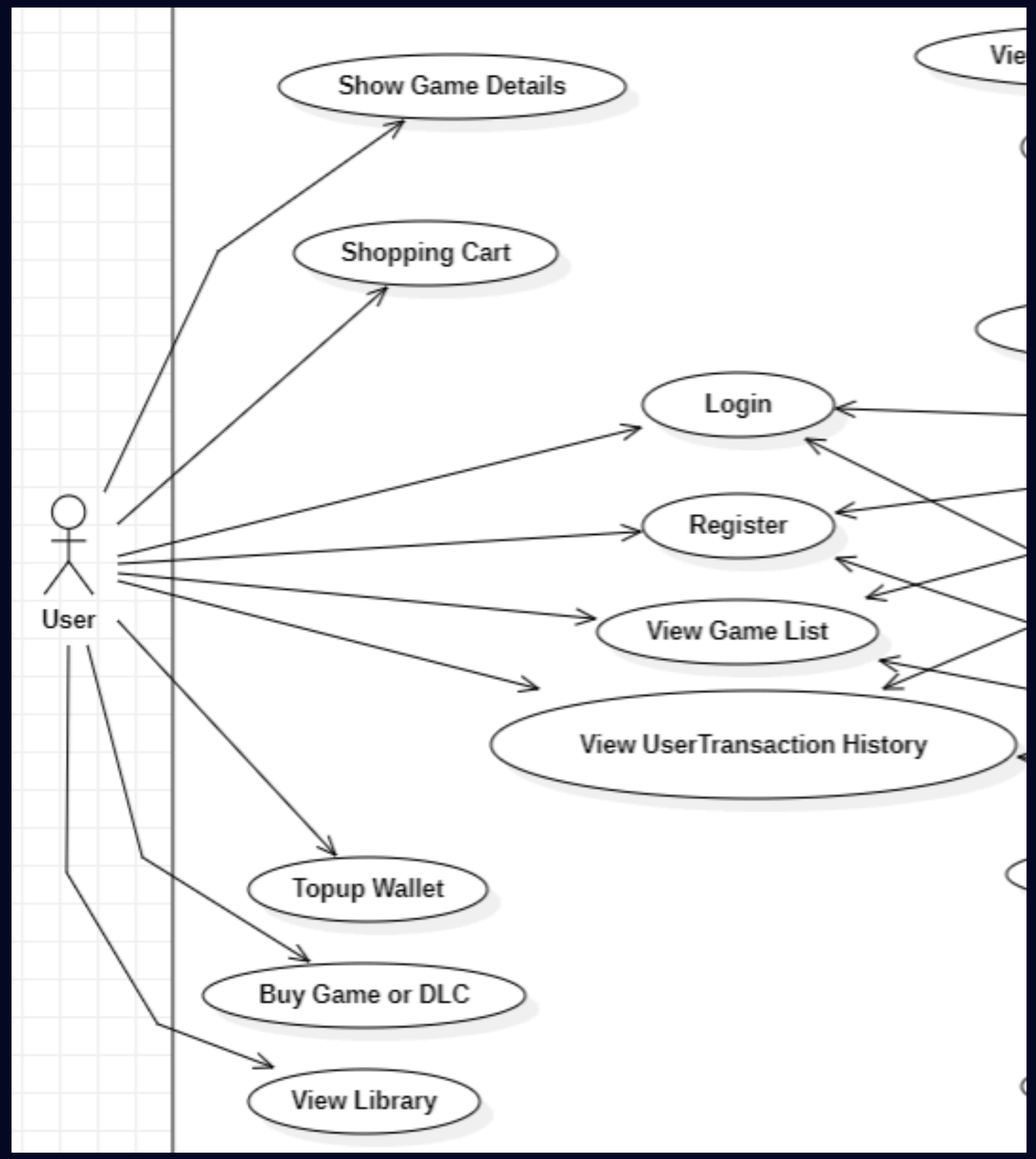
Use Case Diagram



Use Case Diagram: **User**

Menu User, antara lain:

1. Login
2. Register
3. View Game List
4. View User Transaction History
5. Shopping Cart
6. Top up Wallet
7. Buy Game or DLC
8. View Library



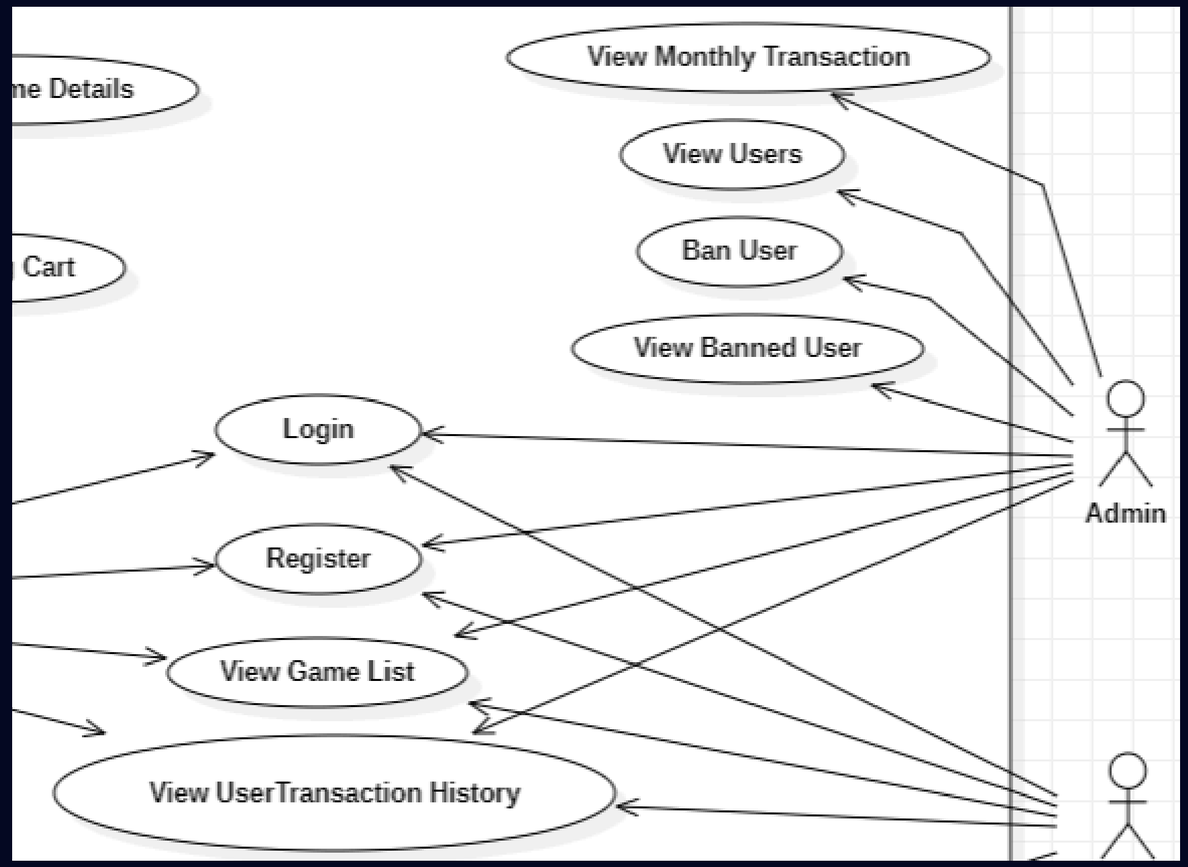
Use Case Diagram: **Admin**

07 **Menu Admin, antara lain:**

08

09

1. Login
2. Register
3. View Game List
4. View User Transaction History
5. View Monthly Transaction
6. View Users
7. Ban User
8. View Banned User



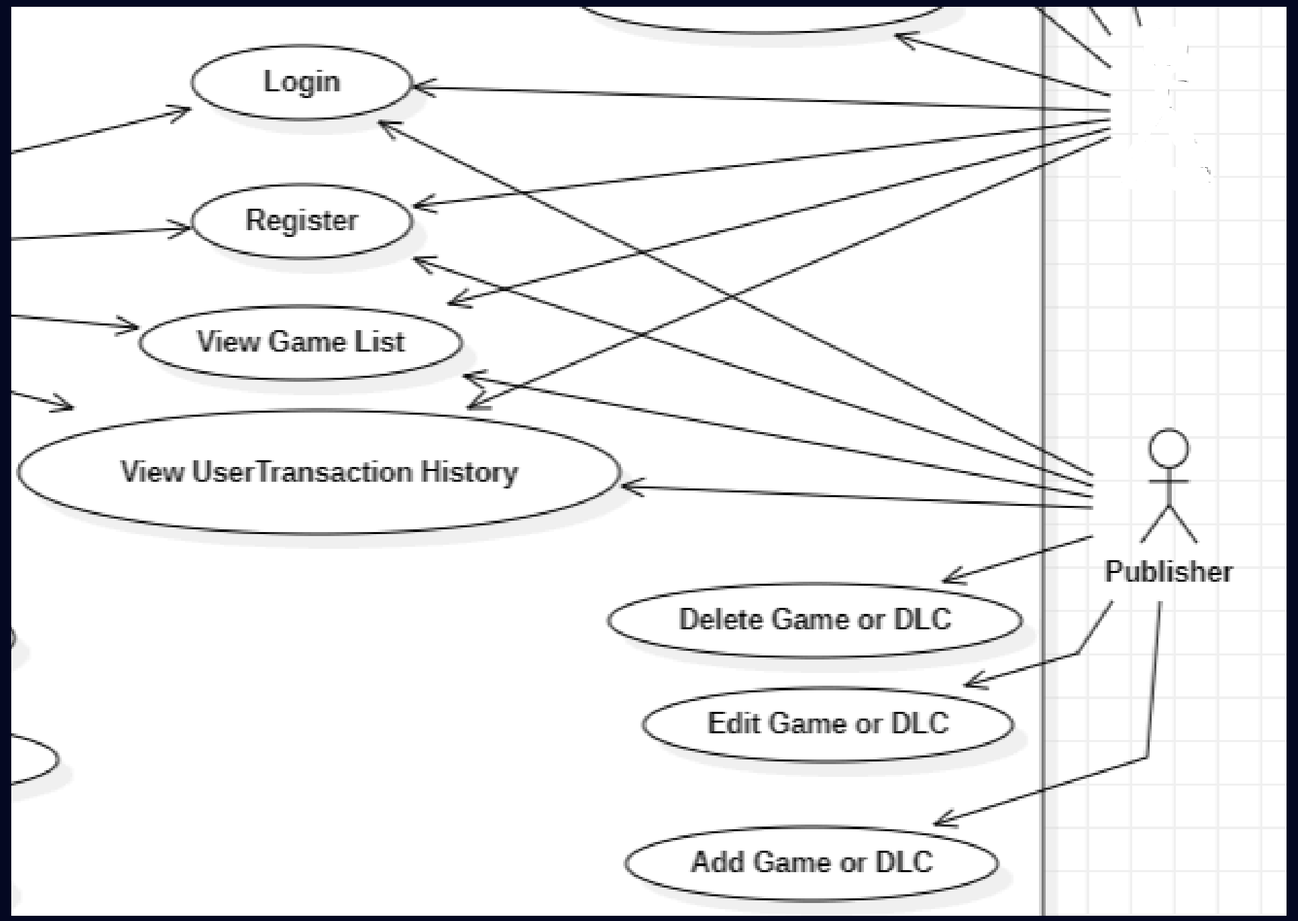
Use Case Diagram: **Publisher**

08 **Menu Publisher, antara lain:**

09

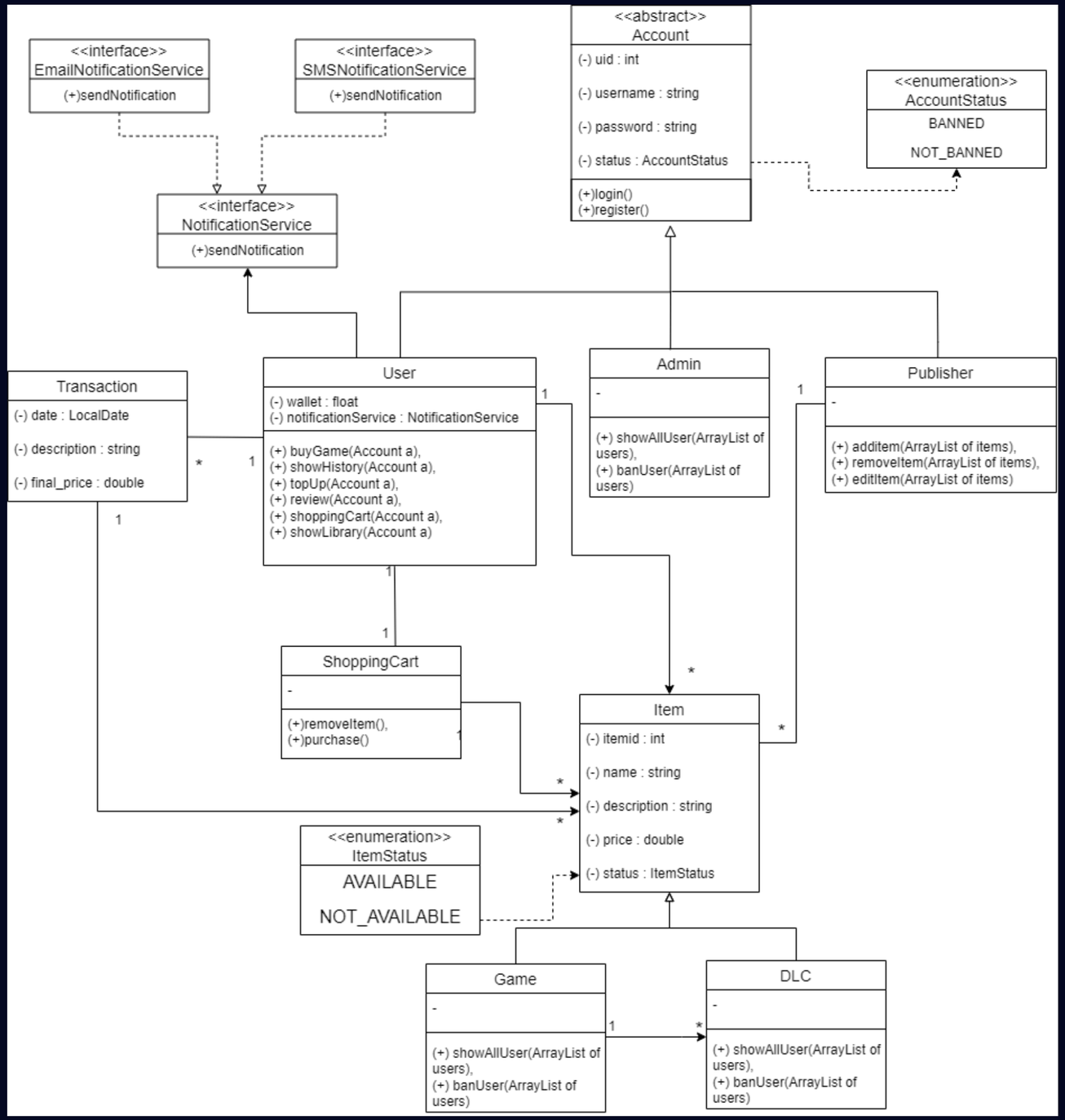
10

1. Login
2. Register
3. View Game List
4. View User Transaction History
5. Delete Game or DLC
6. Edit Game or DLC
7. Add Game or DLC

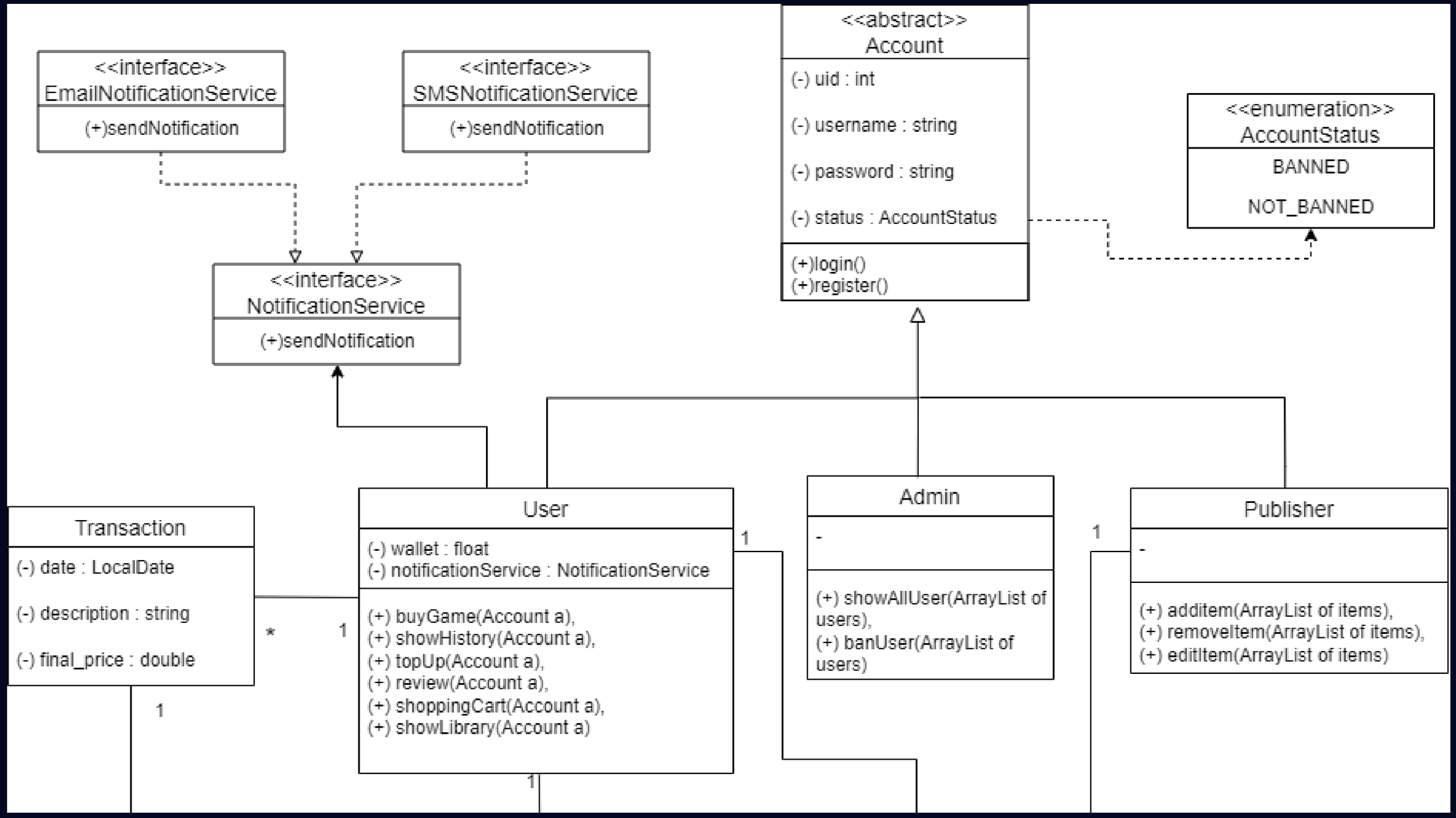


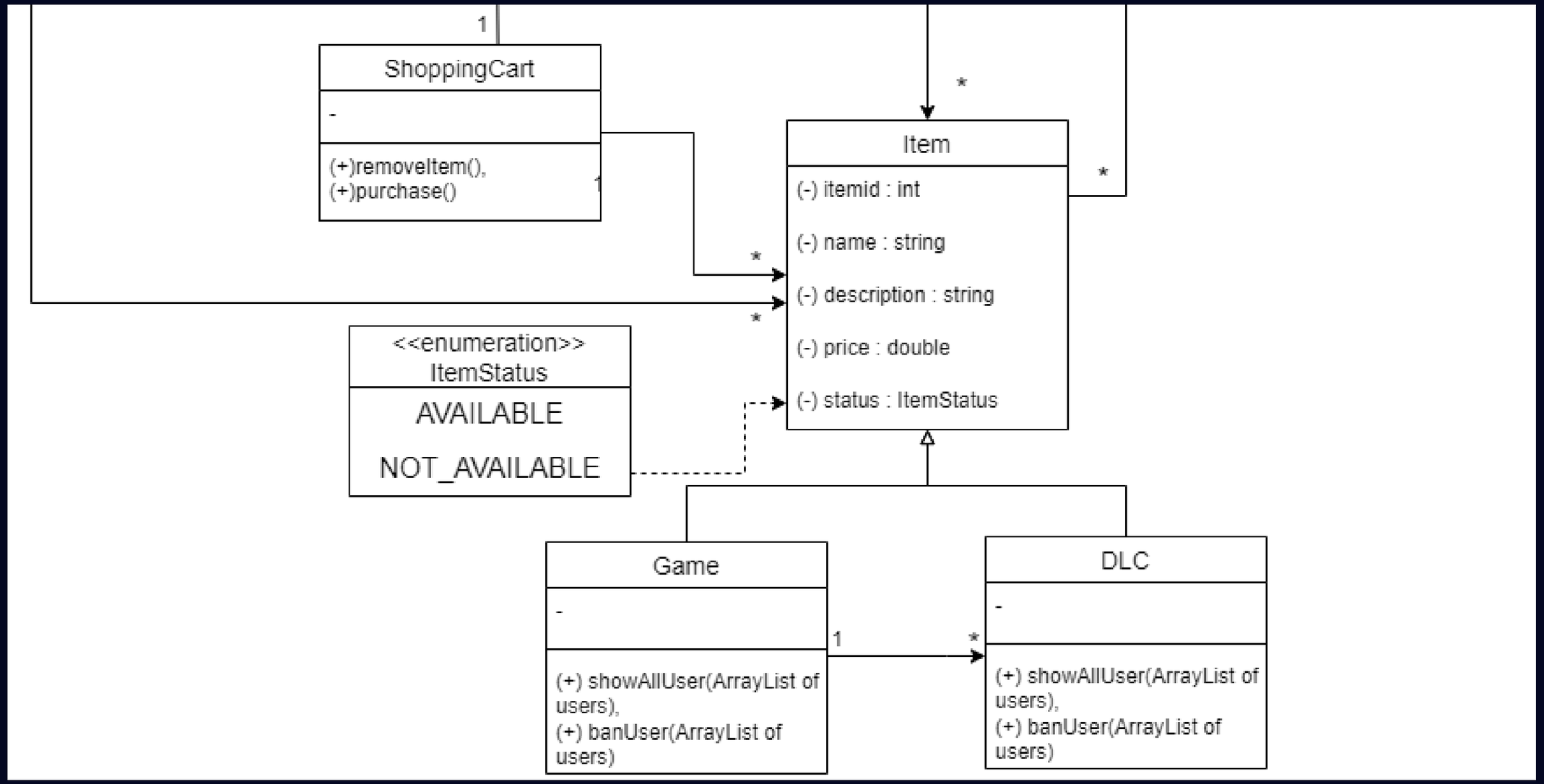
Class Diagram

09
10
11



Tugas Besar RPL


<<interface>>
EmailNotificationService
(+)sendNotification
<<interface>>
SMSNotificationService
(+)sendNotification
<<interface>>
NotificationService
(+)sendNotification
<<abstract>>
Account
(-) uid : int
(-) username : string
(-) password : string
(-) status : AccountStatus
(+)login()
(+)register()
<<enumeration>>
AccountStatus
BANNED
NOT_BANNED
Transaction
(-) date : LocalDate
(-) description : string
(-) final_price : double
User
(-) wallet : float
(-) notificationService : NotificationService
(+) buyGame(Account a),
(+) showHistory(Account a),
(+) topUp(Account a),
(+) review(Account a),
(+) shoppingCart(Account a),
(+) showLibrary(Account a)
Admin
-
(+) showAllUser(ArrayList of users),
(+) banUser(ArrayList of users)
Publisher
-
(+) additem(ArrayList of items),
(+) removeItem(ArrayList of items),
(+) editItem(ArrayList of items)
*
1
1
1
1
1
1


1
ShoppingCart
-
(+)removeItem(),
(+)purchase()
1
*
Item
(-) itemid : int
(-) name : string
(-) description : string
(-) price : double
(-) status : ItemStatus
*
*
*
<<enumeration>>
ItemStatus
AVAILABLE
NOT_AVAILABLE
Game
-
(+) showAllUser(ArrayList of users),
(+) banUser(ArrayList of users)
1
*
DLC
-
(+) showAllUser(ArrayList of users),
(+) banUser(ArrayList of users)

12
13
14

Software Development Principles

Clean Code

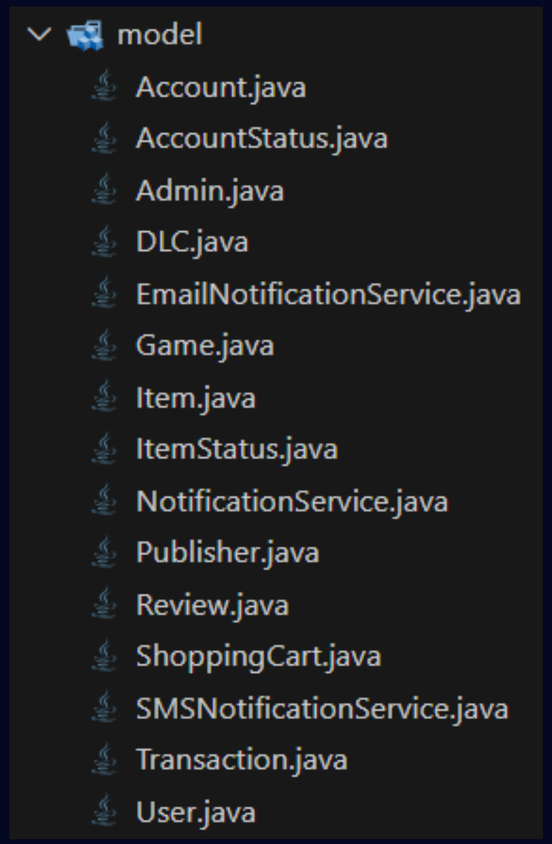
Meaningful Names:
Example (Function & Variable)

14

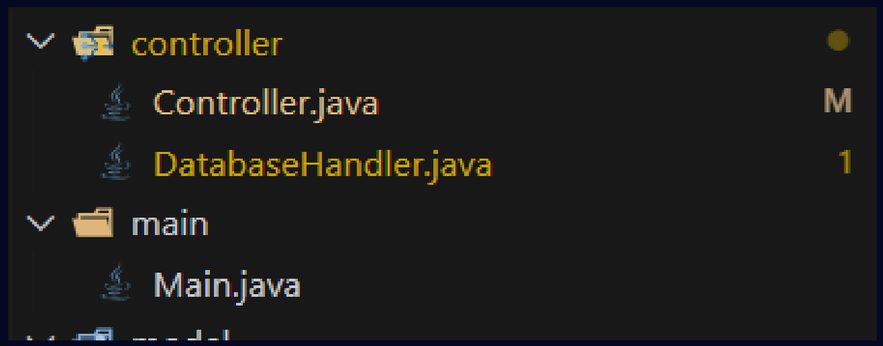
```
15
16
```

```
1  public EditGame(Publisher publisher, String type, int id) {
2      Item item = null;
3      for (Item v : con.getAvailableItems()) {
4          if (v.getItemID() == id) {
5              item = v;
6              break;
7          }
8      }
```

Meaningful Names : Classes



Nama Class kita menggunakan Noun (Kata Benda) mengikuti kaidah Clean Code

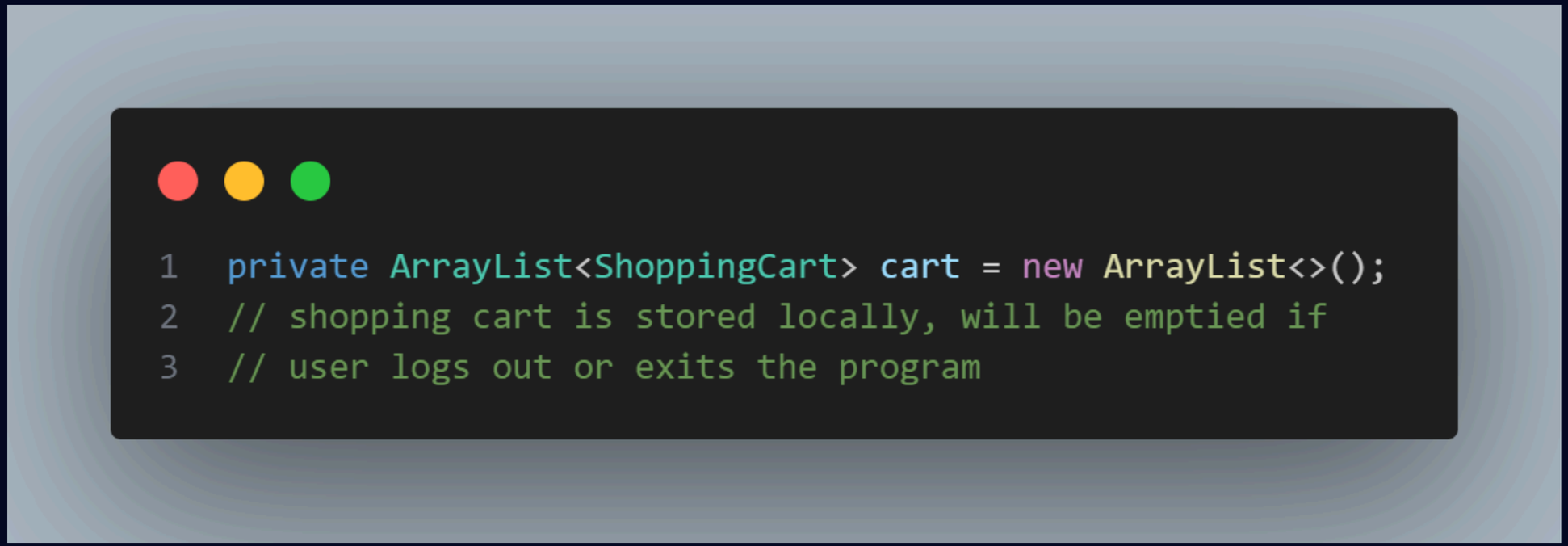


Meaningful Names : Functions

```
 44 >    public Account getUser(String username, String password) {···
110
111 >    public boolean insertNewUser(User user) {···
138
139 >    public boolean insertNewGame(Game game, Publisher publisher) {···
171
172 >    public boolean insertNewDLC(DLC dlc, Publisher publisher) {···
203
204 >    public boolean topUpWallet(User user, double topUpAmount) {···
220
221 >    public ArrayList<User> getAllUserList() {···
243
244 >    public boolean banUser(int id) {···
257
258 >    public ArrayList<User> getUnbannedUsers() {···
292
293 >    public ArrayList<User> getBannedUsers() {···
315
316 >    public ArrayList<Game> getGames() {···
358
359 >    public ArrayList<Game> getPublishedGames(Publisher publisher) {···
400
401 >    public ArrayList<DLC> getDLCs(Game game) {···
```

Nama Function pada Controller.java kita menggunakan Verb (Kata Kerja) mengikuti kaidah Clean Code

Comments



```
17
18
19
```

```java
private ArrayList<ShoppingCart> cart = new ArrayList<>();
// shopping cart is stored locally, will be emptied if
// user logs out or exits the program
```

Comments

18
19
20

```java
public class Item {
    private int itemID;
    private String name;
    private String type; // game or DLC
    private String description;
    private double price;
    private int publisherID;
    private ArrayList<Review> reviews;
    private ItemStatus status; // available or not available
```

Functions

```
621 >   public boolean insertIntoShoppingCart(User user, Item item) {···
653
654 >   public ArrayList<Transaction> getTransactions() {···
677
678 >   public boolean purchase(User user, ShoppingCart cart) {···
731
732 >   public ArrayList<ShoppingCart> getShoppingCart(int userID) {···
756
757 >   public ArrayList<ShoppingCart> getShoppingCartByMonth(int month, int year) {···
781
782 >   public ArrayList<Item> getAvailableItems() {···
822
823 >   public ArrayList<Item> getAllItems() {···
864
865 >   public ArrayList<Item> getUnavailableItems() {···
906
907 >   public boolean removeItem(int id) {···
920
921 >   public boolean updateStatusItem(int id, String status) {···
934
935 >   public ArrayList<Item> getRemovedItem() {···
960
961 >   public ArrayList<Item> getUserItem(User user) {···
1002
1003 >   public ArrayList<Item> getPublisherItem(Publisher publisher) {···
1045
1046 >   public Transaction getTransactionByID(int id) {···
1068
1069 >   public Item getItemById(int id) {···
```

Formatting


```java
public ArrayList<ShoppingCart> getShoppingCart(int userID) {
    conn.connect();
    String query = "SELECT * FROM shoppingcart sc JOIN transaction t ON t.transaction_id = sc.transaction_id WHERE t.user_id = "
            + userID;
    ArrayList<ShoppingCart> transactions = new ArrayList<>();

    try {
        Statement stmt = conn.con.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            ShoppingCart transaction = new ShoppingCart();
            transaction.setTransactionID(rs.getInt("transaction_id"));
            transaction.setitemID(rs.getInt("item_id"));
            transaction.setDescription(rs.getString("description"));
            transactions.add(transaction);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        conn.disconnect(); // Close the connection when done
    }

    return transactions;
}
```

Design Pattern: **Singleton**


```
// Singleton design pattern for the controller so only a single instance of
    // controller exists while the program runs
    public static Controller getInstance() {
        if (instance == null) {
            instance = new Controller();
        }
        return instance;
    }
```

Design Pattern: **Adapter**

Tugas Besar RPL

```java
package model;

public interface NotificationService {
    void sendNotification(String message);
}
```

Konsep

Konsep kita menggunakan notifikasi untuk mengirimkan notifikasi via SMS atau Email kepada pengguna jika terdapat game baru yang di-publish oleh publisher atau jika ada game yang berubah status dari not available menjadi available.

```java
package model;

public class SMSNotificationService implements NotificationService {
    @Override
    public void sendNotification(String message) {
        System.out.println("Sending SMS: " + message);
    }
}
```

```java
package model;

public class EmailNotificationService implements NotificationService {
    @Override
    public void sendNotification(String message) {
        System.out.println("Sending email: " + message);
    }
}
```

Design Pattern: **Adapter**

Tugas Besar RPL


```java
// Check if publisher is not null before passing to insertNewGame
if (publisher != null) {
    boolean insert = con.insertNewGame(game, publisher);
    if (insert) {
        JOptionPane.showMessageDialog(null, "Insert successful");
        new HomePublisher(publisher);
        addItem.dispose();

        for (User user : userList) {
            user.notifyUser(gameName + " is now available!");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Insert failed");
    }
} else {
    JOptionPane.showMessageDialog(null, "Publisher information missing");
}
```

Implementasi

```java

if (statusCheckBox.isSelected()) {
    // ambil user List
    ArrayList<User> userList = con.getUnbannedUsers();

    for (User user : userList) {
        user.notifyUser(itemNameField.getText() + " received an update!");
    }
}
```

Tugas Besar RPL

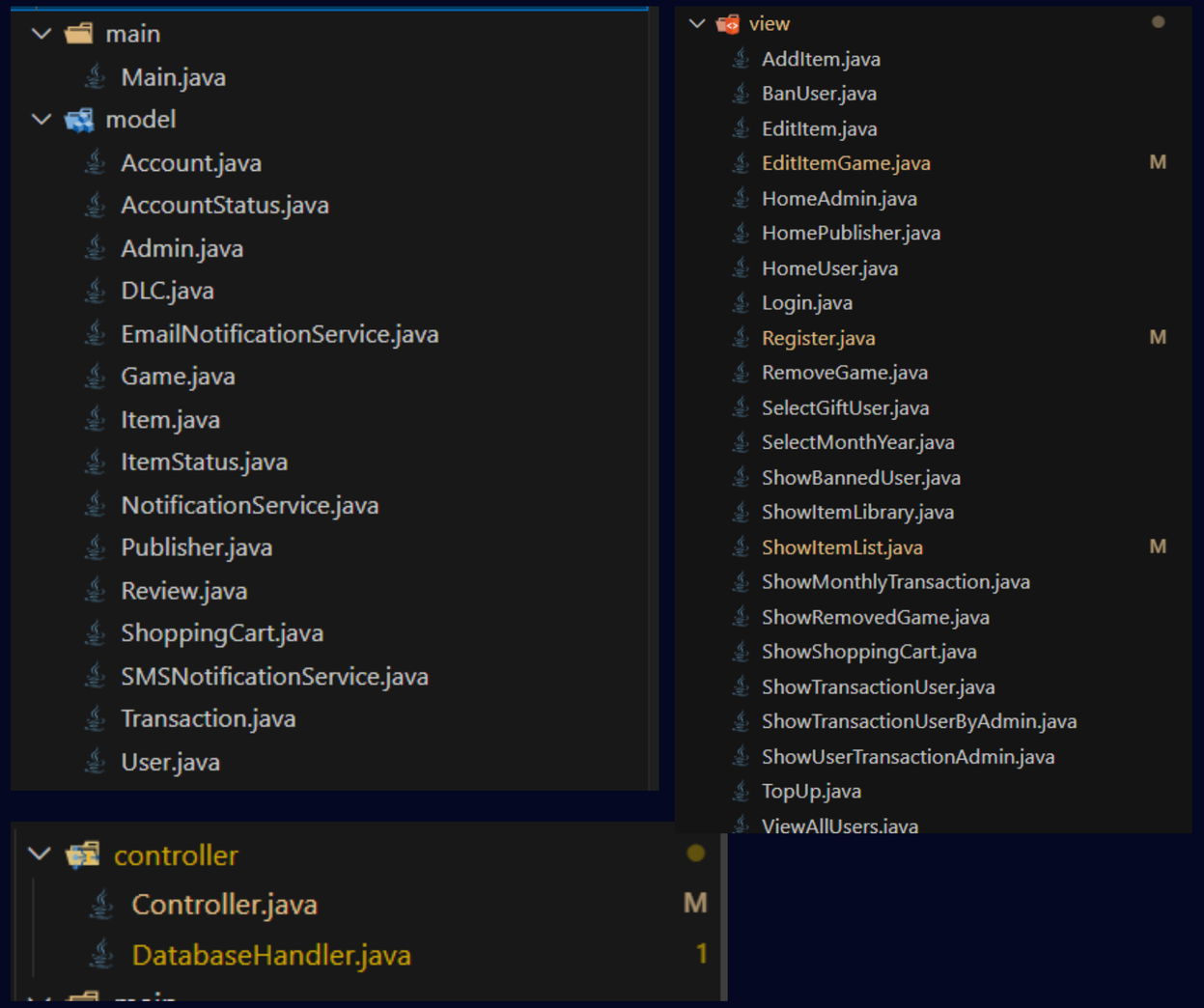
24
25
26

SOLID Principle

Single Responsibility Principle

25
26
27

Kami memastikan bahwa setiap kelas hanya memiliki satu tugas atau tanggung jawab



Open/Closed Principle (OCP)

Kami memastikan bahwa setiap kelas harus terbuka untuk ekstensi, namun tertutup untuk modifikasi.

26
27
28

Class Item

```java
public class Item {
    private int itemID;
    private String name;
    private String type; // game or DLC
    private String description;
    private double price;
    private int publisherID;
    private ArrayList<Review> reviews;
    private ItemStatus status; // available or not available
}
```

Class Game

```java
import java.util.ArrayList;

public class Game extends Item {
    private ArrayList<DLC> DLC;

    public Game() {

    }

    public ArrayList<model.DLC> getDLC() {
        return DLC;
    }

    public void setDLC(ArrayList<model.DLC> DLC) {
        this.DLC = DLC;
    }
}
```

Kami mengimplementasikan salah satu contohnya adalah untuk Class Game dan Class DLC yang meng-extend ke Class Item

Class DLC

```java
public class DLC extends Item {

    public DLC(int itemID, String name, String type, String description, double price, int publisherID,
            ArrayList<Review> reviews) {
        super(itemID, name, "DLC", description, price, publisherID, reviews);
    }

    public DLC() {
    }

}
```

Liskov Substitution Principle

```
1  String name = username.getText();
2  String pass = new String(password.getPassword());
3  Account newAccount = con.getUser(name, pass);
4  User newUser = (User) newAccount;
```

Class Registration

Pada Line 3, kelas User (Kelas Turunan) menggantikan kelas Account (Kelas Induk/Parent). Pewarisan ini tidak mempengaruhi keakuratan program

Dependency Inversion Principle

Konsep

- High-level modules tidak boleh bergantung pada low-level modules. Keduanya harus bergantung pada abstractions.
- Abstractions tidak boleh bergantung pada details. Details harus bergantung pada abstractions.

```java
package model;

public interface NotificationService {
    void sendNotification(String message);
}
```

```java
package model;

public class SMSNotificationService implements NotificationService {
    @Override
    public void sendNotification(String message) {
        System.out.println("Sending SMS: " + message);
    }
}

```

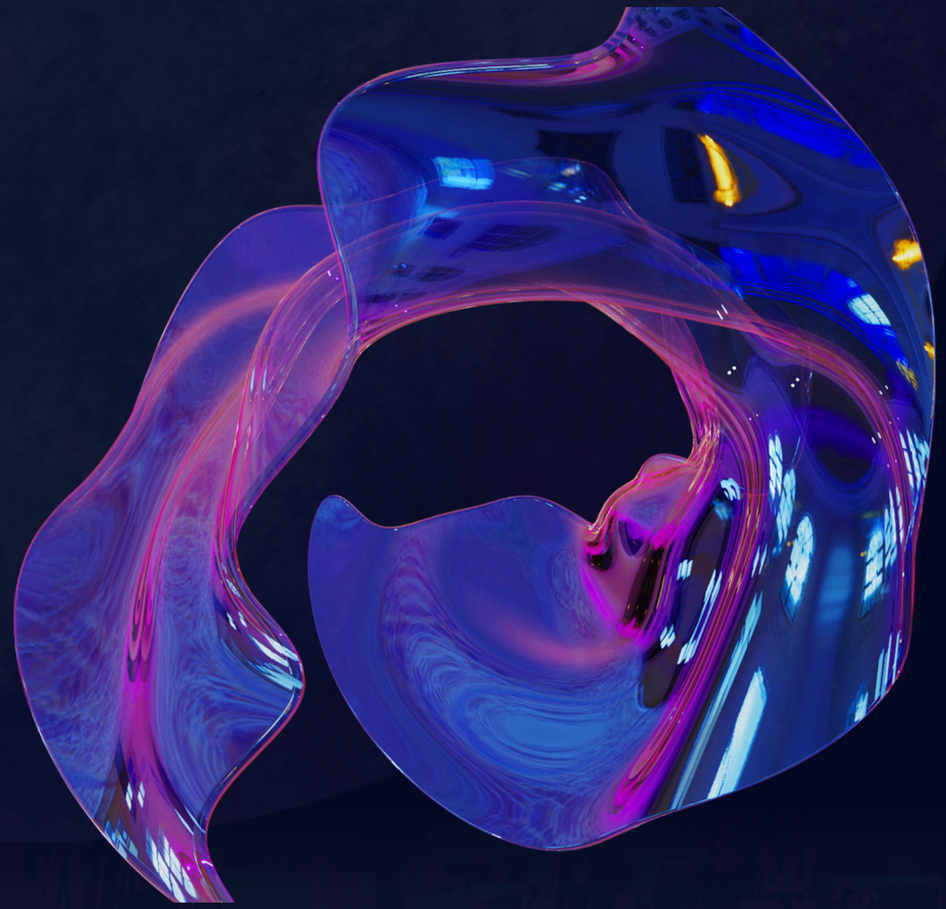
```java
package model;

public class EmailNotificationService implements NotificationService {
    @Override
    public void sendNotification(String message) {
        System.out.println("Sending email: " + message);
    }
}

```

Dependency Inversion Principle


```
// Check if publisher is not null before passing to insertNewGame
if (publisher != null) {
    boolean insert = con.insertNewGame(game, publisher);
    if (insert) {
        JOptionPane.showMessageDialog(null, "Insert successful");
        new HomePublisher(publisher);
        addItem.dispose();

        for (User user : userList) {
            user.notifyUser(gameName + " is now available!");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Insert failed");
    }
} else {
    JOptionPane.showMessageDialog(null, "Publisher information missing");
}
```

Implementasi

```

if (statusCheckBox.isSelected()) {
    // ambil user List
    ArrayList<User> userList = con.getUnbannedUsers();

    for (User user : userList) {
        user.notifyUser(itemNameField.getText() + " received an update!");
    }
}
```



Terima Kasih Atas

Perhatiannya