

**LAPORAN TUGAS AKHIR SEMESTER
PEMROGRAMAN BERORIENTASI OBJEK**

“RENTAL PS GDA”



DOSEN PENGAMPU MATA KULIAH:

Taufik Ridwan, S.T, M.T

DISUSUN OLEH:

Anisa Ayu Yandani	2210631250002
Gerald Dustin Albert	2210631250011
Muhammad Rizky Saputra	2210631250021
Amaliyah	2210631250042

**PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2024**

DAFTAR ISI

BAB I.....	4
1.1. LATAR BELAKANG.....	4
1.2. BATASAN.....	5
1.3. RUMUSAN MASALAH.....	5
1.4. TUJUAN MASALAH.....	6
BAB II.....	7
2.1. CLASS DIAGRAM.....	7
2.2. USECASE DIAGRAM.....	8
2.3. ACTIVITY DIAGRAM.....	9
2.3.1. Sebagai Customer.....	9
2.3.1.1. Customer mendaftarkan akun.....	9
2.3.1.2. Customer melakukan penyewaan.....	10
2.3.1.3. Customer melakukan pengembalian barang sewaan.....	11
2.3.1.4. Customer melihat bagian riwayat penyewaan.....	12
2.3.2. Sebagai Admin.....	13
2.3.1.1. Admin mendaftarkan akun.....	13
2.3.1.2. Admin menambahkan barang baru.....	14
2.3.1.3. Admin melihat riwayat penyewaan.....	15
BAB III.....	16
3.1. IMPLEMENTASI KODE.....	16
3.1.1. Penjelasan folder Controller.....	16
3.1.1.1. AddNewAccount.java.....	16
3.1.1.2. AddNewps.java.....	20
3.1.1.3. ChangePassword.java.....	24
3.1.1.4. Deleteps.java.....	26
3.1.1.5. EditUserData.java.....	30
3.1.1.6. Main.java.....	33
3.1.1.7. Quit.java.....	36
3.1.1.8. Rentps.java.....	37
3.1.1.9. Returnps.java.....	41
3.1.1.10. ShowAllRents.java.....	43
3.1.1.11. ShowSpecUserRents.java.....	46
3.1.1.12. ShowUserRents.java.....	48
3.1.1.13. Updateps.java.....	51
3.1.1.14. Viewpss.java.....	55
3.1.2. Penjelasan folder Model.....	58
3.1.1.1. Admin.java.....	58
3.1.1.2. Client.java.....	62
3.1.1.3. Database.java.....	64

3.1.1.4. JButton.java.....	66
3.1.1.5. JComboBox.java.....	67
3.1.1.6. JLabel.java.....	68
3.1.1.7. JPasswordField.java.....	69
3.1.1.8. JTable.java.....	70
3.1.1.9. JTextField.java.....	71
3.1.1.10. Operation.java.....	71
3.1.1.11. Rent.java.....	72
3.1.1.12. User.java.....	73
3.1.1.13. ps.java.....	75
3.2. PENGUJIAN.....	77
3.2.1. Tampilan Pengujian.....	77
3.2.1.1. Registrasi.....	77
3.2.1.2. Login.....	79
3.2.1.3. Dashboard.....	80
3.2.1.4. Penyewaan.....	84
3.2.1.5. Pengembalian.....	86
3.2.1.6. Input Data.....	87
BAB IV.....	94
LAMPIRAN.....	95

BAB I

PENDAHULUAN

1.1. LATAR BELAKANG

GUI atau *Graphical User Interface* merupakan sistem operasi antarmuka berbasis grafis dengan menggunakan ikon, tombol, menu dan representasi visual lainnya untuk mendukung interaksi pengguna dengan sistem. Dengan menggunakan GUI, sistem menjadi lebih *user-friendly* kepada pengguna nya, dan mempermudah dalam menyelesaikan tugas lebih cepat, serta visualisasi yang lebih menarik. Rental merupakan sebuah kegiatan persewaan yaitu pemakaian suatu barang atau jasa dengan membayar uang atau ongkos yang dibayarkan karena memakai atau meminjam sesuatu. Rental PS pada kegiatan ini membutuhkan aplikasi pendataan yang lebih akurat dan cepat dalam menangani kegiatan persewaan para penyewa dalam mencatat jumlah waktu dan banyaknya barang yang disewakan.

Pada pengembangan GUI dengan konsep OOP (Object Oriented Program) menggunakan bahasa pemrograman Java merupakan pilihan yang tepat. Konsep OOP merupakan jenis pemrograman yang mendefinisikan tipe data dari struktur data jenis operasi yang dapat diterapkan pada struktur data tersebut. Java merupakan bahasa pemrograman bersifat *open source* dan modern untuk mengembangkan aplikasi platform desktop, web, mobile hingga embedded dan IoT. Java memiliki kelebihan fleksibilitas, portabilitas serta keamanan.

Implementasi program yang dilakukan menjadi kebutuhan bagi pemilik rental dalam melakukan pendataan dan *controlling* terhadap barang yang disewakan, jumlah waktu setiap penyewaan, dan transaksi yang dilakukan serta melakukan pemeliharaan dan pemeriksaan rutin terhadap barang sewaannya, sehingga barang sewaan bisa lebih terawat dan terjaga. Selain untuk kebutuhan pendataan, penyewa juga dengan mudah melakukan penyewaan dengan pencatatan waktu yang *real-time* dan lebih akurat. Dengan manfaat ini, diharapkan dapat membantu pemilik untuk melakukan pendataan lebih efisien, akurat, dan efektif.

Oleh karena itu, pada laporan ini, kelompok kami akan membahas implementasi konsep OOP dalam pembuatan aplikasi GUI Rental PS menggunakan bahasa pemrograman Java, serta integrasi fitur CRUD untuk memudahkan aktivitas penyewaan dan pendataan. Diharapkan aplikasi yang dikembangkan dapat memberikan solusi yang efektif dan efisien dalam pendataan rental.

1.2. BATASAN

Aplikasi rental PS ini merupakan sistem berbasis GUI yang membantu dalam mengatasi permasalahan dalam melakukan penyewaan dan pendataan manual yang memberikan efisiensi, akurasi dan keamanan data pada proses pendataan dan pengelolaannya. Dalam sistem ini memiliki fitur berikut :

- Penyewa dapat melakukan penyewaan saat awal untuk memilih lama waktu penyewaan, barang yang disewa serta jumlahnya. Kemudian data waktu langsung ditampilkan untuk lama penyewaan secara *real-time*.
- Admin dapat mengecek rekap penyewaan barang harian.
- Admin dapat menambahkan dan menghapus data barang sewaan.

1.3. RUMUSAN MASALAH

Dalam rumusan masalah ini akan membahas secara mendalam mengenai beberapa aspek terkait aplikasi Rental PS. Berikut adalah aspek-aspek yang akan dibahas:

- Bagaimana implementasi konsep OOP dalam pembuatan aplikasi GUI Rental PS menggunakan bahasa pemrograman Java?
- Bagaimana efektivitas dan efisiensi penggunaan aplikasi Rental PS dalam manajemen data dan kemudahan penyewaan bagi customer?
- Bagaimana proses pendataan pada penyewa dan bagaimana data penyewaan tercatat dalam database?
- Bagaimana admin dapat mengakses dan mengecek rekap barang sewaan?
- Bagaimana admin dapat menambahkan dan menghapus data barang sewaan?

1.4. TUJUAN MASALAH

Tujuan dari laporan ini adalah untuk membahas mengenai aspek-aspek terkait aplikasi Rental PS. Dalam laporan ini, akan diuraikan dengan detail tujuan-tujuan yang ingin dicapai dari implementasi aplikasi Rental PS, berikut ini:

- Menerapkan konsep OOP dalam pembuatan aplikasi GUI Rental PS menggunakan bahasa pemrograman Java.
- Meningkatkan efektivitas dan efisiensi manajemen data melalui penggunaan aplikasi Rental PS yang dikembangkan.
- Memudahkan penyewa untuk melakukan proses penyewaan dan merekam data ke dalam database.
- Memungkinkan admin untuk mengakses dan mengecek rekap data barang sewaan.
- Membantu admin dalam menambahkan data barang sewaan.

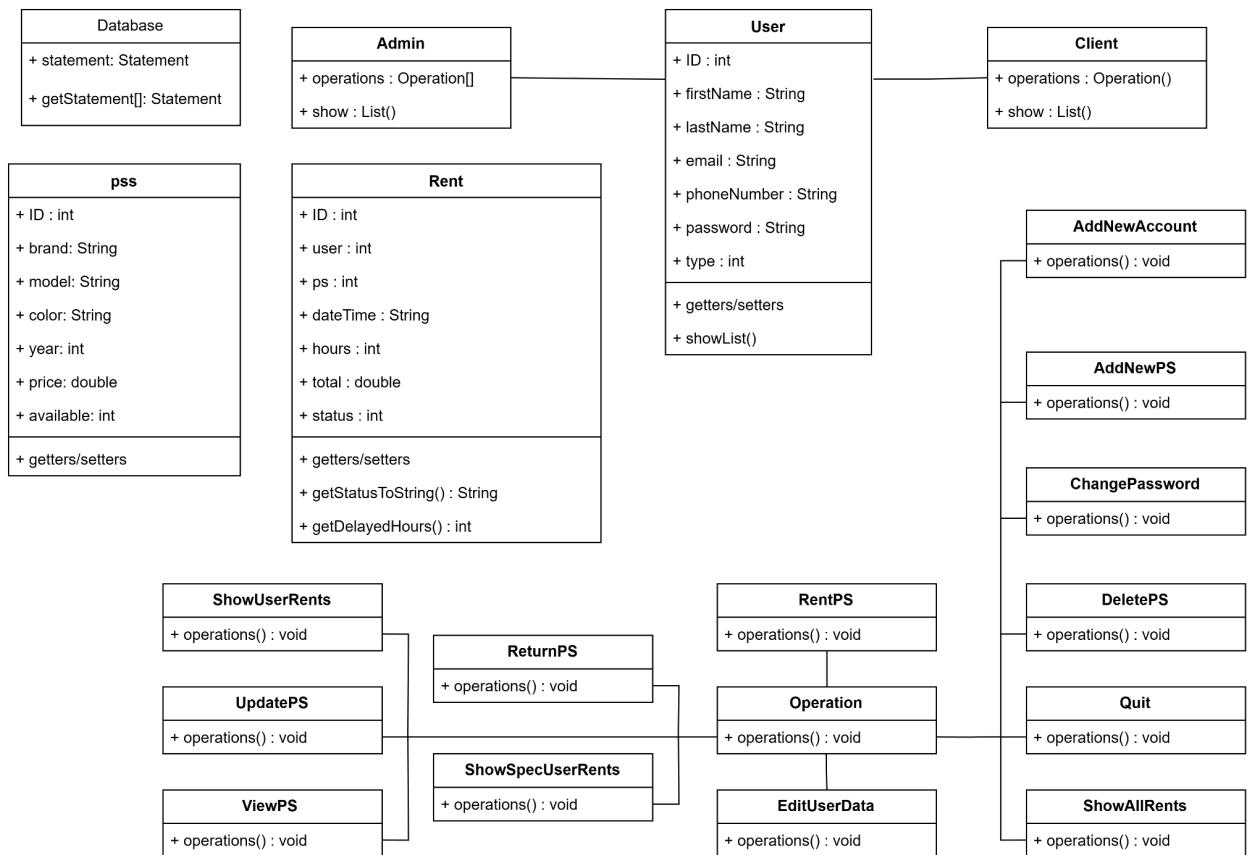
BAB II

PERANCANGAN UML DIAGRAM

2.1. CLASS DIAGRAM

Class diagram merupakan diagram yang menggambarkan struktur sistem dengan menunjukkan class, atribut, metode, dan hubungan antar objek. Class diagram membantu dalam penjelasan struktur sistem dan menunjukkan objek yang saling berinteraksi dan berkomunikasi satu sama lain. Class adalah spesifikasi apabila di instansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Pada class diagram berikut, terdapat tiga tabel utama, yaitu *cars*, *rents*, dan *users*.

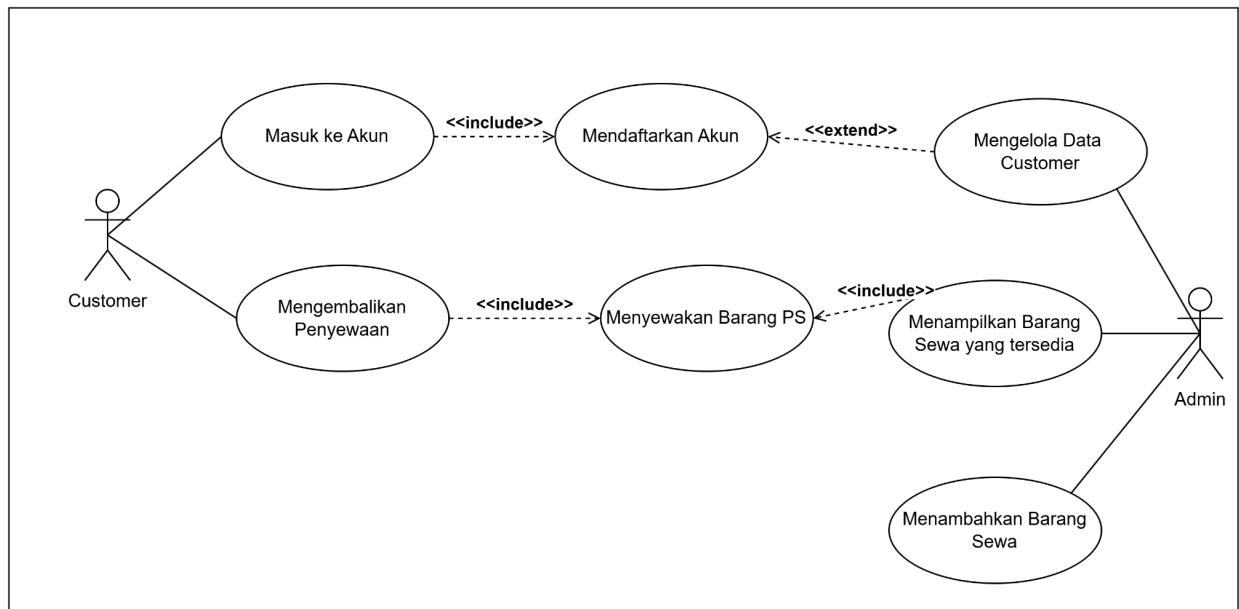
Pada tabel *users*, di definisikan sebagai 2 jenis yaitu admin dan *client*. Kemudian di tabel *pss* merupakan isi spesifikasi dari tipe ps berupa *brand*, *model*, *color* dan lainnya. Dan di bagian tabel *rents* yang berisi detail dari rental yang dilakukan oleh *client* dengan data nya adalah waktu dari tanggal penyewaan, jam awal sampai batas akhir penyewaan dan juga status.



2.2. USECASE DIAGRAM

Usecase diagram merupakan diagram yang menggambarkan fungsi, ruang lingkup dan interaksi pengguna dengan sistem. Dengan memvisualisasikan interaksi antara pengguna (aktor) dan sistem (*use case*), serta tindakan yang dapat dilakukan aktor terhadap *use case* secara rinci.

Pada usecase berikut, terdapat 2 aktor yang pertama adalah customer dan kedua ada Admin. Customer harus masuk ke akun jika ingin melakukan penyewaan, dan apabila belum memiliki akun maka langsung untuk mendaftarkan akunnya. Dan data customer dikelola oleh Admin. Dan Customer yang melakukan penyewaan harus melakukan pengembalian, dan Admin akan menampilkan barang-barang PS yang bisa disewakan oleh customer dan Admin juga bisa melakukan penambahan barang PS apabila diperlukan.



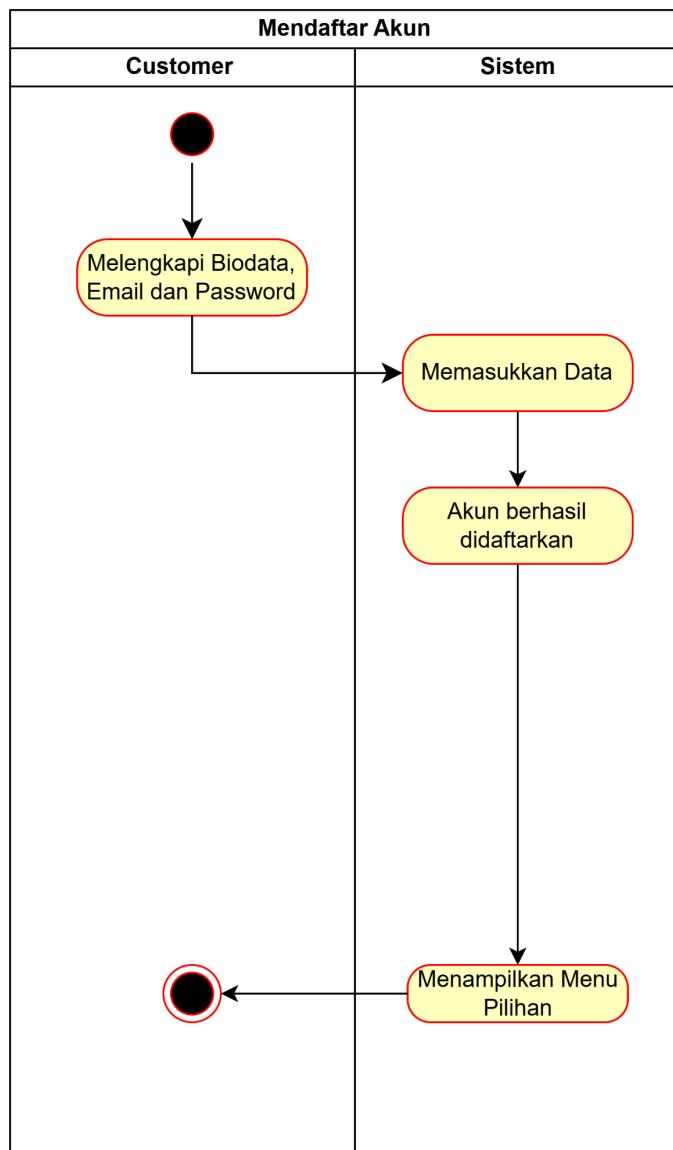
2.3. ACTIVITY DIAGRAM

Activity diagram merupakan diagram yang memodelkan proses-proses pada sebuah sistem dengan urutan yang digambarkan secara virtual.

2.3.1. Sebagai Customer

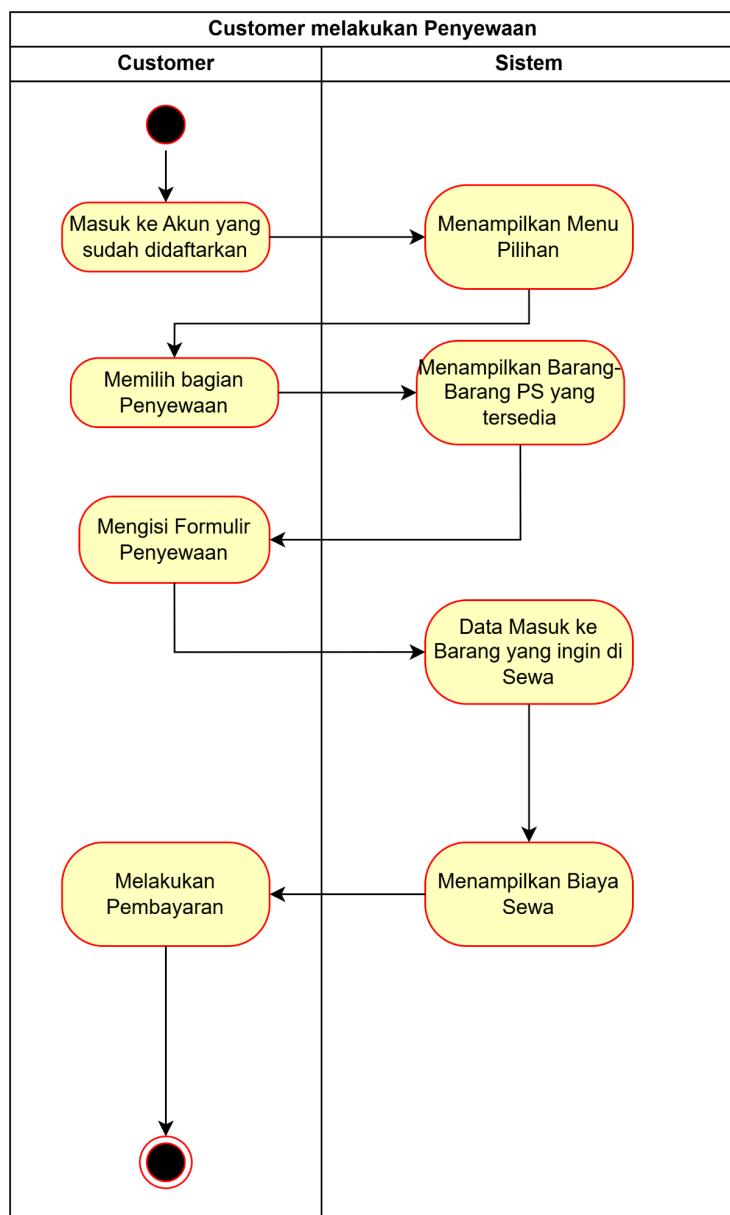
2.3.1.1. Customer mendaftarkan akun

Sebelum melakukan aktivitas penyewaan, customer diharuskan membuat akun dengan melengkapi biodata seperti nama, email dan *password*. Selanjutnya sistem akan menyimpan hasil sebagai bukti sudah membuat akun. Kemudian sistem akan menampilkan menu pilihan agar customer bisa melanjutkan untuk ke bagian perentalan.



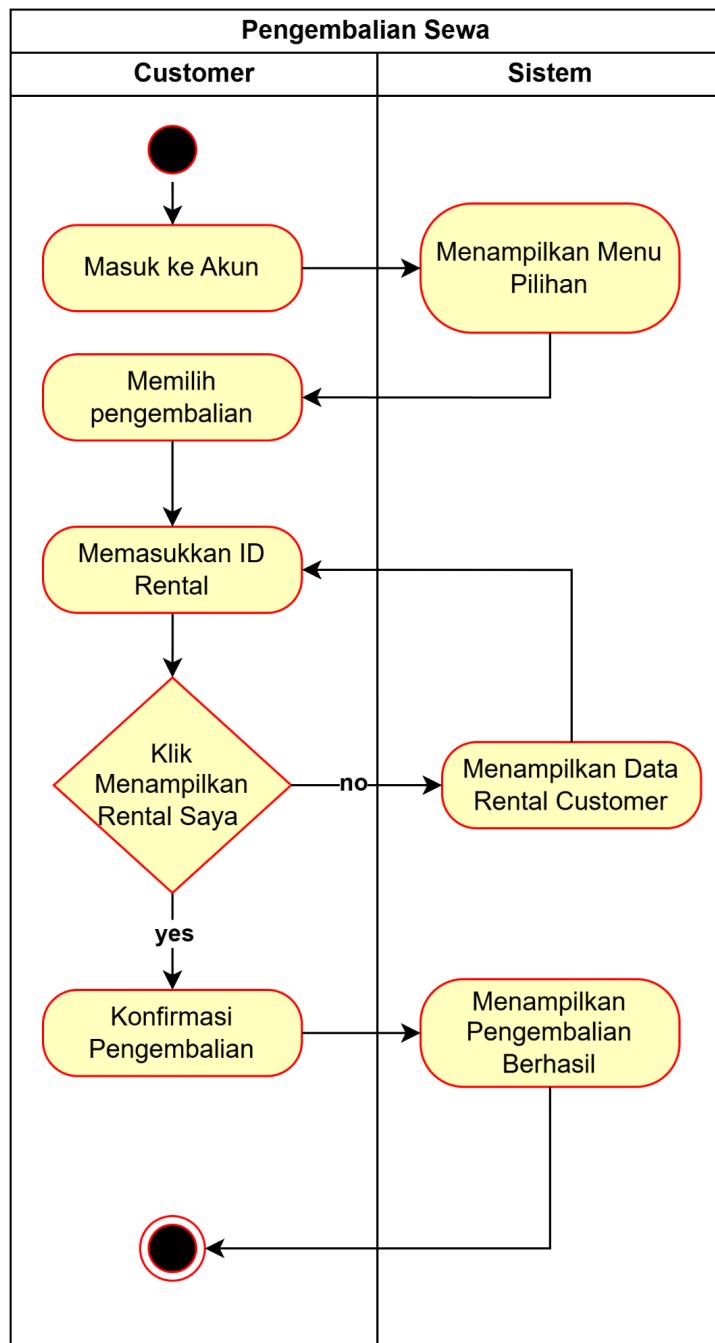
2.3.1.2. Customer melakukan penyewaan

Customer yang sudah melakukan Login atau masuk ke akun akan diarahkan pada tampilan menu pilihan. Setelahnya customer memilih pada button penyewaan dan sistem kemudian menampilkan barang-barang PS yang bisa disewakan. Selanjutnya customer akan melakukan pengisian form untuk nama, tipe PS yang ingin disewa, dan waktu penyewaan. Lanjut, customer selesai mengisi form maka sistem akan langsung meneruskan hasil form pada database untuk disimpan dan sistem selanjutnya menampilkan total harga sewaan yang harus dibayarkan oleh customer. Maka kegiatan penyewaan berhasil dilakukan.



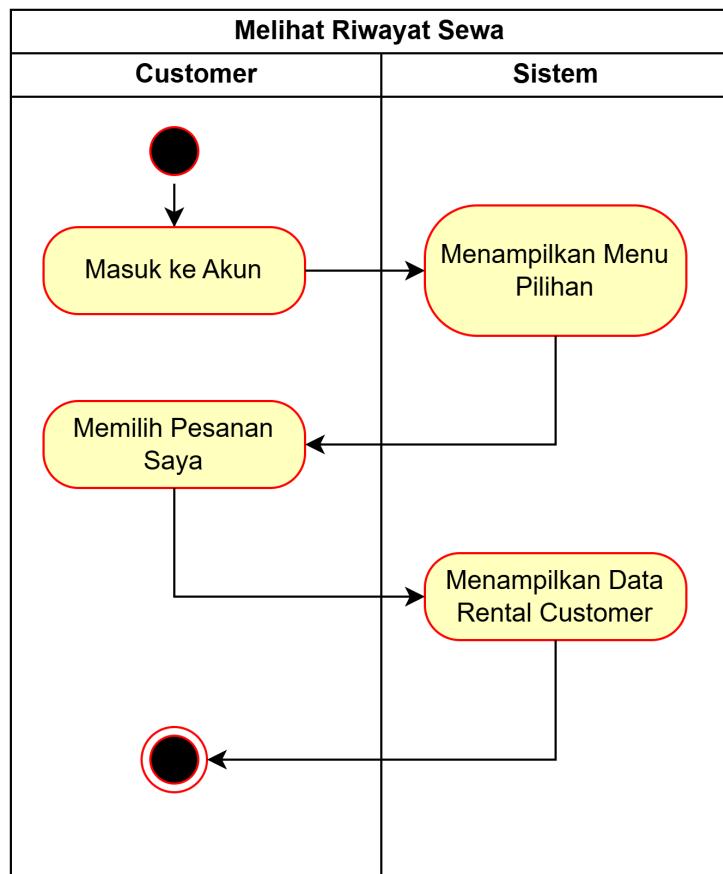
2.3.1.3. Customer melakukan pengembalian barang sewaan

Untuk melakukan pengembalian barang, customer bisa memasukkan akunnya terlebih dahulu kemudian sistem akan menampilkan bagian menu pilihan. Setelahnya customer bisa memilih ke bagian pengembalian dan selanjutnya mengisi ID yang sebelumnya sudah didapat di bagian penyewaan. Selanjutnya customer masuk dengan klik bagian menampilkan rental saya, dan kemudian terdapat dua pilihan, yaitu untuk menampilkan data rental customer atau melakukan pengembalian. Karena sudah memasukkan ID sebelum melakukan konfirmasi pengembalian, maka cukup di klik dan sistem langsung menampilkan pengembalian berhasil.



2.3.1.4. Customer melihat bagian riwayat penyewaan

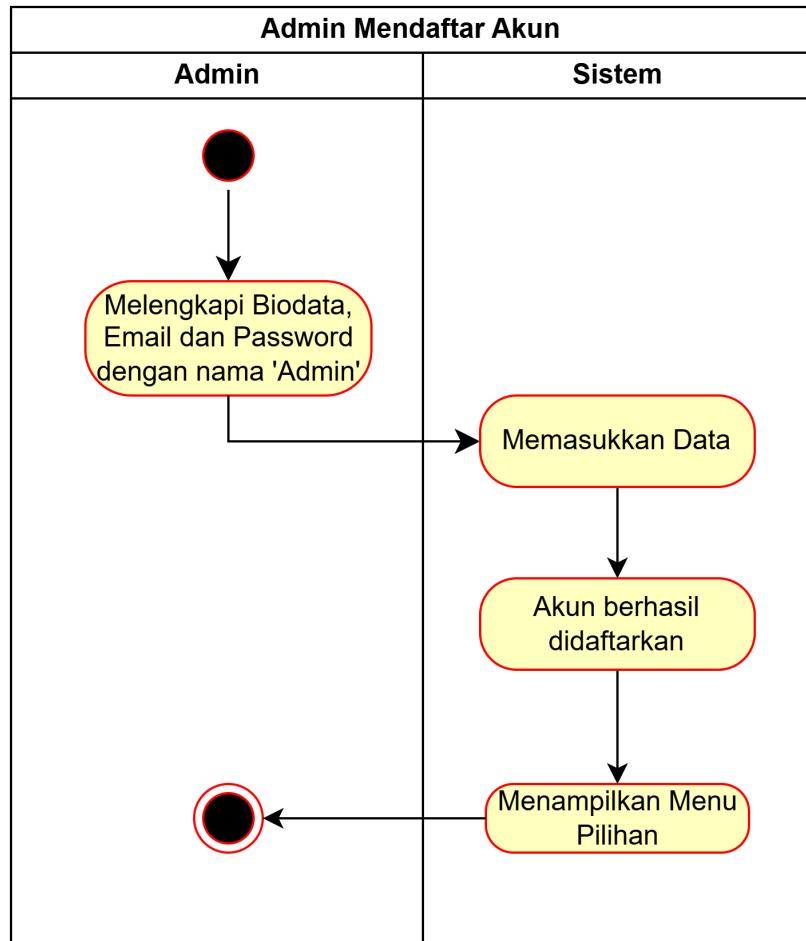
Sebelum ditampilkan menu pilihan oleh sistem, customer harus memasukkan akunnya terlebih dahulu. Lanjut customer bisa memilih ke bagian Pesanan saya, dan kemudian sistem akan menampilkan data rental milik customer.



2.3.2. Sebagai Admin

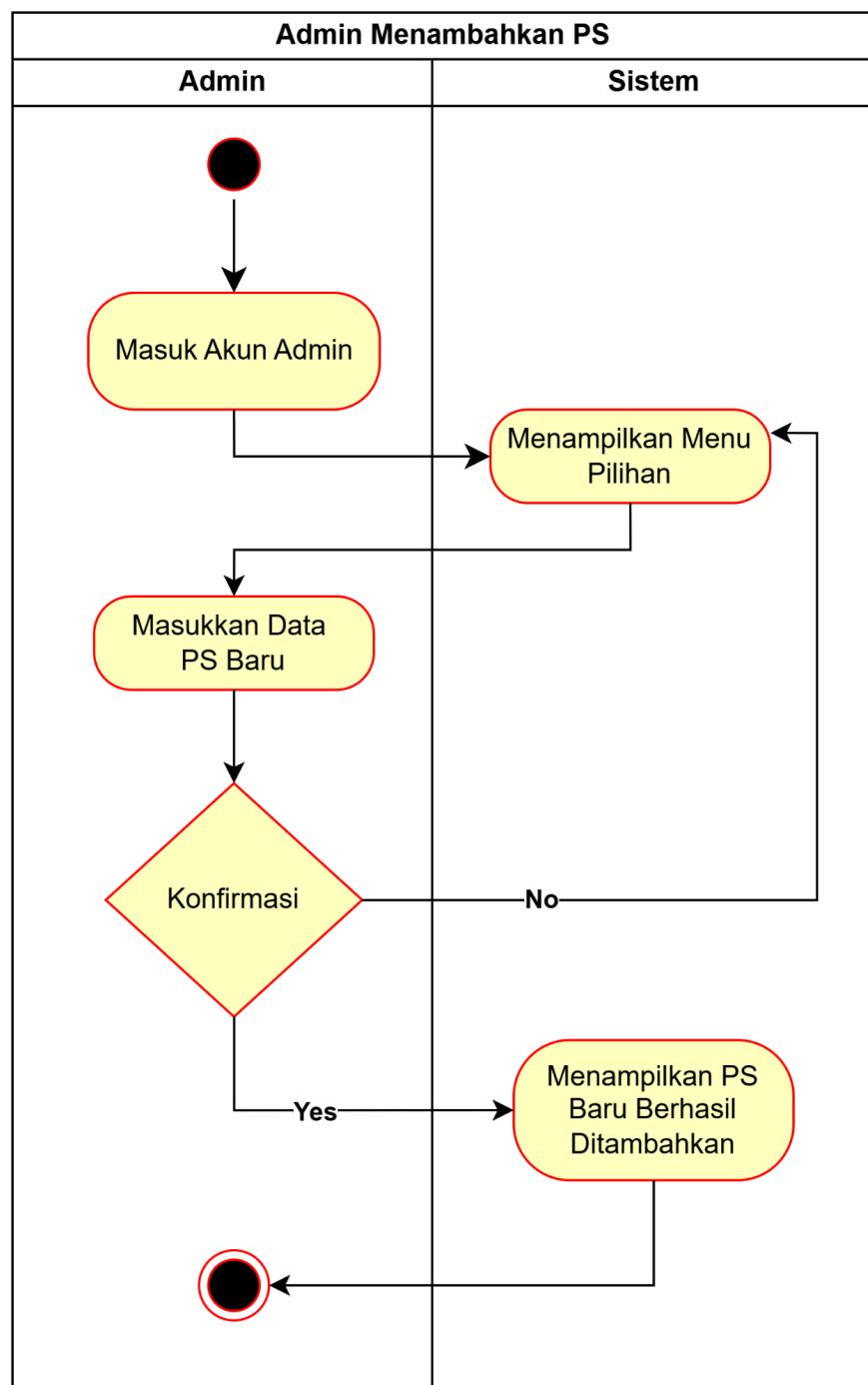
2.3.1.1. Admin mendaftarkan akun

Admin yang ingin membuat akun diharuskan melengkapi biodata dengan nama admin, kemudian email dan password. Setelah itu sistem akan memasukkan data yang sudah diinput ke bagian database dan akun berhasil didaftarkan dan sistem menampilkan menu pilihan.



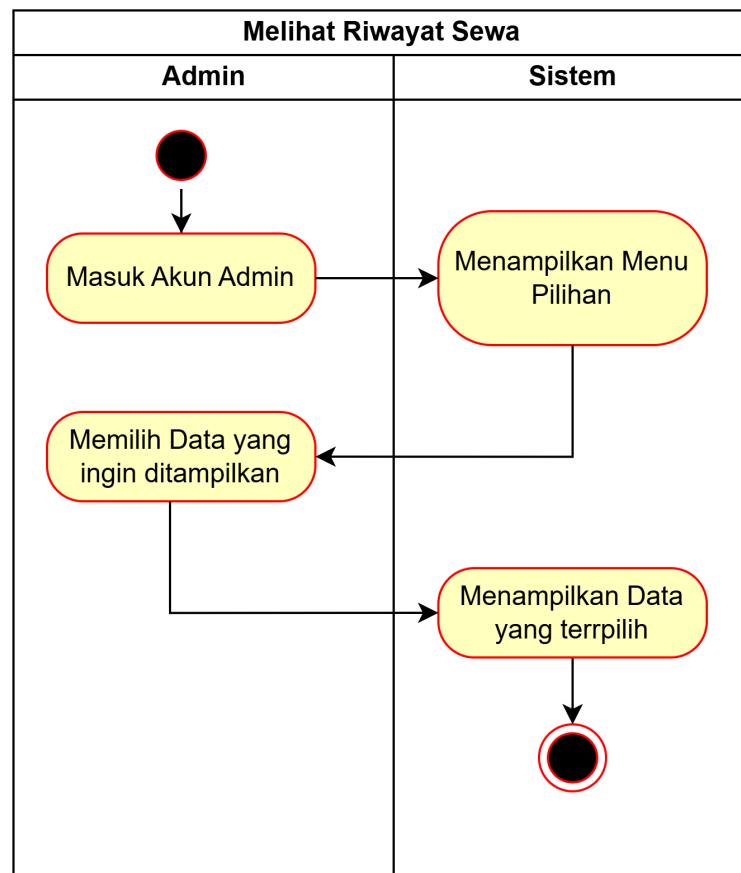
2.3.1.2. Admin menambahkan barang baru

Admin memiliki hak sebagai penambah pasokan barang, dengan masuk akun kemudian sistem akan menampilkan menu pilihan. Admin kemudian masuk ke bagian Masukkan data PS baru. Setelahnya melakukan konfirmasi, apabila klik Ya, maka sistem akan menampilkan berhasil ditambahkan, apabila tidak maka kembali ke menu pilihan.



2.3.1.3. Admin melihat riwayat penyewaan

Admin dapat mengakses data penyewaan dengan masuk ke bagian data yang ingin ditampilkan. Kemudian sistem akan langsung menampilkan data yang ingin ditampilkan.



BAB III

IMPLEMENTASI DAN PENGUJIAN

3.1. IMPLEMENTASI KODE

Pada tahap implementasi, menggunakan `java.awt` sebagai tampilan GUI. Untuk database menggunakan server lokal yaitu *XAMPP* dan *database management system* menggunakan *Mysql* sebagai tempat penyimpanan sentral database. Terdapat dua folder yang dibuat sebagai implementasi, yaitu folder Controller dan Model. Pada folder controller untuk membuat tampilan dan pada model untuk bagian sistem yang akan dijalankan pada aplikasi.

3.1.1. Penjelasan folder Controller

3.1.1.1. AddNewAccount.java

`AddNewAccount` digunakan untuk membuat akun baru pada sistem aplikasi rental PlayStation (PS) yang ditampilkan dalam GUI berbasis Swing.

```
1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.GradientPaint;
6 import java.awt.Graphics;
7 import java.awt.Graphics2D;
8 import java.awt.GridLayout;
9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14
15 import javax.swing.BorderFactory;
16 import javax.swing.JFrame;
17 import javax.swing.JOptionPane;
18 import javax.swing.JPanel;
19
20 import Model.Client;
21 import Model.Database;
22 import Model.JButton;
23 import Model.JLabel;
24 import Model.JPasswordField;
25 import Model.JTextField;
26 import Model.Operation;
27 import Model.User;
```

Kode di atas mengimpor paket-paket yang diperlukan untuk menjalankan program. `java.awt` dan `javax.swing` digunakan untuk membuat antarmuka grafis, `java.sql` untuk berinteraksi dengan database, dan `Model` adalah paket kustom yang berisi kelas-kelas model aplikasi.

```

29  public class AddNewAccount implements Operation {
30
31      private int accType;
32
33      public AddNewAccount(int accType) {
34          this.accType = accType;
35      }

```

Kelas AddNewAccount mengimplementasikan antarmuka Operation. Kelas ini memiliki satu atribut accType yang menunjukkan jenis akun yang akan dibuat, dan konstruktor untuk menginisialisasi atribut tersebut.

```

37  @Override
38  public void operation(Database database, JFrame f, User u) {
39
40      JFrame frame = new JFrame(title:"Create New Account");
41      frame.setSize(width:600, height:600);
42      frame.setLocationRelativeTo(f);
43      frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
44

```

Bagian ini membuat jendela baru berjudul "Create New Account" dengan ukuran 600x600 piksel, menempatkannya di tengah relatif terhadap frame induk, dan mengatur agar jendela ditutup saat tidak digunakan lagi.

```

45  // Panel dengan latar belakang gradasi
46  JPanel gradientPanel = new JPanel() {
47      @Override
48      protected void paintComponent(Graphics g) {
49          super.paintComponent(g);
50          Graphics2D g2d = (Graphics2D) g;
51          int width = getWidth();
52          int height = getHeight();
53          Color color1 = new Color(r:52, g:73, b:94);
54          Color color2 = new Color(r:44, g:62, b:80);
55          GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, height, color2);
56          g2d.setPaint(gp);
57          g2d.fillRect(x:0, y:0, width, height);
58      }
59  };
60  gradientPanel.setLayout(new BorderLayout());

```

Bagian ini membuat JPanel dengan latar belakang gradasi. Metode paintComponent dioverride untuk menggambar gradasi warna menggunakan GradientPaint.

```

62  JLabel title = new JLabel(text:"WELCOME TO RENTAL PS GDA", fontSize:35);
63  title.setForeground(Color.WHITE);
64  title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
65  gradientPanel.add(title, BorderLayout.NORTH);

```

Judul "WELCOME TO RENTAL PS GDA" dibuat sebagai JLabel dengan ukuran font 35 dan warna teks putih, kemudian ditambahkan ke bagian atas panel gradasi.

```

67  JPanel panel = new JPanel(new GridLayout(rows:7, cols:2, hgap:15, vgap:15));
68  panel.setBackground(new Color(r:0, g:0, b:0, a:0));
69  panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
70

```

Form input dibuat menggunakan GridLayout 7x2 dengan celah 15 piksel antar elemen dan padding 20 piksel. Panel ini transparan.

```

71   JLabel firstNameLabel = new JLabel(text:"First Name:", textSize:22);
72   firstNameLabel.setForeground(Color.WHITE); // Mengatur warna teks label menjadi putih
73   panel.add(firstNameLabel);
74   JTextField firstName = new JTextField(textSize:22);
75   panel.add(firstName);
76
77   JLabel lastNameLabel = new JLabel(text:"Last Name:", textSize:22);
78   lastNameLabel.setForeground(Color.WHITE);
79   panel.add(lastNameLabel);
80   JTextField lastName = new JTextField(textSize:22);
81   panel.add(lastName);
82
83   JLabel emailLabel = new JLabel(text:"Email:", textSize:22);
84   emailLabel.setForeground(Color.WHITE);
85   panel.add(emailLabel);
86   JTextField email = new JTextField(textSize:22);
87   panel.add(email);
88
89   JLabel phoneNumberLabel = new JLabel(text:"Phone Number:", textSize:22);
90   phoneNumberLabel.setForeground(Color.WHITE);
91   panel.add(phoneNumberLabel);
92   JTextField phoneNumber = new JTextField(textSize:22);
93   panel.add(phoneNumber);
94
95   JLabel passwordLabel = new JLabel(text:"Password:", textSize:22);
96   passwordLabel.setForeground(Color.WHITE);
97   panel.add(passwordLabel);
98   JPasswordField password = new JPasswordField(textSize:22);
99   panel.add(password);

101  JLabel confirmPasswordLabel = new JLabel(text:"Confirm Password:", textSize:22);
102  confirmPasswordLabel.setForeground(Color.WHITE);
103  panel.add(confirmPasswordLabel);
104  JPasswordField confirmPassword = new JPasswordField(textSize:22);
105  panel.add(confirmPassword);

```

Setiap label dan field input ditambahkan ke panel form. Warna teks label diatur menjadi putih agar kontras dengan latar belakang gradasi. Field input termasuk First Name, Last Name, Email, Phone Number, Password, dan Confirm Password.

```

106  JButton login = new JButton(text:"Login", textSize:22);
107  login.setBackground(new Color(r:41, g:128, b:185));
108  login.setForeground(Color.WHITE);
109  login.addActionListener(new ActionListener() {
110      @Override
111      public void actionPerformed(ActionEvent e) {
112          Main.start();
113          frame.dispose();
114      }
115  });
116  panel.add(login);
117

```

Tombol Login ditambahkan ke panel form. Tombol ini memiliki latar belakang biru dan teks putih. Ketika tombol diklik, metode Main.start() dipanggil dan frame pendaftaran ditutup.

```

119 JButton createAcc = new JButton(text:'Create Account', textSize:22);
120 createAcc.setBackground(new Color(r:41, g:128, b:185));
121 createAcc.setForeground(Color.WHITE);
122 createAcc.addActionListener(new ActionListener() {
123     @SuppressWarnings("deprecation")
124     @Override
125     public void actionPerformed(ActionEvent e) {
126         if (firstName.getText().equals(anObject:"")) {
127             JOptionPane.showMessageDialog(frame, message:"First Name cannot be empty");
128             return;
129         }
130         if (lastName.getText().equals(anObject:"")) {
131             JOptionPane.showMessageDialog(frame, message:"Last Name cannot be empty");
132             return;
133         }
134         if (email.getText().equals(anObject:"")) {
135             JOptionPane.showMessageDialog(frame, message:"Email cannot be empty");
136             return;
137         }
138         if (phoneNumber.getText().equals(anObject:"")) {
139             JOptionPane.showMessageDialog(frame, message:"Phone Number cannot be empty");
140             return;
141         }
142         if (password.getText().equals(anObject:"")) {
143             JOptionPane.showMessageDialog(frame, message:"Password cannot be empty");
144             return;
145         }
146         if (confirmPassword.getText().equals(anObject:"")) {
147             JOptionPane.showMessageDialog(frame, message:"Confirm Password cannot be empty");
148             return;
149         }
150         if (!password.getText().equals(confirmPassword.getText())) {
151             JOptionPane.showMessageDialog(frame, message:"Password doesn't match");
152             return;
153         }
154         try {
155             ArrayList<String> emails = new ArrayList<>();
156             ResultSet rs0 = database.getStatement().executeQuery(sql:"SELECT Email FROM users");
157             while (rs0.next()) {
158                 emails.add(rs0.getString(columnLabel:"Email"));
159             }
160             if (emails.contains(email.getText())) {
161                 JOptionPane.showMessageDialog(frame, message:"This email is already used");
162                 return;
163             }
164             ResultSet rs = database.getStatement().executeQuery(sql:"SELECT COUNT(*) FROM users");
165             rs.next();
166             int ID = rs.getInt(columnLabel:"COUNT(*)");
167
168             String insert = "INSERT INTO users(ID, FirstName, LastName,"
169                         + " Email, PhoneNumber, Password, Type) VALUES"
170                         + " (" + ID + "," + firstName.getText() + "','" + lastName.getText() + "','" +
171                         + email.getText() + "','" +
172                         + phoneNumber.getText() + "','" + password.getText() + "','" + accType + "')";
173             database.getStatement().execute(insert);
174             JOptionPane.showMessageDialog(frame, message:"Account created successfully");
175
176             if (accType == 0) {
177                 User user = new Client();
178                 user.setID(ID);
179                 user.setFirstName(firstName.getText());
180                 user.setLastName(lastName.getText());
181                 user.setEmail(email.getText());
182                 user.setPhoneNumber(phoneNumber.getText());
183                 user.setPassword(password.getText());
184                 user.showList(database, frame);
185                 frame.dispose();
186             }
187
188         } catch (SQLException e1) {
189             JOptionPane.showMessageDialog(frame, e1.getMessage());
190         }
191     }
192 }
193 }
194 }
195 });
196 panel.add(createAcc);

```

Tombol Create Account juga ditambahkan dengan warna latar belakang biru dan teks putih. Ketika tombol diklik, metode ini melakukan validasi pada setiap field input untuk memastikan tidak ada yang kosong dan password cocok dengan konfirmasi password. Setelah validasi, metode ini memeriksa apakah email sudah digunakan dengan melakukan query ke database. Jika email belum digunakan,

data pengguna baru disimpan ke dalam tabel users di database. Jika akun berhasil dibuat dan tipe akun adalah 0, objek Client dibuat dan diinisialisasi dengan data pengguna baru, lalu frame pendaftaran ditutup.

```
198     gradientPanel.add(panel, BorderLayout.CENTER);
199     frame.add(gradientPanel, BorderLayout.CENTER);
200     frame.setVisible(b:true);
201 }
202 }
```

Panel form ditambahkan ke panel gradasi, dan panel gradasi ditambahkan ke frame. Frame kemudian ditampilkan kepada pengguna dengan mengatur visibilitasnya menjadi true.

3.1.1.2. AddNewps.java

Kode AddNewps bertujuan untuk menampilkan antarmuka pengguna untuk menambahkan unit PS baru.

```
1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.GradientPaint;
6 import java.awt.Graphics;
7 import java.awt.Graphics2D;
8 import java.awt.GridLayout;
9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13
14 import javax.swing.BorderFactory;
15 import javax.swing.JFrame;
16 import javax.swing.JOptionPane;
17 import javax.swing.JPanel;
18
19 import Model.Database;
20 import Model.JButton;
21 import Model.JLabel;
22 import Model.JTextField;
23 import Model.Operation;
24 import Model.User;
25
```

Paket yang diperlukan diimpor untuk mengakses kelas dan fungsi yang digunakan dalam program, termasuk paket untuk pengaturan GUI dan interaksi dengan database.

```
26 public class AddNewps implements Operation {
27
28     @Override
29     public void operation(Database database, JFrame f, User user) {
30 }
```

Metode operation adalah metode utama yang dijalankan saat operasi ini dipanggil. Metode ini menerima objek Database, frame JFrame, dan objek User.

```

28     @Override
29     public void operation(Database database, JFrame f, User user) {
30
31         JFrame frame = new JFrame(title:"Add New PS Unit");
32         frame.setSize(width:800, height:600);
33         frame.setLocationRelativeTo(f);
34         frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
35

```

JFrame baru dibuat dengan judul "Add New PS Unit", ukuran 800x600 piksel, dan ditutup saat tidak diperlukan.

```

36     // Panel dengan latar belakang gradasi
37     JPanel gradientPanel = new JPanel() {
38         @Override
39         protected void paintComponent(Graphics g) {
40             super.paintComponent(g);
41             Graphics2D g2d = (Graphics2D) g;
42             int width = getWidth();
43             int height = getHeight();
44             Color color1 = new Color(r:224, g:255, b:255);
45             Color color2 = new Color(r:0, g:0, b:255);
46             GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, height, color2);
47             g2d.setPaint(gp);
48             g2d.fillRect(x:0, y:0, width, height);
49         }
50     };
51     gradientPanel.setLayout(new BorderLayout());
52

```

Panel gradasi dibuat untuk efek latar belakang gradasi dengan warna biru dan putih.

```

52     JLabel title = new JLabel(text:"Add New PS Unit", fontSize:35);
53     title.setForeground(Color.BLUE);
54     title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
55     gradientPanel.add(title, BorderLayout.NORTH);
56
57

```

Judul "Add New PS Unit" ditambahkan ke panel dengan ukuran font 35 dan warna teks biru.

```

57
58     JPanel panel = new JPanel(new GridLayout(rows:6, cols:2, hgap:15, vgap:15));
59     panel.setOpaque(isOpaque:false);
60     panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
61
62     JLabel brandLabel = new JLabel(text:"Brand:", fontSize:22);
63     brandLabel.setForeground(Color.BLACK);
64     panel.add(brandLabel);
65
66     JTextField brand = new JTextField(fontSize:22);
67     brand.setForeground(Color.BLACK);
68     brand.setBackground(Color.LIGHT_GRAY);
69     panel.add(brand);
70
71     JLabel modelLabel = new JLabel(text:"Model:", fontSize:22);
72     modelLabel.setForeground(Color.BLACK);
73     panel.add(modelLabel);
74
75     JTextField model = new JTextField(fontSize:22);
76     model.setForeground(Color.BLACK);
77     model.setBackground(Color.LIGHT_GRAY);
78     panel.add(model);
79
80     JLabel colorLabel = new JLabel(text:"Color:", fontSize:22);
81     colorLabel.setForeground(Color.BLACK);
82     panel.add(colorLabel);
83
84     JTextField color = new JTextField(fontSize:22);
85     color.setForeground(Color.BLACK);
86     color.setBackground(Color.LIGHT_GRAY);

```

```

88     JLabel yearLabel = new JLabel(text:"Year:", textSize:22);
89     yearLabel.setForeground(Color.BLACK);
90     panel.add(yearLabel);
91
92
93     JTextField year = new JTextField(textSize:22);
94     year.setForeground(Color.BLACK);
95     year.setBackground(Color.LIGHT_GRAY);
96     panel.add(year);
97
98     JLabel pricelabel = new JLabel(text:"Price per Hour:", textSize:22);
99     priceLabel.setForeground(Color.BLACK);
100    panel.add(priceLabel);
101
102    JTextField price = new JTextField(textSize:22);
103    price.setForeground(Color.BLACK);
104    price.setBackground(Color.LIGHT_GRAY);
105    panel.add(price);
106
107    JButton cancel = new JButton(text:"Cancel", textSize:22);
108    cancel.setBackground(Color.RED);
109    cancel.setForeground(Color.WHITE);
110    cancel.addActionListener(new ActionListener() {
111        @Override
112        public void actionPerformed(ActionEvent e) {
113            frame.dispose();
114        }
115    });
116    panel.add(cancel);

```

Panel form dibuat dengan tata letak grid untuk menampung elemen-elemen input seperti merek, model, warna, tahun, dan harga per jam. Kemudian Label dan field input seperti "Brand", "Model", "Color", "Year", dan "Price per Hour" ditambahkan ke panel form untuk meminta informasi unit PS. Lalu Tombol "Cancel" ditambahkan untuk memungkinkan pengguna membatalkan penambahan unit PS.

```

118    JButton confirm = new JButton(text:"Confirm", textSize:22);
119    confirm.setBackground(Color.GREEN);
120    confirm.setForeground(Color.WHITE);
121    confirm.addActionListener(new ActionListener() {
122        @Override
123        public void actionPerformed(ActionEvent e) {
124            if (brand.getText().equals(anObject:"")) {
125                JOptionPane.showMessageDialog(frame, message:"Brand cannot be empty");
126                return;
127            }
128            if (model.getText().equals(anObject:"")) {
129                JOptionPane.showMessageDialog(frame, message:"Model cannot be empty");
130                return;
131            }
132            if (color.getText().equals(anObject:"")) {
133                JOptionPane.showMessageDialog(frame, message:"Color cannot be empty");
134                return;
135            }
136            if (year.getText().equals(anObject:"")) {
137                JOptionPane.showMessageDialog(frame, message:"Year cannot be empty");
138                return;
139            }
140            if (price.getText().equals(anObject:"")) {
141                JOptionPane.showMessageDialog(frame, message:"Price cannot be empty");
142                return;
143            }

```

```

44         int yearInt;
45         double priceDoub;
46         try {
47             yearInt = Integer.parseInt(year.getText());
48         } catch (Exception e1) {
49             JOptionPane.showMessageDialog(frame, message:"Year must be int");
50             return;
51         }
52         try {
53             priceDoub = Double.parseDouble(price.getText());
54         } catch (Exception e1) {
55             JOptionPane.showMessageDialog(frame, message:"Price must be double");
56             return;
57         }
58
59         int available = 0;
60
161     try {
162         ResultSet rs = database.getStatement().executeQuery(sql:"SELECT COUNT(*) FROM `pss`;");
163         rs.next();
164         int ID = rs.getInt(columnLabel:"COUNT(*)");
165
166         String insert = "INSERT INTO `pss`(`ID`, `Brand`, `Model`, `Color`, "
167                     + " `Year`, `Price`, `Available`) VALUES ('" + ID + "','" + brand.getText() + "',"
168                     + "'" + model.getText() + "','" + color.getText() + "','" + yearInt + "','" + priceDoub
169                     + "','""
170                     + "'" + available + "');";
171         database.getStatement().execute(insert);
172         JOptionPane.showMessageDialog(frame, message:"PS unit added successfully");
173         frame.dispose();
174
175     } catch (SQLException er) {
176         JOptionPane.showMessageDialog(frame, er.getMessage());
177     }
178 }
179 });
180 panel.add(confirm);

```

Tombol "Confirm" ditambahkan untuk mengonfirmasi penambahan unit PS. Sebelum unit PS ditambahkan, validasi dilakukan untuk memastikan tidak ada input yang kosong dan tahun serta harga yang dimasukkan sesuai dengan format yang diharapkan.

```

180     panel.add(confirm);
181
182     gradientPanel.add(panel, BorderLayout.CENTER);
183     frame.add(gradientPanel, BorderLayout.CENTER);
184
185     frame.setVisible(b:true);
186 }
187 }

```

Semua komponen UI yang telah dibuat ditambahkan ke frame, lalu frame tersebut ditampilkan kepada pengguna.

3.1.1.3. ChangePassword.java

Kode ChangePassword bertujuan untuk memberikan antarmuka pengguna untuk mengubah kata sandi akun mereka.

```
1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.Font;
6 import java.awt.GridLayout;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9 import java.sql.SQLException;
10
11 import javax.swing.BorderFactory;
12 import javax.swing.JFrame;
13 import javax.swing.JOptionPane;
14 import javax.swing.JPanel;
15 import javax.swing.JButton;
16 import javax.swing.JLabel;
17
18 import Model.Database;
19 import Model.JPasswordField;
20 import Model.Operation;
21 import Model.User;
```

Pada bagian ini, dilakukan import paket yang diperlukan untuk pengembangan aplikasi GUI dengan Java Swing. Selain itu, kelas ChangePassword dideklarasikan dan diimplementasikan dengan menggunakan antarmuka Operation.

```
22
23 public class ChangePassword implements Operation {
24
25     @Override
26     public void operation(Database database, JFrame f, User user) {
27 }
```

Metode operation mengambil objek database, objek JFrame, dan objek User sebagai parameter. Ini adalah metode yang akan dipanggil untuk memulai proses penggantian kata sandi.

```
27
28     JFrame frame = new JFrame(title:"Change Password");
29     frame.setSize(width:600, height:380);
30     frame.setLocationRelativeTo(f);
31     frame.getContentPane().setBackground(new Color(r:236, g:240, b:241));
32     frame.setLayout(new BorderLayout());
33 }
```

Sebuah JFrame baru dibuat dengan judul "Change Password", ukuran 600x380, dan posisi relatif terhadap JFrame induk yang disediakan.

```
33
34     JLabel title = new JLabel(text:"Change Password", JLabel.CENTER);
35     title.setBackground(new Color(r:44, g:62, b:80));
36     title.setFont(new Font(name:"Arial", Font.BOLD, size:30));
37     title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
38     frame.add(title, BorderLayout.NORTH);
39 }
```

Ini adalah bagian yang menampilkan judul "Change Password" di bagian atas jendela. Judul ini ditengahkan dan memiliki warna latar belakang serta gaya font tertentu.

```

39
40 JPanel panel = new JPanel(new GridLayout(rows:4, cols:2, hgap:15, vgap:15));
41 panel.setBackground(new Color(r:236, g:240, b:241));
42 panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
43
44 panel.add(createLabel(text:"Old Password:"));
45
46 JPasswordField oldPassword = new JPasswordField(textSize:22);
47 panel.add(oldPassword);
48
49 panel.add(createLabel(text:"New Password:"));
50
51 JPasswordField newPassword = new JPasswordField(textSize:22);
52 panel.add(newPassword);
53
54 panel.add(createLabel(text:"Confirm Password:"));
55
56 JPasswordField confirmPassword = new JPasswordField(textSize:22);
57 panel.add(confirmPassword);

```

Bagian ini membuat sebuah panel dengan grid layout yang berisi beberapa label dan field password untuk memasukkan password lama, password baru, dan konfirmasi password baru.

```

59 JButton cancel = new JButton(text:"Cancel");
60 cancel.setBackground(new Color(r:192, g:57, b:43));
61 cancel.setForeground(Color.WHITE);
62 cancel.setFont(new Font(name:"Arial", Font.BOLD, size:16));
63 cancel.setFocusPainted(b:false);
64 cancel.addActionListener(new ActionListener() {
65     @Override
66     public void actionPerformed(ActionEvent e) {
67         frame.dispose();
68     }
69 });
70 panel.add(cancel);
71
72 JButton confirm = new JButton(text:"Confirm");
73 confirm.setBackground(new Color(r:39, g:174, b:96));
74 confirm.setForeground(Color.WHITE);
75 confirm.setFont(new Font(name:"Arial", Font.BOLD, size:16));
76 confirm.addActionListener(new ActionListener() {
77     @SuppressWarnings("deprecation")
78     @Override
79     public void actionPerformed(ActionEvent ev) {
80
81         if (oldPassword.getText().equals(anObject:"")) {
82             JOptionPane.showMessageDialog(frame, message:"Old Password cannot be empty");
83             return;
84         }
85         if (newPassword.getText().equals(anObject:"")) {
86             JOptionPane.showMessageDialog(frame, message:"New Password cannot be empty");
87             return;
88         }
89         if (confirmPassword.getText().equals(anObject:"")) {
90             JOptionPane.showMessageDialog(frame, message:"Confirm Password cannot be empty");
91             return;
92         }
93         if (!oldPassword.getText().equals(user.getPassword())) {
94             JOptionPane.showMessageDialog(frame, message:"Incorrect Password");
95             return;
96         }
97         if (!newPassword.getText().equals(confirmPassword.getText())) [
98             JOptionPane.showMessageDialog(frame, message:"Password doesn't match");
99             return;
100        ]
101
102        try {
103            String update = "UPDATE `users` SET "
104                + "Password='" + newPassword.getText() + "' WHERE `ID` = '" + user.getID() + "'";
105            database.getStatement().execute(update);
106            JOptionPane.showMessageDialog(frame, message:"Password changed successfully");
107            user.setPassword(newPassword.getText());
108            frame.dispose();
109        } catch (SQLException e) {
110            JOptionPane.showMessageDialog(frame, e.getMessage());
111        }
112    }
113 });
114 panel.add(confirm);
115

```

Dua tombol, "Cancel" dan "Confirm", ditambahkan ke panel. Aksi untuk masing-masing tombol juga ditentukan: tombol "Cancel" akan menutup jendela, sedangkan tombol "Confirm" akan memvalidasi input pengguna dan mengubah kata sandi pengguna dalam basis data jika inputnya valid. Jika terjadi kesalahan saat mengubah kata sandi dalam basis data, pesan kesalahan akan ditampilkan kepada pengguna melalui dialog.

```
122     private JLabel createLabel(String text) {  
123         JLabel label = new JLabel(text);  
124         label.setForeground(new Color(r:44, g:62, b:80));  
125         label.setFont(new Font(name:"Arial", Font.BOLD, size:18));  
126         return label;  
127     }  
128 }
```

Metode ini digunakan untuk membuat label dengan teks dan properti tertentu untuk digunakan dalam formulir.

```
117     frame.add(panel, BorderLayout.CENTER);  
118     frame.setVisible(b:true);
```

Terakhir, panel dengan formulir password dan tombol ditambahkan ke jendela utama, dan jendela tersebut ditampilkan kepada pengguna.

3.1.1.4. Deleteps.java

Kode Deleteps bertujuan untuk menyediakan antarmuka pengguna yang memungkinkan pengguna untuk menghapus unit PS dari database.

```
1 package Controller;  
2  
3 import java.awt.BorderLayout;  
4 import java.awt.Color;  
5 import java.awt.GradientPaint;  
6 import java.awt.Graphics;  
7 import java.awt.Graphics2D;  
8 import java.awt.GridLayout;  
9 import java.awt.event.ActionEvent;  
10 import java.awt.event.ActionListener;  
11 import java.sql.ResultSet;  
12 import java.sql.SQLException;  
13 import java.util.ArrayList;  
14  
15 import javax.swing.BorderFactory;  
16 import javax.swing.JFrame;  
17 import javax.swing.JOptionPane;  
18 import javax.swing.JPanel;  
19  
20 import Model.ps;  
21 import Model.Database;  
22 import Model.JButton;  
23 import Model.JLabel;  
24 import Model.JTextField;  
25 import Model.JComboBox;  
26 import Model.Operation;  
27 import Model.User;  
28
```

Pada bagian ini, terdapat deklarasi package Controller dan import beberapa kelas yang dibutuhkan untuk membangun antarmuka pengguna (UI) serta berinteraksi dengan basis data.

```
29 public class Deleteps implements Operation {  
30 }
```

Kelas Deleteps mengimplementasikan antarmuka Operation. Ini menunjukkan bahwa kelas ini akan menyediakan fungsionalitas operasi tertentu, yang didefinisikan dalam antarmuka Operation.

```
30  
31     private JTextField brand, model, color, year, price;  
32     private Database database;  
33     private JFrame frame;  
34
```

Kelas ini memiliki beberapa atribut: brand, model, color, year, dan price adalah objek JTextField yang digunakan untuk menampilkan dan memperbarui informasi tentang PS yang dipilih; database adalah objek Database yang digunakan untuk berinteraksi dengan basis data; frame adalah objek JFrame yang mewakili jendela aplikasi.

```
34  
35     @Override  
36     public void operation(Database database, JFrame f, User user) {  
37 }
```

Metode operation() adalah bagian utama dari kelas Deleteps. Metode ini menerima tiga parameter: objek Database, objek JFrame, dan objek User. Objek Database digunakan untuk berinteraksi dengan basis data, JFrame adalah jendela UI di mana operasi ini akan ditampilkan, dan User adalah pengguna yang melakukan operasi.

```
39  
40     frame = new JFrame(title:"Delete PS Unit");  
41     frame.setSize(width:600, height:600);  
42     frame.setLocationRelativeTo(f);  
43     frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);  
44
```

Kode ini membuat jendela baru dengan judul "Delete PS Unit" dan mengatur ukuran serta lokasinya relatif terhadap frame yang diberikan sebagai argumen. Pengaturan setDefaultCloseOperation (JFrame.DISPOSE_ON_CLOSE) menyatakan bahwa ketika jendela ditutup, hanya jendela itu saja yang ditutup, bukan seluruh aplikasi.

```
45     // Panel dengan latar belakang gradasi  
46     JPanel gradientPanel = new JPanel() {  
47         @Override  
48         protected void paintComponent(Graphics g) {  
49             super.paintComponent(g);  
50             Graphics2D g2d = (Graphics2D) g;  
51             int width = getWidth();  
52             int height = getHeight();  
53             Color color1 = new Color(r:224, g:255, b:255);  
54             Color color2 = new Color(r:0, g:0, b:255);  
55             GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, height, color2);  
56             g2d.setPaint(gp);  
57             g2d.fillRect(x:0, y:0, width, height);  
58         }  
59     };  
60     gradientPanel.setLayout(new BorderLayout());  
61
```

Panel ini digunakan sebagai wadah untuk komponen-komponen UI lainnya. Latar belakang panel diisi dengan gradasi warna biru.

```
51  
52     JLabel title = new JLabel(text:"Delete PS Unit", fontSize:35);  
53     title.setForeground(Color.BLUE);  
54     title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));  
55     gradientPanel.add(title, BorderLayout.NORTH);  
56
```

Judul "Delete PS Unit" ditambahkan ke panel dan ditampilkan di bagian atas jendela dengan warna biru.

```

66
67 JPanel panel = new JPanel(new GridLayout(rows:7, cols:2, hgap:15, vgap:15));
68 panel.setBackground(bg:null);
69 panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
70
71 panel.add(new JLabel(text:"ID:", fontSize:22));
72
73 String[] ids = new String[] {" "};
74 ArrayList<Integer> idsArray = new ArrayList<>();
75 try {
76     ResultSet rs0 = database.getStatement()
77         .executeQuery(sql:"SELECT `ID` , `Available` FROM `ps`");
78     while (rs0.next()) {
79         if (rs0.getInt(columnLabel:"Available") < 2)
80             idsArray.add(rs0.getInt(columnLabel:"ID"));
81     }
82 } catch (Exception e0) {
83     JOptionPane.showMessageDialog(frame, e0.getMessage());
84     frame.dispose();
85 }
86
87 ids = new String[idsArray.size() + 1];
88 ids[0] = " ";
89 for (int i = 1; i <= idsArray.size(); i++) {
90     ids[i] = String.valueOf(idsArray.get(i - 1));
91 }
92
93 JComboBox id = new JComboBox(ids, fontSize:22);
94 id.addActionListener(new ActionListener() {
95     @Override
96     public void actionPerformed(ActionEvent e) {
97         updateData(id.getSelectedItem().toString());
98     }
99 });
100 panel.add(id);
101
102 panel.add(new JLabel(text:"Brand:", fontSize:22));
103
104 brand = new JTextField(fontSize:22);
105 brand.setEditable(b:false);
106 panel.add(brand);
107
108 panel.add(new JLabel(text:"Model:", fontSize:22));
109
110 model = new JTextField(fontSize:22);
111 model.setEditable(b:false);
112 panel.add(model);
113
114 panel.add(new JLabel(text:"Color:", fontSize:22));
115
116 color = new JTextField(fontSize:22);
117 color.setEditable(b:false);
118 panel.add(color);
119
120 panel.add(new JLabel(text:"Year:", fontSize:22));
121
122 year = new JTextField(fontSize:22);
123 year.setEditable(b:false);
124 panel.add(year);
125
126 panel.add(new JLabel(text:"Price per Hour:", fontSize:22));
127
128 price = new JTextField(fontSize:22);
129 price.setEditable(b:false);
130 panel.add(price);
131

```

Panel ini berisi sebuah JComboBox yang memuat daftar ID PS yang tersedia untuk dihapus. Pengguna dapat memilih ID dari daftar ini untuk menampilkan informasi terkait PS yang akan dihapus.

```

132 JButton cancel = new JButton(text:"Cancel", textSize:22);
133 cancel.addActionListener(new ActionListener() {
134     @Override
135     public void actionPerformed(ActionEvent e) {
136         frame.dispose();
137     }
138 });
139 panel.add(cancel);
140
141 JButton confirm = new JButton(text:"Confirm", textSize:22);
142 confirm.addActionListener(new ActionListener() {
143     @Override
144     public void actionPerformed(ActionEvent e) {
145         if (id.getSelectedItem().toString().equals(anObject: " ")) {
146             JOptionPane.showMessageDialog(frame, message:"ID cannot be empty");
147             return;
148         }
149
150         try {
151             String update = "UPDATE `pss` SET `Available`='2' WHERE `ID` = '" + id.getSelectedItem().toString()
152             | | + "'";
153             database.getStatement().execute(update);
154             JOptionPane.showMessageDialog(frame, message:"PS deleted successfully");
155             frame.dispose();
156         } catch (SQLException e3) {
157             JOptionPane.showMessageDialog(frame, e3.getMessage());
158         }
159     }
160 });
161 panel.add(confirm);

```

Dua tombol, "Cancel" dan "Confirm", ditambahkan ke panel. Aksi untuk masing-masing tombol juga ditentukan: tombol "Cancel" akan menutup jendela, sedangkan tombol "Confirm" akan menghapus PS dari basis data.

```

169     private void updateData(String ID) {
170         if (ID.equals(anObject: " ")) {
171             brand.setText(t:"");
172             model.setText(t:"");
173             color.setText(t:"");
174             year.setText(t:"");
175             price.setText(t:"");
176         } else {
177             try {
178                 ResultSet rs1 = database.getStatement()
179                 .executeQuery("SELECT * FROM `pss` WHERE `ID` = '" + ID + "'");
180                 rs1.next();
181                 ps ps = new ps();
182                 ps.setID(rs1.getInt(columnLabel:"ID"));
183                 brand.setText(rs1.getString(columnLabel:"Brand"));
184                 model.setText(rs1.getString(columnLabel:"Model"));
185                 color.setText(rs1.getString(columnLabel:"Color"));
186                 year.setText(String.valueOf(rs1.getInt(columnLabel:"Year")));
187                 price.setText(String.valueOf(rs1.getDouble(columnLabel:"Price")));
188             } catch (Exception e1) {
189                 JOptionPane.showMessageDialog(frame, e1.getMessage());
190                 frame.dispose();
191             }
192         }
193     }
194 }

```

Metode ini akan dipanggil setiap kali pengguna memilih ID PS dari JComboBox. Metode ini akan mengambil informasi PS terkait dari basis data dan menampilkannya di JTextField yang sesuai.

```

162
163     gradientPanel.add(panel, BorderLayout.CENTER);
164     frame.add(gradientPanel, BorderLayout.CENTER);
165     frame.setVisible(b:true);
166     frame.requestFocus();
167 }
168

```

Terakhir, panel dengan formulir dan tombol ditambahkan ke jendela utama, dan jendela tersebut ditampilkan kepada pengguna.

3.1.1.5. EditUserData.java

EditUserData merupakan bagian dari sebuah program yang bertujuan untuk mengimplementasikan antarmuka pengguna (UI) yang memungkinkan pengguna untuk mengedit data pengguna (user).

```
1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.Font;
6 import java.awt.GridLayout;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9 import java.sql.SQLException;
10
11 import javax.swing.BorderFactory;
12 import javax.swing.JFrame;
13 import javax.swing.JOptionPane;
14 import javax.swing.JPanel;
15 import javax.swing.JButton;
16 import javax.swing.JLabel;
17
18 import Model.Database;
19 import Model.JTextField;
20 import Model.Operation;
21 import Model.User;
```

Pada bagian ini, terdapat deklarasi package Controller dan import beberapa kelas yang dibutuhkan untuk membangun antarmuka pengguna (UI) serta berinteraksi dengan basis data.

```
23 public class EditUserData implements Operation {
24 }
```

Kelas EditUserData mengimplementasikan antarmuka Operation. Ini menunjukkan bahwa kelas ini akan menyediakan fungsionalitas operasi tertentu, yang didefinisikan dalam antarmuka Operation.

```
24
25     @Override
26     public void operation(Database database, JFrame f, User user) {
27 }
```

Metode operation() adalah bagian utama dari kelas EditUserData. Metode ini menerima tiga parameter: objek Database, objek JFrame, dan objek User. Objek Database digunakan untuk berinteraksi dengan basis data, JFrame adalah jendela UI di mana operasi ini akan ditampilkan, dan User adalah pengguna yang melakukan operasi.

```
27
28     JFrame frame = new JFrame(title:"Edit Data");
29     frame.setSize(width:600, height:450);
30     frame.setLocationRelativeTo(f);
31     frame.getContentPane().setBackground(new Color(r:236, g:240, b:241));
32     frame.setLayout(new BorderLayout());
33 }
```

Kode ini membuat jendela baru dengan judul "Edit Data" dan mengatur ukuran serta lokasinya relatif terhadap frame yang diberikan sebagai argumen. Latar belakang jendela diatur menjadi warna abu-abu muda, dan layout dari konten jendela diatur menggunakan BorderLayout.

```
33
34     JLabel title = new JLabel(text:"Edit Data", JLabel.CENTER);
35     title.setForeground(new Color(r:44, g:62, b:80));
36     title.setFont(new Font(name:"Arial", Font.BOLD, size:30));
37     title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
38     frame.add(title, BorderLayout.NORTH);
39 }
```

Judul "Edit Data" ditampilkan di bagian atas jendela dengan warna teks yang sesuai dan gaya font yang khas.

```
40 JPanel panel = new JPanel(new GridLayout(rows:5, cols:2, hgap:15, vgap:15));
41 panel.setBackground(new Color(r:236, g:240, b:241));
42 panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
43
44 panel.add(createLabel(text:"First Name"));
45
46 JTextField firstName = new JTextField(textSize:22);
47 firstName.setText(user.getFirstName());
48 panel.add(firstName);
49
50 panel.add(createLabel(text:"Last Name"));
51
52 JTextField lastName = new JTextField(textSize:22);
53 lastName.setText(user.getLastName());
54 panel.add(lastName);
55
56 panel.add(createLabel(text:"Email"));
57
58 JTextField email = new JTextField(textSize:22);
59 email.setText(user.getEmail());
60 panel.add(email);
61
62 panel.add(createLabel(text:"Phone Number"));
63
64 JTextField phoneNumber = new JTextField(textSize:22);
65 phoneNumber.setText(user.getPhoneNumber());
66 panel.add(phoneNumber);
```

Panel ini berisi beberapa label dan field input untuk memasukkan informasi pengguna seperti nama depan, nama belakang, email, dan nomor telepon. Nilai default dari objek User yang diberikan sebagai argumen digunakan untuk menginisialisasi nilai awal dari field input.

```
68 JButton cancel = new JButton(text:"Cancel");
69 cancel.setBackground(new Color(r:192, g:57, b:43));
70 cancel.setForeground(Color.WHITE);
71 cancel.setFont(new Font(name:"Arial", Font.BOLD, size:16));
72 cancel.addActionListener(new ActionListener() {
73     @Override
74     public void actionPerformed(ActionEvent e) {
75         frame.dispose();
76     }
77 });
78 panel.add(cancel);
79
80 JButton confirm = new JButton(text:"Confirm");
81 confirm.setBackground(new Color(r:39, g:174, b:96));
82 confirm.setForeground(Color.WHITE);
83 confirm.setFont(new Font(name:"Arial", Font.BOLD, size:16));
84 confirm.setFocusPainted(b:false);
85 confirm.addActionListener(new ActionListener() {
86     @Override
87     public void actionPerformed(ActionEvent ev) {
88
89         if (firstName.getText().equals(anObject:"")) {
90             JOptionPane.showMessageDialog(frame, message:"First Name cannot be empty");
91             return;
92         }
93         if (lastName.getText().equals(anObject:"")) {
94             JOptionPane.showMessageDialog(frame, message:"Last Name cannot be empty");
95             return;
96         }
97     }
98 });
99 panel.add(confirm);
100
```

```

97     if (email.getText().equals(anObject:"")) {
98         JOptionPane.showMessageDialog(frame, message:"Email cannot be empty");
99         return;
100    }
101    if (phoneNumber.getText().equals(anObject:"")) {
102        JOptionPane.showMessageDialog(frame, message:"Phone Number cannot be empty");
103        return;
104    }
105
106    String update = "UPDATE `users` SET `FirstName`='"
107        + firstName.getText() + "','" + `LastName`=''
108        + lastName.getText() + "','" + `Email`=''
109        + email.getText() + "','" + `PhoneNumber`=''
110        + phoneNumber.getText() + "','" +
111        "WHERE `ID` = '" + user.getID() + "'"; 
112
113    try {
114        database.getStatement().execute(update);
115        JOptionPane.showMessageDialog(frame, message:"Data updated successfully");
116        user.setFirstName(firstName.getText());
117        user.setLastName(lastName.getText());
118        user.setEmail(email.getText());
119        user.setPhoneNumber(phoneNumber.getText());
120        frame.dispose();
121    } catch (SQLException e) {
122        JOptionPane.showMessageDialog(frame, e.getMessage());
123    }
124 });
panel.add(confirm);

```

Dua tombol, "Cancel" dan "Confirm", ditambahkan ke panel. Aksi untuk masing-masing tombol juga ditentukan: tombol "Cancel" akan menutup jendela, sedangkan tombol "Confirm" akan memperbarui data pengguna dalam basis data.

```

130
131     private JLabel createLabel(String text) {
132         JLabel label = new JLabel(text);
133         label.setForeground(new Color(r:44, g:62, b:80));
134         label.setFont(new Font(name:"Arial", Font.BOLD, size:18));
135         return label;
136     }
137 }
138

```

Metode ini digunakan untuk membuat label dengan teks dan properti tertentu untuk digunakan dalam formulir.

```

124     panel.add(confirm);
125
126     frame.add(panel, BorderLayout.CENTER);
127     frame.setVisible(b:true);
128

```

Terakhir, panel dengan formulir dan tombol ditambahkan ke jendela utama, dan jendela tersebut ditampilkan kepada pengguna.

3.1.1.6. Main.java

```
1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.GradientPaint;
6 import java.awt.Graphics;
7 import java.awt.Graphics2D;
8 import java.awt.GridLayout;
9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14
15 import javax.swing.BorderFactory;
16 import javax.swing.ImageIcon;
17 import javax.swing.JFrame;
18 import javax.swing.JOptionPane;
19 import javax.swing.JPanel;
20 import javax.swing.JButton;
21 import javax.swing.JLabel;
22
23 import Model.Admin;
24 import Model.Client;
25 import Model.Database;
26 import Model.JPasswordField;
27 import Model.JTextField;
28 import Model.User;
```



```
30 public class Main {
31
32     private static Database database;
33
34     Run main | Debug main | Run | Debug
35     public static void main(String[] args) {
36         database = new Database();
37         start();
38     }
39
40     public static void start() {
41         JFrame frame = new JFrame(title:"Login");
42         frame.setSize(width:800, height:330); // Sesuaikan ukuran frame untuk memberikan ruang bagi gambar
43         frame.setLocationRelativeTo(null);
44         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45         frame.setLayout(new BorderLayout());
```

Kelas Main adalah entry point dari aplikasi. Dalam method main, objek Database diinisialisasi dan method start dipanggil untuk memulai aplikasi.

```
46     JLabel title = new JLabel(text:"WELCOME TO RENTAL PS GDA", JLabel.CENTER);
47     title.setFont(title.getFont().deriveFont(size:35f));
48     title.setForeground(Color.BLUE);
49     title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
50     frame.add(title, BorderLayout.NORTH);
```

Method start() menginisialisasi jendela utama aplikasi dengan judul "Login", ukuran yang sesuai, dan pengaturan layout BorderLayout.

```
51 }
```

Label judul "WELCOME TO RENTAL PS GDA" ditampilkan di bagian atas jendela dengan font dan warna yang sesuai.

```

51
52     JPanel gradientPanel = new JPanel() {
53         @Override
54             protected void paintComponent(Graphics g) {
55                 super.paintComponent(g);
56                 Graphics2D g2d = (Graphics2D) g;
57                 int width = getWidth();
58                 int height = getHeight();
59                 Color color1 = new Color(r:224, g:255, b:255);
60                 Color color2 = new Color(r:0, g:0, b:255);
61                 GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, height, color2);
62                 g2d.setPaint(gp);
63                 g2d.fillRect(x:0, y:0, width, height);
64             }
65         };
66     gradientPanel.setLayout(new BorderLayout());

```

Panel ini memberikan efek gradasi sebagai latar belakang dengan menggunakan kelas Graphics2D untuk melukis gradasi dari color1 ke color2.

```

67
68     JPanel panel = new JPanel(new GridLayout(rows:3, cols:2, hgap:15, vgap:15));
69     panel.setOpaque(isOpaque:false);
70     panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
71
72     addStyledLabel(panel, text:"Email:");
73
74     JTextField email = new JTextField(textSize:22);
75     email.setForeground(Color.BLACK);
76     email.setBackground(Color.LIGHT_GRAY);
77     panel.add(email);
78
79     addStyledLabel(panel, text:"Password:");
80
81     JPasswordField password = new JPasswordField(textSize:22);
82     password.setForeground(Color.BLACK);
83     password.setBackground(Color.LIGHT_GRAY);
84     panel.add(password);
85

```

Panel ini berisi field input untuk email dan password pengguna. Setiap label dan field input ditambahkan ke panel dengan pengaturan grid layout.

```

85
86     JButton createAcc = createStyledButton(text:"Create New Account", Color.GREEN, Color.WHITE);
87     createAcc.addActionListener(new ActionListener() {
88         @Override
89             public void actionPerformed(ActionEvent e) {
90                 new AddNewAccount(accType:0).operation(database, frame, u:null);
91                 frame.dispose();
92             }
93         });
94     panel.add(createAcc);
95

```

Tombol ini memungkinkan pengguna untuk membuat akun baru. Ketika tombol ditekan, method operation dari kelas AddNewAccount dipanggil dan jendela login ditutup.

```

96     ArrayList<User> users = new ArrayList<>();
97     try {
98         String select = "SELECT * FROM `users`";
99         ResultSet rs = database.getStatement().executeQuery(select);
100        while (rs.next()) {
101            User user;
102            int ID = rs.getInt(columnLabel:"ID");
103            String firstName = rs.getString(columnLabel:"FirstName");
104            String lastName = rs.getString(columnLabel:"LastName");
105            String em = rs.getString(columnLabel:"Email");
106            String phoneNumber = rs.getString(columnLabel:"PhoneNumber");
107            String pass = rs.getString(columnLabel:"Password");
108            int type = rs.getInt(columnLabel:"Type");
109
110            if (type == 0) {
111                user = new Client();
112                user.setID(ID);
113                user.setFirstName(firstName);
114                user.setLastName(lastName);
115                user.setEmail(em);
116                user.setPhoneNumber(phoneNumber);
117                user.setPassword(pass);
118                users.add(user);
119
120            } else if (type == 1) {
121                user = new Admin();
122                user.setID(ID);
123                user.setFirstName(firstName);
124                user.setLastName(lastName);
125                user.setEmail(em);
126                user.setPhoneNumber(phoneNumber);
127                user.setPassword(pass);
128                users.add(user);
129            }
130        } catch (SQLException e) {
131            e.printStackTrace();
132        }
    
```

Bagian ini memuat semua pengguna dari basis data ke dalam daftar users. Setiap pengguna dikelompokkan sebagai Client atau Admin berdasarkan tipe mereka.

```

134     JButton login = createStyledButton(text:"Login", Color.BLUE, Color.WHITE);
135     login.addActionListener(new ActionListener() {
136         @SuppressWarnings("deprecation")
137         @Override
138         public void actionPerformed(ActionEvent e) {
139             if (email.getText().equals(anObject:"")) {
140                 JOptionPane.showMessageDialog(frame, message:"Email cannot be empty");
141                 return;
142             }
143
144             if (password.getText().equals(anObject:"")) {
145                 JOptionPane.showMessageDialog(frame, message:"Password cannot be empty");
146                 return;
147             }
148
149             boolean loggedIn = false;
150             for (User u : users) {
151                 if (u.getEmail().equals(email.getText()) && u.getPassword().equals(password.getText())) {
152                     loggedIn = true;
153                     u.showList(database, frame);
154                     frame.dispose();
155                 }
156             }
157             if (!loggedIn) {
158                 JOptionPane.showMessageDialog(frame, message:"Email or password doesn't match");
159             }
160         });
161     panel.add(login);
162 }
    
```

Tombol login memeriksa input pengguna dan membandingkannya dengan data pengguna yang dimuat dari basis data. Jika kecocokan ditemukan, pengguna berhasil login dan jendela login ditutup.

```

163
164 // Membuat panel utama untuk menampung panel input dan gambar
165 JPanel mainPanel = new JPanel(new BorderLayout());
166 mainPanel.setOpaque(isOpaque:false);
167 mainPanel.add(panel, BorderLayout.CENTER);
168
169 // Memuat gambar dan menambahkannya ke label
170 ClassLoader classLoader = Main.class.getClassLoader();
171 ImageIcon imageIcon = new ImageIcon(classLoader.getResource(name:"resources/lumba.png"));
172 javax.swing.JLabel jLabel = new javax.swing.JLabel(imageIcon);
173
174 // Menambahkan label gambar ke panel utama di sisi kanan
175 mainPanel.add(jLabel, BorderLayout.EAST);
176
177 // Menambahkan panel utama ke panel gradasi
178 gradientPanel.add(mainPanel, BorderLayout.CENTER);
179
180 // Menambahkan panel gradasi ke frame
181 frame.add(gradientPanel, BorderLayout.CENTER);
182
183 frame.setVisible(b:true);
184 }
185

```

Bagian ini membuat panel utama yang berisi panel input dan gambar di sebelah kanan. Gambar dimuat dari resources dan ditambahkan ke label. Seluruh panel utama kemudian ditambahkan ke panel gradasi dan akhirnya ke frame utama.

```

185
186     private static void addStyledLabel(JPanel panel, String text) {
187         JLabel label = new JLabel(text);
188         label.setFont(label.getFont().deriveFont(size:18f));
189         label.setForeground(new Color(r:44, g:62, b:80));
190         label.setBorder(BorderFactory.createEmptyBorder(top:0, left:0, bottom:5, right:0));
191         panel.add(label);
192     }
193
194     private static JButton createStyledButton(String text, Color background, Color foreground) {
195         JButton button = new JButton(text);
196         button.setFont(button.getFont().deriveFont(size:22f));
197         button.setBackground(background);
198         button.setForeground(foreground);
199         button.setFocusPainted(b:false);
200         button.setBorder(BorderFactory.createEmptyBorder(top:10, left:20, bottom:10, right:20));
201         return button;
202     }
203 }

```

Method addStyledLabel dan createStyledButton digunakan untuk membuat label dan tombol dengan gaya tertentu untuk konsistensi UI.

3.1.1.7. Quit.java

Kelas Quit yang mengimplementasikan antarmuka Operation. Kelas ini berfungsi untuk mengakhiri program dengan memberikan pesan terima kasih kepada pengguna dan menutup input dari pengguna (scanner).

```

1 package Controller;
2
3 import java.util.Scanner;
4
5 import Model.Database;
6 import Model.Operation;
7 import Model.User;
8
9 public class Quit implements Operation {
10

```

Bagian ini mendeklarasikan package Controller dan mengimpor kelas Scanner dari package java.util. Selain itu, kelas Database, Operation, dan User diimporkan dari package Model. Kelas Quit dideklarasikan sebagai public dan mengimplementasikan antarmuka Operation.

```
10
11     @Override
12     public void operation(Database database, Scanner s, User user) {
13
14         System.out.println("Thanks for visiting us!");
15         s.close();
16
17     }
18
19 }
```

Method operation adalah satu-satunya method yang didefinisikan dalam kelas ini. Method ini mengambil tiga parameter:

- ❖ Database database: Objek Database yang mungkin diperlukan untuk operasi terkait basis data (tidak digunakan dalam method ini).
- ❖ Scanner s: Objek Scanner yang digunakan untuk membaca input dari pengguna.
- ❖ User user: Objek User yang mewakili pengguna saat ini (tidak digunakan dalam method ini).

3.1.1.8. Rentps.java

Rentps yang mengimplementasikan antarmuka Operation. Kode ini memungkinkan pengguna untuk melakukan booking PlayStation (PS) dan menampilkan informasi terkait PS yang dipilih.

```
1  package Controller;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.GridLayout;
6  import java.awt.event.ActionEvent;
7  import java.awt.event.ActionListener;
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10 import java.util.ArrayList;
11
12 import javax.swing.BorderFactory;
13 import javax.swing.JButton;
14 import javax.swing.JFrame;
15 import javax.swing.JLabel;
16 import javax.swing.JOptionPane;
17 import javax.swing.JPanel;
18 import javax.swing.JComboBox;
19
20 import Model.Database;
21 import Model.Operation;
22 import Model.Rent;
23 import Model.User;
24 import Model.ps;
25 import Model.JTextField;
26
27 public class Rentps implements Operation {
28
29     private JTextField brand, model, color, year, price;
30     private Database database;
31     private JFrame frame;
```

Kelas Rentps mengimplementasikan antarmuka Operation. Beberapa variabel digunakan untuk menyimpan referensi ke elemen UI dan objek database.

```

33     @Override
34     public void operation(Database database, JFrame f, User user) {
35
36         this.database = database;
37
38         frame = new JFrame(title:"Booking PS");
39         frame.setSize(width:600, height:650);
40         frame.setLocationRelativeTo(f);
41         frame.getContentPane().setBackground(new Color(r:236, g:240, b:241));
42         frame.setLayout(new BorderLayout());
43

```

Method ini digunakan untuk melakukan operasi booking PS. Kode ini dimulai dengan inisialisasi variabel dan pengaturan tampilan frame.

```

43     JPanel header = new JPanel(new BorderLayout());
44     header.setBackground(new Color(r:44, g:62, b:80));
45     JLabel title = new JLabel(text:"Booking PS", JLabel.CENTER);
46     title.setFont(title.getFont().deriveFont(size:35f));
47     title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:20, right:0));
48     frame.add(title, BorderLayout.NORTH);
49
50

```

Bagian ini mengatur header dengan judul "Booking PS".

```

50
51     JPanel panel = new JPanel(new GridLayout(rows:8, cols:2, hgap:15, vgap:15));
52     panel.setBackground(bg:null);
53     panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
54

```

Panel utama diatur dengan layout grid untuk menampilkan elemen input.

```

55     addStyledLabel(panel, text:"ID:");
56
57     String[] ids = new String[] { " " };
58     ArrayList<Integer> idsArray = new ArrayList<>();
59
60     try {
61         ResultSet rs0 = database.getStatement()
62             .executeQuery(sql:"SELECT `ID`, `Available` FROM `ps`;");
63         while (rs0.next()) {
64             if (rs0.getInt(columnLabel:"Available") < 2)
65                 idsArray.add(rs0.getInt(columnLabel:"ID"));
66         }
67     } catch (Exception e0) {
68         JOptionPane.showMessageDialog(frame, e0.getMessage());
69         frame.dispose();
70     }
71
72     ids = new String[idsArray.size() + 1];
73     ids[0] = " ";
74     for (int i = 1; i <= idsArray.size(); i++) {
75         ids[i] = String.valueOf(idsArray.get(i - 1));
76     }
77
78     JComboBox<String> id = new JComboBox<>(ids);
79     id.addActionListener(new ActionListener() {
80         @Override
81         public void actionPerformed(ActionEvent e) {
82             updateData(id.getSelectedItem().toString());
83         }
84     });
85     panel.add(id);

```

Bagian ini menambahkan label untuk ID dan combobox yang menampilkan daftar ID PS yang tersedia. ID diambil dari database dan disimpan dalam idsArray.

```

86     addStyledLabel(panel, text:"Brand:");
87
88     brand = new JTextField(textSize:22);
89     panel.add(brand);
90
91     addStyledLabel(panel, text:"Model:");
92
93     model = new JTextField(textSize:22);
94     panel.add(model);
95
96     addStyledLabel(panel, text:"Color:");
97
98     color = new JTextField(textSize:22);
99     panel.add(color);
100
101    addStyledLabel(panel, text:"Year:");
102
103    year = new JTextField(textSize:22);
104    panel.add(year);
105
106    addStyledLabel(panel, text:"Price per Hour:");
107
108    price = new JTextField(textSize:22);
109    panel.add(price);
110
111    addStyledLabel(panel, text:"Hours:");
112
113    JTextField hours = new JTextField(textSize:22);
114    panel.add(hours);

```

Berbagai input fields ditambahkan untuk mengumpulkan informasi tentang PS dan waktu sewa.

```

116 JButton showps = new JButton(text:"Cek PS");
117 showps.addActionListener(new ActionListener() {
118     @Override
119     public void actionPerformed(ActionEvent e) {
120         new Viewps().operation(database, frame, user);
121     }
122 });
123 panel.add(showps);
124
125 JButton confirm = new JButton(text:"Confirm");
126 confirm.addActionListener(new ActionListener() {
127     @Override
128     public void actionPerformed(ActionEvent e) {
129         if (id.getSelectedItem().toString().equals(anObject:" ")) {
130             JOptionPane.showMessageDialog(frame, message:"ID cannot be empty");
131             return;
132         }
133         if (hours.getText().equals(anObject:"")) {
134             JOptionPane.showMessageDialog(frame, message:"Hours cannot be empty");
135             return;
136         }
137         int hoursInt;
138         try {
139             hoursInt = Integer.parseInt(hours.getText());
140         } catch (Exception e5) {
141             JOptionPane.showMessageDialog(frame, message:"Hours must be int");
142             return;
143         }

```

```

144 |     try {
145 |         ResultSet rs0 = database.getStatement()
146 |             .executeQuery(
147 |                 "SELECT * FROM `ps` WHERE `ID` = '" + id.getSelectedItem().toString() + "';");
148 |         rs0.next();
149 |         ps ps = new ps();
150 |         ps.setID(rs0.getInt(columnLabel:"ID"));
151 |         ps.setBrand(rs0.getString(columnLabel:"Brand"));
152 |         ps.setModel(rs0.getString(columnLabel:"Model"));
153 |         ps.setColor(rs0.getString(columnLabel:"Color"));
154 |         ps.setYear(rs0.getInt(columnLabel:"Year"));
155 |         ps.setPrice(rs0.getDouble(columnLabel:"Price"));
156 |         ps.setAvailable(rs0.getInt(columnLabel:"Available"));
157 |
158 |         if (ps.isAvailable() != 0) {
159 |             JOptionPane.showMessageDialog(frame, message:"PS isn't available");
160 |             return;
161 |         }
162 |
163 |         ResultSet rs1 = database.getStatement()
164 |             .executeQuery(sql:"SELECT COUNT(*) FROM `rents`");
165 |         rs1.next();
166 |         int ID = rs1.getInt(columnLabel:"COUNT(*)");
167 |
168 |         double total = ps.getPrice() * hoursInt;
169 |
170 |         Rent rent = new Rent();
171 |
172 |         String insert = "INSERT INTO `rents`(`ID`, `User`, `ps`, `DateTime`, `Hours`, "
173 |             + " `Total`, `Status`) VALUES ('" + ID + "','" + user.getID() + "',"
174 |             + "'" + ps.getID() + "','" + rent.getDateTime() + "','" + hoursInt + "',"
175 |             + "'" + total + "','0')";
176 |
177 |         database.getStatement().execute(insert);
178 |         JOptionPane.showMessageDialog(frame, "Selamat bermain"
179 |             + "\nTotal bayar = " + "Rp" + total);
180 |         frame.dispose();
181 |     } catch (SQLException exception) {
182 |         JOptionPane.showMessageDialog(frame, exception.getMessage());
183 |     }
184 |
185 | };
186 | panel.add(confirm);
187 |

```

Bagian ini menambahkan dua tombol: Cek PS dan Confirm. Tombol Cek PS membuka tampilan daftar PS, sedangkan tombol Confirm melakukan validasi input, memeriksa ketersediaan PS, dan menambahkan entri sewa ke database.

```

187 |
188 |         frame.add(panel, BorderLayout.CENTER);
189 |         frame.setVisible(b:true);
190 |         frame.requestFocus();
191 |

```

Panel utama ditambahkan ke frame, dan frame ditampilkan.

```

192 |
193 |     private void addStyledLabel(JPanel panel, String text) {
194 |         JLabel label = new JLabel(text);
195 |         label.setFont(label.getFont().deriveFont(size:18f));
196 |         label.setForeground(new Color(r:44, g:62, b:80));
197 |         label.setBorder(BorderFactory.createEmptyBorder(top:0, left:0, bottom:5, right:0));
198 |
199 |     }
200 |

```

Method ini menambahkan label dengan gaya khusus ke panel.

```

200
201     private void updateData(String ID) {
202         if (ID.equals(anObject: " ")) {
203             brand.setText(t: "");
204             model.setText(t: "");
205             color.setText(t: "");
206             year.setText(t: "");
207             price.setText(t: "");
208         } else {
209             try {
210                 ResultSet rs1 = database.getStatement()
211                     .executeQuery("SELECT * FROM `pss` WHERE `ID` = '" + ID + "'");
212                 rs1.next();
213                 ps ps = new ps();
214                 ps.setInt(rs1.getInt(columnLabel:"ID"));
215                 brand.setText(rs1.getString(columnLabel:"Brand"));
216                 model.setText(rs1.getString(columnLabel:"Model"));
217                 color.setText(rs1.getString(columnLabel:"Color"));
218                 year.setText(String.valueOf(rs1.getInt(columnLabel:"Year")));
219                 price.setText("Rp" + String.valueOf(rs1.getDouble(columnLabel:"Price")));
220             } catch (Exception e1) {
221                 JOptionPane.showMessageDialog(frame, e1.getMessage());
222                 frame.dispose();
223             }
224         }
225     }
226 }
```

Method ini memperbarui data input fields berdasarkan ID PS yang dipilih dari combobox.

3.1.1.9. Returnps.java

Kode Returps memungkinkan pengguna untuk mengembalikan PlayStation (PS) yang telah disewa dan menyelesaikan transaksi sewa.

```

1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.GridLayout;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.sql.ResultSet;
9 import java.sql.SQLException;
10 import java.util.ArrayList;
11
12 import javax.swing.BorderFactory;
13 import javax.swing.JFrame;
14 import javax.swing.JOptionPane;
15 import javax.swing.JPanel;
16
17 import Model.Database;
18 import Model.JButton;
19 import Model.JComboBox;
20 import Model.JLabel;
21 import Model.Operation;
22 import Model.Rent;
23 import Model.User;
24
25 public class Returnps implements Operation {
26 }
```

Kelas Returnps mengimplementasikan antarmuka Operation. Kelas ini menyediakan method untuk mengembalikan PS yang telah disewa.

```

26
27     @Override
28     public void operation(Database database, JFrame f, User user) {
29
30         JFrame frame = new JFrame(title:"Finish Play");
31         frame.setSize(width:600, height:260);
32         frame.setLocationRelativeTo(f);
33         frame.getContentPane().setBackground(new Color(r:236, g:240, b:241));
34         frame.setLayout(new BorderLayout());
35
36         JLabel title = new JLabel(text:"Rental PS GDA", fontSize:35);
37         title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
38         frame.add(title, BorderLayout.NORTH);
39 }
```

Method operation digunakan untuk melakukan operasi pengembalian PS. Parameter yang digunakan adalah objek Database, JFrame yang mewakili frame utama, dan User yang mewakili pengguna yang sedang aktif. Frame baru dibuat dengan judul "Finish Play", ukuran, lokasi, dan warna latar belakang yang diatur. Kemudian label judul ditambahkan ke frame bagian atas.

```

39 | JPanel panel = new JPanel(new GridLayout(rows:2, cols:2, hgap:15, vgap:15));
40 | panel.setBackground(bg:null);
41 | panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
42 |
43 | panel.add(new JLabel(text:"Rent ID:", font:font));
44 |
45 | String[] ids = new String[1] { " " };
46 | ArrayList<Integer> idsArray = new ArrayList<>();
47 | try {
48 |     ResultSet rs0 = database.getStatement()
49 |         .executeQuery("SELECT `ID` FROM `rents` WHERE `User` = '" + user.getID() + "'");
50 |     while (rs0.next()) {
51 |         idsArray.add(rs0.getInt(columnLabel:"ID"));
52 |     }
53 | } catch (Exception e0) {
54 |     JOptionPane.showMessageDialog(frame, e0.getMessage());
55 |     frame.dispose();
56 | }
57 |
58 | ids = new String[idsArray.size() + 1];
59 | ids[0] = " ";
60 | for (int i = 1; i <= idsArray.size(); i++) {
61 |     ids[i] = String.valueOf(idsArray.get(i - 1));
62 | }
63 |
64 | JComboBox id = new JComboBox(ids, font:font);
65 | panel.add(id);
66 |
67 |

```

Panel utama diatur dengan layout grid untuk menampilkan elemen input. Selanjutnya menambahkan label untuk ID sewa dan combobox yang menampilkan daftar ID sewa yang aktif untuk pengguna saat ini. ID diambil dari database dan disimpan dalam idsArray.

```

68 | JButton showRents = new JButton(text:"Show my Rents", font:font);
69 | showRents.addActionListener(new ActionListener() {
70 |     @Override
71 |     public void actionPerformed(ActionEvent e) {
72 |         new ShowUserRents(user.getID()).operation(database, frame, user);
73 |     }
74 | });
75 | panel.add(showRents);
76 |
77 | JButton confirm = new JButton(text:"Confirm", font:font);
78 | confirm.addActionListener(new ActionListener() {
79 |     @Override
80 |     public void actionPerformed(ActionEvent e) {
81 |         if (id.getSelectedItem().toString().equals(anObject: " ")) {
82 |             JOptionPane.showMessageDialog(frame, message:"Rent ID cannot be empty");
83 |             return;
84 |         }
85 |
86 |         try {
87 |             String select = "SELECT * FROM `rents` WHERE `ID` = '" + id.getSelectedItem().toString() + "'";
88 |             ResultSet rs = database.getStatement().executeQuery(select);
89 |             rs.next();
90 |             Rent r = new Rent();
91 |             r.setID(rs.getInt(columnLabel:"ID"));
92 |             r.setUser(user);
93 |             r.setDateTime(rs.getString(columnLabel:"DateTime"));
94 |             r.setHours(rs.getInt(columnLabel:"Hours"));
95 |             r.setTotal(rs.getDouble(columnLabel:"Total"));
96 |             r.setStatus(rs.getInt(columnLabel:"Status"));

```

```

98     if (r.getStatusToString().equals(anObject:"Delayed")) {
99         JOptionPane.showMessageDialog(frame, r.getDelayedHours()
100            + " delayed hours\nYou will have to pay Rp1000 as fine");
101    }
102
103    String update = "UPDATE `rents` SET `Status`='1' WHERE `ID` = '" + id.getSelectedItem().toString()
104        + "'";
105    database.createStatement().execute(update);
106    JOptionPane.showMessageDialog(frame, message:"ps returned successfully");
107    frame.dispose();
108 } catch (SQLException exception) {
109     JOptionPane.showMessageDialog(frame, exception.getMessage());
110 }
111
112 };
113 panel.add(confirm);
114

```

Tombol Show my Rents membuka tampilan daftar sewa yang dimiliki oleh pengguna saat ini. Tombol Confirm memvalidasi input, memeriksa apakah ID sewa dipilih, dan memperbarui status sewa di database. Jika sewa terlambat, pesan peringatan akan ditampilkan.

3.1.1.10. ShowAllRents.java

ShowAllRents adalah implementasi antarmuka Operation yang memungkinkan pengguna untuk melihat semua data sewa PS melalui antarmuka grafis (GUI). Kode ini mengatur frame dan panel dengan latar belakang gradasi, mengambil data sewa dari database, dan menampilkannya dalam tabel. Setiap baris dalam tabel menampilkan informasi detail tentang setiap sewa, termasuk informasi pengguna dan PS yang disewa.

```

1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.GradientPaint;
6 import java.awt.Graphics;
7 import java.awt.Graphics2D;
8 import java.sql.ResultSet;
9 import java.sql.SQLException;
10 import java.util.ArrayList;
11
12 import javax.swing.BorderFactory;
13 import javax.swing.JFrame;
14 import javax.swing.JOptionPane;
15 import javax.swing.JPanel;
16 import javax.swing.JScrollPane;
17
18 import Model.ps;
19 import Model.Client;
20 import Model.Database;
21 import Model.JLabel;
22 import Model.JTable;
23 import Model.Operation;
24 import Model.Rent;
25 import Model.User;
26
27 public class ShowAllRents implements Operation {
28

```

Kelas ShowAllRents mengimplementasikan antarmuka Operation. Kelas ini menyediakan method untuk menampilkan semua data sewa.

```

29     @Override
30     public void operation(Database database, JFrame f, User user) {
31         JFrame frame = new JFrame(title:"Rents");
32         frame.setSize(width:1200, height:600);
33         frame.setLocationRelativeTo(f);
34         frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
35
36         // Panel dengan latar belakang gradasi
37         JPanel gradientPanel = new JPanel() {
38             @Override
39             protected void paintComponent(Graphics g) {
40                 super.paintComponent(g);
41                 Graphics2D g2d = (Graphics2D) g;
42                 int width = getWidth();
43                 int height = getHeight();
44                 Color color1 = new Color(r:224, g:255, b:255);
45                 Color color2 = new Color(r:0, g:0, b:255);
46                 GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, height, color2);
47                 g2d.setPaint(gp);
48                 g2d.fillRect(x:0, y:0, width, height);
49             }
50         };
51         gradientPanel.setLayout(new BorderLayout());

```

Method operation digunakan untuk melakukan operasi penampilan data sewa. Parameter yang digunakan adalah objek Database, JFrame yang mewakili frame utama, dan User yang mewakili pengguna yang sedang aktif. Frame baru dibuat dengan judul "Rents", ukuran, lokasi, dan default close operation yang diatur. Panel dengan latar belakang gradasi dibuat menggunakan override method paintComponent. Panel ini diatur dengan layout BorderLayout.

```

52     JLabel title = new JLabel(text:"Rents", fontSize:35);
53     title.setForeground(Color.BLUE);
54     title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
55     gradientPanel.add(title, BorderLayout.NORTH);
56
57     String[] header = {
58         "ID", "Name", "Email", "Tel", "ps ID", "ps", "Date Time",
59         "Hours", "Total", "Status"
60     };
61
62     ArrayList<Rent> rents = new ArrayList<>();
63     ArrayList<Integer> psIDs = new ArrayList<>();
64     ArrayList<Integer> userIDs = new ArrayList<>();
65
66     try {
67         String select = "SELECT * FROM `rents`";
68         ResultSet rs = database.getStatement().executeQuery(select);
69         while (rs.next()) {
70             Rent rent = new Rent();
71             rent.setID(rs.getInt(columnLabel:"ID"));
72             userIDs.add(rs.getInt(columnLabel:"User"));
73             psIDs.add(rs.getInt(columnLabel:"PS"));
74             rent.setDateTime(rs.getString(columnLabel:"DateTime"));
75             rent.setHours(rs.getInt(columnLabel:"Hours"));
76             rent.setTotal(rs.getDouble(columnLabel:"Total"));
77             rent.setStatus(rs.getInt(columnLabel:"Status"));
78             rents.add(rent);
79         }
80     }

```

```

80
81     for (int j = 0; j < rents.size(); j++) {
82         Rent r = rents.get(j);
83
84         String selectUser = "SELECT * FROM `users` WHERE `ID` = '" + userIDs.get(j) + "'";
85         ResultSet rs2 = database.getStatement().executeQuery(selectUser);
86         rs2.next();
87         User u = new Client();
88         u.setID(rs2.getInt(columnLabel:"ID"));
89         u.setFirstName(rs2.getString(columnLabel:"FirstName"));
90         u.setLastName(rs2.getString(columnLabel:"LastName"));
91         u.setEmail(rs2.getString(columnLabel:"Email"));
92         u.setPhoneNumber(rs2.getString(columnLabel:"PhoneNumber"));
93         u.setPassword(rs2.getString(columnLabel:"Password"));
94         r.setUser(u);
95
96         ResultSet rs3 = database.getStatement()
97             .executeQuery("SELECT * FROM `pss` WHERE `ID` = '" + psIDs.get(j) + "'");
98         rs3.next();
99         ps ps = new ps();
100        ps.setID(rs3.getInt(columnLabel:"ID"));
101        ps.setBrand(rs3.getString(columnLabel:"Brand"));
102        ps.setModel(rs3.getString(columnLabel:"Model"));
103        ps.setColor(rs3.getString(columnLabel:"Color"));
104        ps.setYear(rs3.getInt(columnLabel:"Year"));
105        ps.setPrice(rs3.getDouble(columnLabel:"Price"));
106        ps.setAvailable(rs3.getInt(columnLabel:"Available"));
107        r.setps(ps);
108    }
109
110 } catch (SQLException e) {
111     JOptionPane.showMessageDialog(frame, e.getMessage());
112     frame.dispose();
113 }
```

Label judul ditambahkan ke panel bagian atas dengan beberapa properti yang diatur, termasuk warna teks dan border. Bagian ini mengambil data sewa dari database dan menyimpannya dalam daftar rents. Selain itu, data PS dan pengguna juga diambil berdasarkan ID masing-masing dari database.

```

115 String[][] rentsData = new String[rents.size()][10];
116 for (int j = 0; j < rents.size(); j++) {
117     Rent r = rents.get(j);
118     rentsData[j][0] = String.valueOf(r.getID());
119     rentsData[j][1] = r.getUser().getfirstName() + " " + r.getUser().getLastName();
120     rentsData[j][2] = r.getUser().getEmail();
121     rentsData[j][3] = r.getUser().getPhoneNumber();
122     rentsData[j][4] = String.valueOf(r.getps().getID());
123     rentsData[j][5] = r.getps().getBrand() + " " + r.getps().getModel() + " " + r.getps().getColor();
124     rentsData[j][6] = r.getDate();
125     rentsData[j][7] = String.valueOf(r.getHours());
126     rentsData[j][8] = "Rp" + String.valueOf(r.getTotal());
127     rentsData[j][9] = r.getStatusToString();
128 }
129
130 Color color2 = new Color(r:252, g:242, b:202);
131
132 JScrollPane panel = new JScrollPane(new JTable(rentsData, header, Color.black, color2));
133 panel.setBackground(bg:null);
134 panel.setViewport().setBackground(bg:null);
135 panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
136
137 gradientPanel.add(panel, BorderLayout.CENTER);
138 frame.add(gradientPanel, BorderLayout.CENTER);
139 frame.setVisible(b:true);
140 }
```

Data sewa yang telah diambil ditampilkan dalam tabel JTable dengan header yang telah ditentukan. Tabel tersebut kemudian ditambahkan ke dalam JScrollPane dan ditampilkan dalam panel dengan latar belakang gradasi.

3.1.1.11. ShowSpecUserRents.java

ShowSpecUserRents adalah implementasi antarmuka Operation yang memungkinkan pengguna untuk melihat daftar sewa berdasarkan pengguna tertentu dan menampilkan semua pengguna melalui antarmuka grafis (GUI). Kode ini mengatur frame dan panel, mengambil data pengguna dari database, dan menampilkan data dalam JComboBox dan JTable. Selain itu, terdapat tombol untuk menampilkan semua pengguna dan tombol untuk mengonfirmasi pilihan pengguna.

```
1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.GradientPaint;
6 import java.awt.Graphics;
7 import java.awt.Graphics2D;
8 import java.sql.ResultSet;
9 import java.sql.SQLException;
10 import java.util.ArrayList;
11
12 import javax.swing.BorderFactory;
13 import javax.swing.JFrame;
14 import javax.swing.JOptionPane;
15 import javax.swing.JPanel;
16 import javax.swing.JScrollPane;
17
18 import Model.ps;
19 import Model.Database;
20 import Model.JLabel;
21 import Model.JTable;
22 import Model.Operation;
23 import Model.User;
24
25 public class Viewps implements Operation {
26
```

Kelas ShowSpecUserRents mengimplementasikan antarmuka Operation. Kelas ini menyediakan method untuk menampilkan daftar sewa berdasarkan pengguna tertentu.

```
27 @Override
28 public void operation(Database database, JFrame f, User user) {
29
30     JFrame frame = new JFrame(title:"All Playstations");
31     frame.setSize(width:1000, height:600);
32     frame.setLocationRelativeTo(f);
33     frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
34
35     // Panel dengan latar belakang gradasi
36     JPanel gradientPanel = new JPanel() {
37         @Override
38         protected void paintComponent(Graphics g) {
39             super.paintComponent(g);
40             Graphics2D g2d = (Graphics2D) g;
41             int width = getWidth();
42             int height = getHeight();
43             Color color1 = new Color(r:224, g:255, b:255);
44             Color color2 = new Color(r:0, g:0, b:255);
45             GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, height, color2);
46             g2d.setPaint(gp);
47             g2d.fillRect(x:0, y:0, width, height);
48         }
49     };
50     gradientPanel.setLayout(new BorderLayout());
51
52     JLabel title = new JLabel(text:"All Playstations", fontSize:35);
53     title.setForeground(Color.BLUE);
54     title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
55     gradientPanel.add(title, BorderLayout.NORTH);
```

Method operation digunakan untuk melakukan operasi penampilan data sewa berdasarkan pengguna tertentu.

Parameter yang digunakan adalah objek Database, JFrame yang mewakili frame utama, dan User yang mewakili pengguna yang sedang aktif. Frame baru dibuat dengan judul "Show User's Rents", ukuran, lokasi, dan warna latar belakang yang diatur. Label judul ditambahkan ke panel bagian atas dengan beberapa properti yang diatur, termasuk border.

```

42 JPanel panel = new JPanel(new GridLayout(rows:2, cols:2, hgap:15, vgap:15));
43 panel.setBackground(bg:null);
44 panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
45
46 panel.add(new JLabel(text:"User ID:", font:Font("Times New Roman", 22)));
47
48 ArrayList<Integer> ids = new ArrayList<>();
49
50 try {
51     ResultSet rs0 = database.getStatement()
52         .executeQuery(sql:"SELECT `ID` FROM `users` WHERE `Type` = '0';");
53     while (rs0.next()) {
54         ids.add(rs0.getInt(columnLabel:"ID"));
55     }
56 } catch (SQLException e1) {
57     JOptionPane.showMessageDialog(frame, e1.getMessage());
58     frame.dispose();
59 }
60
61 String[] idsArr = new String[ids.size() + 1];
62 idsArr[0] = " ";
63 for (int i = 0; i < ids.size(); i++) {
64     idsArr[i + 1] = String.valueOf(ids.get(i));
65 }
66 JComboBox id = new JComboBox(idsArr, font:Font("Times New Roman", 22));
67 panel.add(id);
68
69 JButton showUsers = new JButton(text:"Show All Users", font:Font("Times New Roman", 22));
70 showUsers.addActionListener(new ActionListener() {
71     @Override
72     public void actionPerformed(ActionEvent e) {
73         showUsers(database, frame);
74     }
75 });
76 panel.add(showUsers);
77
78 JButton confirm = new JButton(text:"Confirm", font:Font("Times New Roman", 22));
79 confirm.addActionListener(new ActionListener() {
80     @Override
81     public void actionPerformed(ActionEvent e) {
82         if (id.getSelectedItem().toString().equals(anObject:"")) {
83             JOptionPane.showMessageDialog(frame, message:"User ID cannot be empty");
84             return;
85         }
86         new ShowUserRents(Integer.parseInt(id.getSelectedItem().toString())).operation(database, f, user);
87         frame.dispose();
88     }
89 });
90 panel.add(confirm);

```

Panel dengan layout grid dibuat dan beberapa komponen seperti label "User ID:" ditambahkan ke dalamnya. Kemudian mengambil data ID pengguna dari database yang memiliki tipe '0' dan menyimpannya dalam daftar ids. Data ID pengguna yang telah diambil ditampilkan dalam JComboBox untuk dipilih oleh pengguna. Tombol Show All Users ditambahkan dengan action listener untuk menampilkan semua pengguna. Tombol Confirm ditambahkan dengan action listener untuk memanggil operasi ShowUserRents jika ID pengguna telah dipilih.

```

97 |     private void showUsers(Database database, JFrame f) {
98 |         JFrame frame = new JFrame(title:"Clients List");
99 |         frame.setSize(width:1000, height:600);
100 |         frame.setLocationRelativeTo(f);
101 |         frame.getContentPane().setBackground(new Color(r:224, g:255, b:255));
102 |         frame.setLayout(new BorderLayout());
103 |
104 |         JLabel title = new JLabel(text:"Clients", fontSize:35);
105 |         title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
106 |         frame.add(title, BorderLayout.NORTH);
107 |
108 |         String[] header = new String[] { "ID", "First Name", "Last Name", "Email", "Tel" };
109 |
110 |         ArrayList<User> users = new ArrayList<>();
111 |         try {
112 |             ResultSet rs = database.getStatement().executeQuery(sql:"SELECT * FROM `users`;");
113 |             while (rs.next()) {
114 |                 int accType = rs.getInt(columnLabel:"Type");
115 |                 if (accType == 0) {
116 |                     User u = new Client();
117 |                     u.setID(rs.getInt(columnLabel:"ID"));
118 |                     u.setFirstName(rs.getString(columnLabel:"FirstName"));
119 |                     u.setLastName(rs.getString(columnLabel:"LastName"));
120 |                     u.setEmail(rs.getString(columnLabel:"Email"));
121 |                     u.setPhoneNumber(rs.getString(columnLabel:"PhoneNumber"));
122 |                     users.add(u);
123 |                 }
124 |             }
125 |         } catch (SQLException e) {
126 |             JOptionPane.showMessageDialog(frame, e.getMessage());
127 |             frame.dispose();
128 |         }
129 |
130 |         String[][] usersData = new String[users.size()][5];
131 |         for (int i = 0; i < users.size(); i++) {
132 |             User u = users.get(i);
133 |             usersData[i][0] = String.valueOf(u.getID());
134 |             usersData[i][1] = u.getFirstName();
135 |             usersData[i][2] = u.getLastName();
136 |             usersData[i][3] = u.getEmail();
137 |             usersData[i][4] = u.getPhoneNumber();
138 |         }
139 |
140 |         Color color2 = new Color(r:252, g:242, b:202);
141 |
142 |         JScrollPane panel = new JScrollPane(new JTable(usersData, header, Color.black, color2));
143 |         panel.setBackground(bg:null);
144 |         panel.setViewport().setBackground(bg:null);
145 |         panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
146 |
147 |         frame.add(panel, BorderLayout.CENTER);
148 |         frame.setVisible(b:true);
149 |
150 |     }
151 |
152 |
153 |

```

Method showUsers menampilkan daftar semua pengguna dalam frame baru. Data pengguna diambil dari database dan ditampilkan dalam tabel JTable.

3.1.1.12. ShowUserRents.java

ShowUserRents adalah implementasi antarmuka Operation yang memungkinkan pengguna untuk melihat daftar sewa yang dilakukan oleh pengguna tertentu melalui antarmuka grafis (GUI). Kode ini mengatur frame dan panel, mengambil data sewa dari database, dan menampilkan data dalam tabel JTable. Selain itu, terdapat mekanisme untuk mengambil informasi pengguna dan data sewa terkait dari database.

```

1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.util.ArrayList;
8
9 import javax.swing.BorderFactory;
10 import javax.swing.JFrame;
11 import javax.swing.JOptionPane;
12 import javax.swing.JScrollPane;
13
14 import Model.ps;
15 import Model.Client;
16 import Model.Database;
17 import Model.JLabel;
18 import Model.JTable;
19 import Model.Operation;
20 import Model.Rent;
21 import Model.User;
22
23 public class ShowUserRents implements Operation {
24
25     private int userID;
26
27     public ShowUserRents(int userID) {
28         this.userID = userID;
29     }
30

```

Kelas ShowUserRents mengimplementasikan antarmuka Operation dan memiliki konstruktor yang menerima parameter userID.

```

36
37     @Override
38     public void operation(Database database, JFrame f, User user) {
39         if (userID == -9999)
40             userID = user.getID();
41
42         JFrame frame = new JFrame(title:"Rents");
43         frame.setSize(width:1200, height:600);
44         frame.setLocationRelativeTo(f);
45         frame.getContentPane().setBackground(new Color(r:236, g:240, b:241));
46         frame.setLayout(new BorderLayout());
47
48         JLabel title = new JLabel(text:"Rents", JLabel.CENTER);
49         title.setForeground(new Color(r:44, g:62, b:80));
50         title.setFont(title.getFont().deriveFont(size:35f));
51         title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:20, right:0));
52         frame.add(title, BorderLayout.NORTH);
53
54         String[] header = new String[] {
55             "ID", "Name", "Email", "Tel", "PS ID", "PS", "Date Time",
56             "Hours", "Total", "Status"
57         };

```

Method operation digunakan untuk melakukan operasi penampilan data sewa berdasarkan pengguna tertentu. Jika userID adalah -9999, maka userID akan diisi dengan ID pengguna yang sedang aktif. Frame baru dibuat dengan judul "Rents", ukuran, lokasi, dan warna latar belakang yang diatur. Label judul ditambahkan ke panel bagian atas dengan beberapa properti yang diatur, termasuk border dan font. Kemudian tambahkan header untuk tabel yang akan ditampilkan didefinisikan.

```

52
53     ArrayList<Rent> rents = new ArrayList<>();
54     ArrayList<Integer> psIDs = new ArrayList<>();
55     try {
56         String select = "SELECT * FROM `rents` WHERE `User` = '" + userID + "'";
57         ResultSet rs = database.getStatement().executeQuery(select);
58         while (rs.next()) {
59             Rent rent = new Rent();
60             rent.setID(rs.getInt(columnLabel:"ID"));
61             psIDs.add(rs.getInt(columnLabel:"ps"));
62             rent.setDateTime(rs.getString(columnLabel:"DateTime"));
63             rent.setHours(rs.getInt(columnLabel:"Hours"));
64             rent.setTotal(rs.getDouble(columnLabel:"Total"));
65             rent.setStatus(rs.getInt(columnLabel:"Status"));
66             rents.add(rent);
67         }
68
69         String selectUser = "SELECT * FROM `users` WHERE `ID` = '" + userID + "'";
70         ResultSet rs2 = database.getStatement().executeQuery(selectUser);
71         rs2.next();
72         User u = new Client();
73         u.setID(rs2.getInt(columnLabel:"ID"));
74         u.setFirstName(rs2.getString(columnLabel:"FirstName"));
75         u.setLastName(rs2.getString(columnLabel:"LastName"));
76         u.setEmail(rs2.getString(columnLabel:"Email"));
77         u.setPhoneNumber(rs2.getString(columnLabel:"PhoneNumber"));
78         u.setPassword(rs2.getString(columnLabel:"Password"));
79
80         for (int j = 0; j < rents.size(); j++) {
81             Rent r = rents.get(j);
82             r.setUser(u);
83             ResultSet rs3 = database.getStatement()
84             .executeQuery("SELECT * FROM `pss` WHERE `ID` = '" + psIDs.get(j) + "'");
85             rs3.next();
86             ps ps = new ps();
87             ps.setID(rs3.getInt(columnLabel:"ID"));
88             ps.setBrand(rs3.getString(columnLabel:"Brand"));
89             ps.setModel(rs3.getString(columnLabel:"Model"));
90             ps.setColor(rs3.getString(columnLabel:"Color"));
91             ps.setYear(rs3.getInt(columnLabel:"Year"));
92             ps.setPrice(rs3.getDouble(columnLabel:"Price"));
93             ps.setAvailable(rs3.getInt(columnLabel:"Available"));
94             r.setps(ps);
95         }
96
97     } catch (SQLException e) {
98         JOptionPane.showMessageDialog(frame, e.getMessage());
99         frame.dispose();
100    }

```

Bagian ini mengambil data sewa dan informasi pengguna dari database. Data disimpan dalam objek Rent dan User dan ditambahkan ke dalam daftar rents dan psIDs.

```

101
102     String[][] rentsData = new String[rents.size()][10];
103     for (int j = 0; j < rents.size(); j++) {
104         Rent r = rents.get(j);
105         rentsData[j][0] = String.valueOf(r.getID());
106         rentsData[j][1] = r.getUser().getFirstName() + " " + r.getUser().getLastName();
107         rentsData[j][2] = r.getUser().getEmail();
108         rentsData[j][3] = r.getUser().getPhoneNumber();
109         rentsData[j][4] = String.valueOf(r.getps().getID());
110         rentsData[j][5] = r.getps().getBrand() + " " + r.getps().getModel() + " " + r.getps().getColor();
111         rentsData[j][6] = r.getDateTime();
112         rentsData[j][7] = String.valueOf(r.getHours());
113         rentsData[j][8] = "Rp " + String.valueOf(r.getTotal());
114         rentsData[j][9] = r.getStatusToString();
115     }
116
117     Color color2 = new Color(r:252, g:242, b:202);
118
119     JScrollPane panel = new JScrollPane(new JTable(rentsData, header, Color.black, color2));
120     panel.setBackground(bg:null);
121     panel.setViewport().setBackground(bg:null);
122     panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
123
124     frame.add(panel, BorderLayout.CENTER);
125     frame.setVisible(b:true);
126 }

```

Data sewa yang telah diambil dari database dimasukkan ke dalam array dua dimensi rentsData, yang kemudian akan digunakan untuk mengisi tabel. Data dalam rentsData ditampilkan dalam JTable dengan header yang telah didefinisikan. Tabel dimasukkan ke dalam JScrollPane dan ditambahkan ke frame. Frame kemudian ditampilkan.

3.1.1.13. Updateps.java

```
1 package Controller;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.GradientPaint;
6 import java.awt.Graphics;
7 import java.awt.Graphics2D;
8 import java.awt.GridLayout;
9 import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14
15 import javax.swing.BorderFactory;
16 import javax.swing.JFrame;
17 import javax.swing.JOptionPane;
18 import javax.swing.JPanel;
19
20 import Model.ps;
21 import Model.Database;
22 import Model.JButton;
23 import Model.JComboBox;
24 import Model.JLabel;
25 import Model.JTextField;
26 import Model.Operation;
27 import Model.User;
28
29 public class Updateps implements Operation {
30
31     private JTextField brand, model, color, year, price;
32     private Database database;
33     private JFrame frame;
34 }
```

Deklarasikan package Controller dan mengimpor berbagai kelas dari package java.awt, java.sql, java.util, dan javax.swing. Selain itu, kelas-kelas dari package Model juga diimpor. Kelas Updateps mengimplementasikan antarmuka Operation dan memiliki beberapa variabel instance untuk menyimpan referensi elemen-elemen GUI serta objek Database dan JFrame.

```

35     @Override
36     public void operation(Database database, JFrame f, User user) {
37
38         this.database = database;
39
40         frame = new JFrame(title:"Update PS Data");
41         frame.setSize(width:600, height:600);
42         frame.setLocationRelativeTo(f);
43         frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
44
45         // Panel dengan latar belakang gradasi
46         JPanel gradientPanel = new JPanel() {
47             @Override
48             protected void paintComponent(Graphics g) {
49                 super.paintComponent(g);
50                 Graphics2D g2d = (Graphics2D) g;
51                 int width = getWidth();
52                 int height = getHeight();
53                 Color color1 = new Color(r:224, g:255, b:255);
54                 Color color2 = new Color(r:0, g:0, b:255);
55                 GradientPaint gp = new GradientPaint(x1:0, y1:0, color1, x2:0, height, color2);
56                 g2d.setPaint(gp);
57                 g2d.fillRect(x:0, y:0, width, height);
58             }
59         };
60         gradientPanel.setLayout(new BorderLayout());
61
62         JLabel title = new JLabel(text:"Update PS Data", fontSize:35);
63         title.setForeground(Color.BLUE);
64         title.setBorder(BorderFactory.createEmptyBorder(top:20, left:0, bottom:0, right:0));
65         gradientPanel.add(title, BorderLayout.NORTH);
66
67         JPanel panel = new JPanel(new GridLayout(rows:7, cols:2, hgap:15, vgap:15));
68         panel.setBackground(bg:null);
69         panel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));
70

```

Method operation digunakan untuk melakukan operasi pembaruan data PS. Method ini pertama-tama menyimpan referensi ke objek Database dan mengatur beberapa properti dari frame. Panel utama gradientPanel dengan latar belakang gradasi dibuat dengan menggunakan paintComponent untuk mengatur gradasi warna. Label judul ditambahkan ke panel bagian atas. Sebuah panel GridLayout juga dibuat untuk menampung elemen-elemen input data.

```

72     String[] ids = new String[]{ " " };
73     ArrayList<Integer> idsArray = new ArrayList<>();
74     try {
75         ResultSet rs0 = database.getStatement()
76             .executeQuery(sql:"SELECT `ID`, `Available` FROM `pss`");
77         while (rs0.next()) {
78             if (rs0.getInt(columnLabel:"Available") < 2)
79                 idsArray.add(rs0.getInt(columnLabel:"ID"));
80         }
81     } catch (Exception e0) {
82         JOptionPane.showMessageDialog(frame, e0.getMessage());
83         frame.dispose();
84     }
85
86     ids = new String[idsArray.size() + 1];
87     ids[0] = " ";
88     for (int i = 1; i <= idsArray.size(); i++) {
89         ids[i] = String.valueOf(idsArray.get(i - 1));
90     }
91
92     JComboBox id = new JComboBox(ids, fontSize:22);
93     id.addActionListener(new ActionListener() {
94         @Override
95         public void actionPerformed(ActionEvent e) {
96             updateData(id.getSelectedItem().toString());
97         }
98     });
99     panel.add(id);
100

```

Bagian ini mengisi JComboBox dengan ID PS yang tersedia (dengan Available kurang dari 2). Data diambil dari database dan dimasukkan ke dalam idsArray, kemudian diubah menjadi array ids.

```
102 |         panel.add(new JLabel(text:"Brand:", fontSize:22));
103 |
104 |         brand = new JTextField(fontSize:22);
105 |         panel.add(brand);
106 |
107 |         panel.add(new JLabel(text:"Model:", fontSize:22));
108 |
109 |         model = new JTextField(fontSize:22);
110 |         panel.add(model);
111 |
112 |         panel.add(new JLabel(text:"Color:", fontSize:22));
113 |
114 |         color = new JTextField(fontSize:22);
115 |         panel.add(color);
116 |
117 |         panel.add(new JLabel(text:"Year:", fontSize:22));
118 |
119 |         year = new JTextField(fontSize:22);
120 |         panel.add(year);
121 |
122 |         panel.add(new JLabel(text:"Price per Hour:", fontSize:22));
123 |
124 |         price = new JTextField(fontSize:22);
125 |         panel.add(price);
```

Bagian ini membuat label dan field input untuk setiap atribut PS seperti Brand, Model, Color, Year, dan Price per Hour.

```
127 |         JButton cancel = new JButton(text:"Cancel", textSize:22);
128 |         cancel.addActionListener(new ActionListener() {
129 |             @Override
130 |             public void actionPerformed(ActionEvent e) {
131 |                 frame.dispose();
132 |             }
133 |         });
134 |         panel.add(cancel);
135 |
136 |         JButton confirm = new JButton(text:"Confirm", textSize:22);
137 |         confirm.addActionListener(new ActionListener() {
138 |             @Override
139 |             public void actionPerformed(ActionEvent e) {
140 |                 if (id.getSelectedItem().toString().equals(anObject:"")) {
141 |                     JOptionPane.showMessageDialog(frame, message:"ID cannot be empty");
142 |                     return;
143 |                 }
144 |                 if (brand.getText().equals(anObject:"")) {
145 |                     JOptionPane.showMessageDialog(frame, message:"Brand cannot be empty");
146 |                     return;
147 |                 }
148 |                 if (model.getText().equals(anObject:"")) {
149 |                     JOptionPane.showMessageDialog(frame, message:"Model cannot be empty");
150 |                     return;
151 |                 }
152 |                 if (color.getText().equals(anObject:"")) {
153 |                     JOptionPane.showMessageDialog(frame, message:"Color cannot be empty");
154 |                     return;
155 |                 }
156 |             }
157 |         });
158 |         panel.add(confirm);
```

```

155
156     }
157     if (year.getText().equals(anObject:"")) {
158         JOptionPane.showMessageDialog(frame, message:"Year cannot be empty");
159         return;
160     }
161     if (price.getText().equals(anObject:"")) {
162         JOptionPane.showMessageDialog(frame, message:"Price cannot be empty");
163         return;
164     }
165     int yearInt;
166     double priceDoub;
167     try {
168         yearInt = Integer.parseInt(year.getText());
169     } catch (Exception e1) {
170         JOptionPane.showMessageDialog(frame, message:"Year must be int");
171         return;
172     }
173     try {
174         priceDoub = Double.parseDouble(price.getText());
175     } catch (Exception e1) {
176         JOptionPane.showMessageDialog(frame, message:"Price must be double");
177         return;
178     }

```

```

179     try {
180         String update = "UPDATE `pss` SET `Brand`='"
181             + brand.getText() + "", `Model`='"
182             + model.getText()
183             + ",`Color`='"
184             + color.getText() + ",`Year`='"
185             + yearInt + ",`Price`='"
186             + priceDoub + ""
187             + "WHERE `ID` = '"
188             + id.getSelectedItem().toString() + "'";
189
190         database.createStatement().execute(update);
191         JOptionPane.showMessageDialog(frame, message:"PS updated successfully");
192         frame.dispose();
193     } catch (SQLException e2) {
194         JOptionPane.showMessageDialog(frame, e2.getMessage());
195     }
196 }
197 });
198 panel.add(confirm);

```

Bagian di atas membuat dua tombol: Cancel dan Confirm. Tombol Cancel menutup frame, sedangkan tombol Confirm melakukan validasi input dan mengupdate data PS di database.

```

200
201     private void updateData(String ID) {
202         if (ID.equals(anObject:" ")) {
203             brand.setText(t:"");
204             model.setText(t:"");
205             color.setText(t:"");
206             year.setText(t:"");
207             price.setText(t:"");
208         } else {
209             try {
210                 ResultSet rs1 = database.createStatement()
211                     .executeQuery("SELECT * FROM `pss` WHERE `ID` = '"
212                     + ID + "'");
213                 rs1.next();
214                 ps ps = new ps();
215                 ps.setID(rs1.getInt(columnLabel:"ID"));
216                 brand.setText(rs1.getString(columnLabel:"Brand"));
217                 model.setText(rs1.getString(columnLabel:"Model"));
218                 color.setText(rs1.getString(columnLabel:"Color"));
219                 year.setText(String.valueOf(rs1.getInt(columnLabel:"Year")));
220                 price.setText(String.valueOf(rs1.getDouble(columnLabel:"Price")));
221             } catch (Exception e1) {
222                 JOptionPane.showMessageDialog(frame, e1.getMessage());
223             }
224         }
225     }

```

Method updateData digunakan untuk mengisi field input dengan data PS yang dipilih dari JComboBox. Jika ID PS adalah " ", maka field input akan dikosongkan.

3.1.1.14. Viewpss.java

```
package Controller;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

import Model.ps;
import Model.Database;
import Model.JLabel;
import Model.JTable;
import Model.Operation;
import Model.User;
```

Deklarasi paket dan import untuk semua kelas yang diperlukan pada paket java.awt, java.sql, java.util dan javax.swing, dan juga kelas dari paket Model.

```
public class Viewpss implements Operation {
```

Deklarasi dari kelas Viewpss untuk mengimplementasikan antarmuka Operation.

```
    public void operation(Database database, JFrame f, User user) {
```

Implementasi pada metode operation untuk menampilkan semua data playstation dalam bentuk tabel.

```

 JPanel gradientPanel = new JPanel() {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        int width = getWidth();
        int height = getHeight();
        Color color1 = new Color(224, 255, 255);
        Color color2 = new Color(0, 0, 255);
        GradientPaint gp = new GradientPaint(0, 0, color1, 0, height, color2);
        g2d.setPaint(gp);
        g2d.fillRect(0, 0, width, height);
    }
};
gradientPanel.setLayout(new BorderLayout());

```

Bagian ini untuk membuat Frame dan Panel dengan latar belakang gradasi.

```

JLabel title = new JLabel(text:"All Playstations", fontSize:35);
title.setForeground(Color.BLUE);
title.setBorder(BorderFactory.createEmptyBorder(20, 0, 0, 0));
gradientPanel.add(title, BorderLayout.NORTH);

```

Bagian untuk menambahkan judul ke panel dengan pengaturan font dan warna tertentu.

```

String[] header = new String[] {
    "ID", "Brand", "Model", "Color", "Year", "Price", "Available"
};

```

Bagian yang mendefinisikan header tabel yang digunakan untuk menampilkan data playstation.

```

String select = "SELECT * FROM `pss`";
ArrayList<ps> pss = new ArrayList<>();
try {
    ResultSet rs = database.getStatement().executeQuery(select);
    while (rs.next()) {
        ps ps = new ps();
        ps.setID(rs.getInt("ID"));
        ps.setBrand(rs.getString("Brand"));
        ps.setModel(rs.getString("Model"));
        ps.setColor(rs.getString("Color"));
        ps.setYear(rs.getInt("Year"));
        ps.setPrice(rs.getDouble("Price"));
        int available = rs.getInt("Available");
        if (available < 2) {
            ps.setAvailable(available);
            pss.add(ps);
        }
    }
} catch (SQLException e) {
    JOptionPane.showMessageDialog(frame, e.getMessage());
}

```

Bagian yang mengambil data dari database. Kode ini menjalankan query untuk mengambil data dari tabel pss dan menambahkannya ke dalam daftar pss.

```

String[][] pssData = new String[pss.size()][7];
for (int j = 0; j < pss.size(); j++) {
    ps c = pss.get(j);
    if (c.isAvailable() < 2) {
        pssData[j][0] = String.valueOf(c.getID());
        pssData[j][1] = c.getBrand();
        pssData[j][2] = c.getModel();
        pssData[j][3] = c.getColor();
        pssData[j][4] = String.valueOf(c.getYear());
        pssData[j][5] = "Rp" + String.valueOf(c.getPrice());
        if (c.isAvailable() == 0) {
            pssData[j][6] = "Available";
        } else {
            pssData[j][6] = "Not Available";
        }
    }
}

```

Bagian ini menyusun data dari tabel yang kemudian mengisi array pssData dengan data PS dari daftar pss.

```

Color color2 = new Color(252, 242, 202);

JScrollPane panel = new JScrollPane(new JTable(pssData, header, Color.BLACK, color2));
panel.setBackground(null);
panel.setViewport().setBackground(null);
panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

gradientPanel.add(panel, BorderLayout.CENTER);
frame.add(gradientPanel, BorderLayout.CENTER);
frame.setVisible(true);

```

Bagian yang membuat tabel dengan data PS dan menambahkannya ke panel yang sudah dibuat sebelumnya. Tabel ini ditampilkan di dalam JScrollPane untuk memungkinkan scrolling.

3.1.2. Penjelasan folder Model

3.1.1.1. Admin.java

Admin merupakan subclass pada kelas User. File ini memungkinkan admin untuk melakukan berbagai operasi seperti menambah PS baru, melihat data PS, menghapus bahkan mengubah kata sandi.

```
1 package Model;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.Dimension;
6 import java.awt.Font;
7 import java.awt.GridBagConstraints;
8 import java.awt.GridBagLayout;
9 import java.awt.GridLayout;
10 import java.awt.Insets;
11 import java.awt.event.ActionEvent;
12 import java.awt.event.ActionListener;
13 import java.text.NumberFormat;
14 import java.util.Locale;
15 import javax.swing.BorderFactory;
16 import javax.swing.JButton;
17 import javax.swing.JFrame;
18 import javax.swing.JLabel;
19 import javax.swing.JPanel;
20 import javax.swing.ImageIcon;
21
22 import Controller.AddNewAccount;
23 import Controller.AddNewps;
24 import Controller.ChangePassword;
25 import Controller.Deleteps;
26 import Controller.EditUserData;
27 import Controller.ShowAllRents;
28 import Controller.ShowSpecUserRents;
29 import Controller.Updateps;
30 import Controller.Viewpss;
31
```

Import mencakup kelas dari paket ‘java.awt’ dan ‘java.swing’. ‘java.awt’ digunakan untuk bagian tampilan pada Admin.

```
32 public class Admin extends User {
33
34     private Operation[] operations = new Operation[] {
35         new AddNewps(),
36         new Viewpss(),
37         new Updateps(),
38         new Deletes(),
39         new AddNewAccount(accType:1),
40         new ShowAllRents(),
41         new ShowSpecUserRents(),
42         new EditUserData(),
43         new ChangePassword()
44     };
45
46     private JButton[] btns = new JButton[] {
47         new JButton("Add New PS Unit", new ImageIcon("icons/add.png")),
48         new JButton("View PS", new ImageIcon("icons/view.png")),
49         new JButton("Update PS", new ImageIcon("icons/update.png")),
50         new JButton("Delete PS", new ImageIcon("icons/delete.png")),
51         new JButton("Add New Admin", new ImageIcon("icons/add_admin.png")),
52         new JButton("Show Rents", new ImageIcon("icons/show_rents.png")),
53         new JButton("Show User's Rents", new ImageIcon("icons/show_user_rents.png")),
54         new JButton("Edit my Data", new ImageIcon("icons/edit.png")),
55         new JButton("Change Password", new ImageIcon("icons/change_password.png"))
56     };
57
58     public Admin() {
59         super();
60     }
61 }
```

Deklarasi dari kelas dan variabel.

```
62     @Override
63     public void showList(Database database, JFrame f) {
64         JFrame frame = new JFrame("Admin Dashboard");
65         frame.setSize(1000, 700);
66         frame.setLocationRelativeTo(f);
67         frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
68         frame.setLayout(new BorderLayout());
69 }
```

Metode showlist untuk membuat JFrame baru untuk dashboard admin.

```
// Header
JPanel header = new JPanel(new BorderLayout());
header.setBackground(new Color(44, 62, 80));
JLabel title = new JLabel("Welcome " + getFirstName(), JLabel.CENTER);
title.setForeground(Color.WHITE);
title.setFont(new Font("Arial", Font.BOLD, 30));
header.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0));
header.add(title, BorderLayout.CENTER);
```

Bagian header untuk membuat panel header yang menampilkan pesan Welcome.

```
// Side Menu
JPanel sideMenu = new JPanel(new GridLayout(btns.length, 1, 10, 10));
sideMenu.setBackground(new Color(52, 73, 94));
sideMenu.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

for (int i = 0; i < btns.length; i++) {
    final int j = i;
    JButton button = btns[i];
    button.setBackground(new Color(41, 128, 185));
    button.setForeground(Color.WHITE);
    button.setFocusPainted(false);
    button.setFont(new Font("Arial", Font.BOLD, 16));
    button.setPreferredSize(new Dimension(200, 40));
    sideMenu.add(button);
    button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            operations[j].operation(database, frame, Admin.this);
        }
    });
}
```

Bagian side menu untuk membuat panel menu samping dengan tombol-tombol operasi yang berbeda dan menambahkan ActionListener di setiap tombol agar menjalankan operasi yang sesuai saat tombol diklik.

```

JPanel summaryPanel = new JPanel(new GridBagLayout());
summaryPanel.setBackground(new Color(236, 240, 241));
summaryPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(10, 10, 10, 10);
gbc.anchor = GridBagConstraints.WEST;
gbc.fill = GridBagConstraints.BOTH;

JPanel totalPSUnitsCard = createSummaryCard(title:"Total PS Units", String.valueOf(database.getTotalPSUnits()));
gbc.gridx = 0;
gbc.gridy = 0;
summaryPanel.add(totalPSUnitsCard, gbc);

JPanel totalUsersCard = createSummaryCard(title:"Total Users", String.valueOf(database.getTotalUsers()));
gbc.gridx = 1;
gbc.gridy = 0;
summaryPanel.add(totalUsersCard, gbc);

JPanel totalRentsCard = createSummaryCard(title:"Total Rents", String.valueOf(database.getTotalRents()));
gbc.gridx = 0;
gbc.gridy = 1;
summaryPanel.add(totalRentsCard, gbc);

JPanel totalEarningsCard = createSummaryCard(title:"Total Earnings", formatCurrency(database.getTotalEarnings()));
gbc.gridx = 1;
gbc.gridy = 1;
summaryPanel.add(totalEarningsCard, gbc);

mainPanel.add(summaryPanel, BorderLayout.CENTER);

```

Bagian main panel untuk membuat panel utama yang menampilkan ringkasan data dengan kartu yang berisi informasi seperti jumlah unit PS, jumlah pengguna, jumlah penyewaan, dan total pendapatan.

```

// Footer
JPanel footer = new JPanel();
footer.setBackground(new Color(44, 62, 80));
JLabel footerText = new JLabel("Kelompok 3 - Pemrograman Berorientasi Objek © 2024", JLabel.CENTER);
footerText.setForeground(Color.WHITE);
footerText.setFont(new Font("Arial", Font.PLAIN, 12));
footer.setBorderStyle(BorderFactory.createEmptyBorder(10, 0, 10, 0));
footer.add(footerText);

frame.add(header, BorderLayout.NORTH);
frame.add(sideMenu, BorderLayout.WEST);
frame.add(mainPanel, BorderLayout.CENTER);
frame.add(footer, BorderLayout.SOUTH);

frame.setVisible(true);

```

Footer digunakan untuk membuat panel dengan teks statis dan menampilkan frame.

```
private String formatCurrency(double amount) {
    NumberFormat currencyFormatter = NumberFormat.getCurrencyInstance(new Locale("id", "ID"));
    return currencyFormatter.format(amount);
}

private JPanel createSummaryCard(String title, String value) {
    JPanel card = new JPanel(new BorderLayout());
    card.setBackground(Color.WHITE);
    card.setBorder(BorderFactory.createCompoundBorder(
        BorderFactory.createTitledBorder(new Color(189, 195, 199), 1),
        BorderFactory.createEmptyBorder(20, 20, 20, 20)));
    JLabel titleLabel = new JLabel(title);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 18));
    titleLabel.setForeground(new Color(44, 62, 80));

    JLabel valueLabel = new JLabel(value);
    valueLabel.setFont(new Font("Arial", Font.BOLD, 36));
    valueLabel.setForeground(new Color(41, 128, 185));
    valueLabel.setHorizontalAlignment(JLabel.CENTER);

    card.add(titleLabel, BorderLayout.NORTH);
    card.add(valueLabel, BorderLayout.CENTER);
    return card;
}
```

Sebagai metode pembantu yang menformat angka menjadi mata uang dan membuat kartu ringkasan dengan judul dan nilai tertentu.

3.1.1.2. Client.java

```
public class Client extends User {

    private Operation[] operations = new Operation[] {
        new Viewpss(),
        new Rentps(),
        new Returnps(),
        new ShowUserRents(-9999),
        new EditUserData(),
        new ChangePassword() };

    private JButton[] btns = new JButton[] {
        new JButton("View Schedule"),
        new JButton("Start Play"),
        new JButton("Return"),
        new JButton("Show My Schedule"),
        new JButton("Edit My Data"),
        new JButton("Change Password")
    };

    public Client() {
        super();
    }
}
```

Deklarasi Kelas dan Atribut untuk subclass dari User, dan Client mewarisi semua properti dan metode dari User. Operations merupakan array dari objek operation yang merepresentasikan tindakan yang dilakukan oleh pengguna ‘Client’. Dan di akhir, ada Konstruktor Client digunakan untuk memanggil konstruktor superclass.

```

public void showList(Database database, JFrame f) {
    JFrame frame = new JFrame("Client Panel");
    frame.setSize(400, btns.length * 90);
    frame.setLocationRelativeTo(f);
    frame.getContentPane().setBackground(new Color(236, 240, 241));
    frame.setLayout(new BorderLayout());
    JLabel title = new JLabel("Welcome " + getFirstName(), JLabel.CENTER);
    title.setFont(title.getFont().deriveFont(30f));
    title.setBorder(BorderFactory.createEmptyBorder(15, 0, 0, 0));
    frame.add(title, BorderLayout.NORTH);

    JPanel panel = new JPanel(new GridLayout(btns.length, 1, 15, 15));
    panel.setBackground(null);
    panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

    for (int i = 0; i < btns.length; i++) {
        final int j = i;
        JButton button = btns[i];
        panel.add(button);
        button.setBackground(new Color(41, 128, 185));
        button.setForeground(Color.WHITE);
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                operations[j].operation(database, frame, Client.this);
            }
        });
    }

    frame.add(panel, BorderLayout.CENTER);
    frame.setVisible(true);
}

```

Metode showlist untuk menampilkan antarmuka pengguna.
Membuat JFrame dengan ukuran yang sesuai dan posisi di tengah layar.
Mengatur backgorund dan membuat JLabel, JPanel.
Menambahkan tombol dan panel ke frame.

3.1.1.3. Database.java

```
package Model;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

Mengimport bagian kelas yang diperlukan dengan basis data SQL.

```
public class Database {
    private String user = "root";
    private String password = "";
    private String url = "jdbc:mysql://localhost/psrentalsystem";
    private Statement statement;
```

Deklarasi kelas dan atribut dan bagian user, password, dan url merupakan string yang menyimpan informasi kredensial dan URL koneksi ke basis data MySQL.

```
public Database() {
    try {
        Connection connection = DriverManager.getConnection(url, user, password);
        statement = connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Konstruktor untuk membuat koneksi ke basis data menggunakan DriverManager dan menginisialisasi objek dengan tipe INSENSITIVE dan READ_ONLY.

```
public Statement getStatement() {
    return statement;
}

public int getTotalPSUnits() {
    try {
        ResultSet resultSet = statement.executeQuery("SELECT COUNT(*) AS total FROM pss");
        if (resultSet.next()) {
            return resultSet.getInt("total");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return 0;
}
```

Metode getStatement untuk mengembalikan objek yang dapat digunakan untuk mengeksekusi query SQL. Metode getTotalPSUnits untuk menghitung total unit PS yang tersedia dalam tabel pss.

```

public int getTotalUsers() {
    try {
        ResultSet resultSet = statement.executeQuery("SELECT COUNT(*) AS total FROM users");
        if (resultSet.next()) {
            return resultSet.getInt("total");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return 0;
}

public int getTotalRents() {
    try {
        ResultSet resultSet = statement.executeQuery("SELECT COUNT(*) AS total FROM rents");
        if (resultSet.next()) {
            return resultSet.getInt("total");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return 0;
}

public double getTotalEarnings() {
    double totalEarnings = 0.0;
    try {
        ResultSet resultSet = statement
            .executeQuery("SELECT SUM(Total) AS totalEarned FROM rents WHERE Status = 1");
        if (resultSet.next()) {
            totalEarnings = resultSet.getDouble("totalEarned");
        }
    }
}

```

Metode getTotalUsers untuk menghitung total pengguna yang terdaftar dalam tabel users. Metode getTotalRents untuk menghitung total transaksi penyewaan dalam tabel rents. Metode getTotalEarnings untuk menghitung total pendapatan dari transaksi penyewaan yang statusnya 1 = selesai.

3.1.1.4. JButton.java

```
package Model;

import java.awt.Color;
import java.awt.Font;

@SuppressWarnings("serial")      At least one of the problems in ca
public class JButton extends javax.swing.JButton {

    public JButton(String text, int textSize) {
        super(text);
        setBackground(Color.black);
        setFont(new Font("SansSerif", Font.BOLD, textSize));
        setForeground(Color.white);
        setBorder(null);
    }

}
```

Pada kode, menggunakan deklarasi kelas Annotation dengan mewarisi kelas javax.swing.JButton. Menggunakan konstruktor kelas yang diinisiasi objek JButton dengan teks dan ukuran font yang ditentukan.

3.1.1.5. JComboBox.java

```
package Model;

import java.awt.Color;
import java.awt.Font;

import javax.swing.JLabel;
import javax.swing.SwingConstants;

@SuppressWarnings({ "rawtypes", "serial" })    At least one of the problems in c
public class JComboBox extends javax.swing.JComboBox {

    @SuppressWarnings("unchecked")
    public JComboBox(String[] items, int fontSize) {
        super(items);
        setFont(new Font("SansSerif", Font.BOLD, fontSize));
        setBackground(Color.white);
        ((JLabel) getRenderer()).setHorizontalAlignment(SwingConstants.CENTER);
    }

}
```

Deklarasi Kelas dengan custom JComboBox yang mewarisi kelas kelas javax.swing.JComboBox. Konstruktor yang digunakan untuk menginisiasi objek dengan item dan ukuran font yang ditentukan.

- super(items): Memanggil konstruktor dari javax.swing.JComboBox dengan parameter item.
- setFont(new Font("SansSerif", Font.BOLD, fontSize)): Mengatur font dari teks pada combo box dengan jenis huruf SansSerif, gaya BOLD, dan ukuran yang ditentukan (fontSize).
- setBackground(Color.white): Mengatur latar belakang combo box menjadi putih.
- ((JLabel)
getRenderer()).setHorizontalAlignment(SwingConstants.CENTER):
Mengatur perataan horizontal dari renderer (komponen yang digunakan untuk menampilkan setiap item dalam combo box) menjadi tengah. getRenderer() mengembalikan renderer saat ini untuk combo box, yang diubah menjadi JLabel untuk mengatur perataannya.

3.1.1.6. JLabel.java

```
package Model;
import java.awt.Font;

@SuppressWarnings("serial")      At least one of the problems in cat
public class JLabel extends javax.swing.JLabel {

    public JLabel(String text, int fontSize) {
        super(text);
        setFont(new Font("SansSerif", Font.BOLD, fontSize));
        setBackground(null);
        setHorizontalAlignment(JLabel.CENTER);
    }

}
```

Mendeklarasikan kelas JLabel yang memperluas kelas javax.swing.JLabel, yang berarti mewarisi semua properti dan metode dari javax.swing.JLabel.

- String text: Teks yang akan ditampilkan oleh label.
- int fontSize: Ukuran font yang akan digunakan oleh label.
- super(text): Memanggil konstruktor dari superclass (javax.swing.JLabel) dengan teks yang diberikan. Ini menginisialisasi label dengan teks tersebut.
- setFont(new Font("SansSerif", Font.BOLD, fontSize)): Mengatur font dari label menjadi "SansSerif" dengan gaya BOLD dan ukuran sesuai parameter fontSize.
- setBackground(null): Mengatur latar belakang label menjadi null, yang berarti label tidak akan memiliki latar belakang khusus.
- setHorizontalAlignment(JLabel.CENTER): Mengatur teks pada label untuk berada di tengah secara horizontal.

3.1.1.7. JPasswordField.java

```
package Model;
import java.awt.Font;

@SuppressWarnings("serial")      At least one of the problems in cat
public class JPasswordField extends javax.swing.JPasswordField {

    public JPasswordField(int textSize) {
        super();
        setFont(new Font("SansSerif", Font.BOLD, textSize));
        setHorizontalAlignment(JLabel.CENTER);
        setBorder(null);
    }

}
```

- Konstruktor JPasswordField: Ada sebuah konstruktor dalam kelas JPasswordField yang menerima satu parameter bertipe int, yaitu textSize. Konstruktor ini memanggil konstruktor dari kelas induknya (javax.swing.JPasswordField) menggunakan kata kunci super().
- Pengaturan Font: Konstruktor ini kemudian mengatur font teks pada objek JPasswordField menggunakan metode setFont(). Font yang digunakan adalah SansSerif, dengan gaya tebal (Font.BOLD), dan ukuran sesuai dengan nilai yang diterima dari parameter textSize.
- Pengaturan Horizontal Alignment: Metode setHorizontalAlignment() digunakan untuk mengatur perataan horizontal teks dalam JPasswordField. Dalam kode ini, perataan horizontal diatur ke JLabel.CENTER, yang berarti teks akan diperatakan secara horizontal di tengah.
- Pengaturan Border: Terakhir, metode setBorder() dipanggil untuk mengatur batas (border) pada JPasswordField menjadi null, sehingga tidak ada border yang ditampilkan.

3.1.1.8. JTable.java

```
public JTable(String[][] data, String[] header, Color color1, Color color2) {  
    super(data, header);  
    setRowHeight(40);  
    setBackground(null);
```

Terdapat 4 parameter yang diterima dari konstruktor kelas JTable, yaitu data aray dua dimensi yang berisi data untuk tabel, header untuk array yang berisi nama kolom tabel, color1 dan color2 untuk warna yang digunakan pada latar belakang. Pada konstruktor superclass dipanggil untuk menginisialisasi tabel dengan data dan header yang diberikan.

```
DefaultTableCellRenderer cellRenderer = new DefaultTableCellRenderer() {  
    @Override  
    public Component getTableCellRendererComponent(JTable table, Object value,  
        boolean isSelected,  
        boolean hasFocus, int row, int column) {
```

Merupakan bagian objek yang digunakan untuk mengatur penampilan sel dalam tabel. Metode di overwrite pada getTableCellRendererComponent() untuk mengatur perataan teks, font, latar belakang, dan border sel. Selain itu latar belakang sel diatur bergantung pada indeks baris.

```
DefaultTableCellRenderer headerRenderer = new DefaultTableCellRenderer() {  
    @Override  
    public Component getTableCellRendererComponent(JTable table, Object value,  
        boolean isSelected,  
        boolean hasFocus, int row, int column) {
```

Mirip dengan sebelumnya, namun ini untuk mengatur penampilan header tabel. Metode nya di overwrite untuk perataan, font, warna latar belakang, warna teks dan border header.

```
getTableHeader().setDefaultRenderer(headerRenderer);  
getTableHeader().setBorder(BorderFactory.createMatteBorder(2, 2, 1, 2, color1));  
setBorder(BorderFactory.createMatteBorder(1, 2, 2, 2, color1));  
setGridColor(color1);  
setRowSelectionAllowed(false);
```

Bagian pengaturan tambahan untuk tabel, dalam mengatur render, border, warna grid, dan membatasi seleksi pada baris tabel.

3.1.1.9. JTextField.java

```
package Model;

import java.awt.Font;import javax.swing.JLabel;

@SuppressWarnings("serial")      At least one of the problems in
public class JTextField extends javax.swing.JTextField {

    public JTextField(int textSize) {
        super();
        setFont(new Font("SansSerif", Font.BOLD, textSize));
        setHorizontalAlignment(JLabel.CENTER);
        setBorder(null);
    }

}
```

Memanggil konstruktor superclass untuk menginisialisasi field teks. Font teks yang diatur menggunakan setFont dengan font dan gaya tebal, dan ukuran sesuai dengan nilai dari parameter textSize. Perataan horizontal teks. Border field diatur menjadi null sehingga tidak ada border yang ditampilkan.

3.1.1.10. Operation.java

```
package Model;

import javax.swing.JFrame;

public interface Operation {

    public void operation(Database database, JFrame f, User user);

}
```

File ini bertujuan untuk menyediakan kontrak umum untuk berbagai operasi yang dilakukan dalam aplikasi. Kelas yang mengimplementasikan antarmuka harus menyediakan implementasi untuk metode operation yang sesuai dengan kebutuhan aplikasi.

3.1.1.11. Rent.java

Rent.java bertanggung jawab dalam penyimpanan informasi tentang transaksi penyewaan, dan menyediakan metode dalam mengakses dan mengatur informasi.

```
public class Rent {  
  
    private int ID;  
    private User user;  
    private ps ps;  
    private LocalDateTime dateTime;  
    private int hours;  
    private double total;  
    private int status;  
    private DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-dd-MM HH:mm");
```

Deklarasi kelas untuk variabel anggota menyimpan informasi tentang transaksi penyewaan, seperti ID, user, ps yang disewa, waktu transaksi, durasi penyewaan, total biaya dan status transaksi.

```
public String getDate() {  
    return formatter.format(dateTime);  
}  
  
public LocalDateTime getLocalDate() {  
    return dateTime;  
}  
  
public void setDate(String dateString) {  
    this.dateTime = LocalDateTime.parse(dateString, formatter);  
}
```

Metode getDate untuk mendapatkan waktu transaksi dalam format tertentu. setDate untuk mengatur waktu transaksi dari string.

```
public String getStatusToString() {  
    long passedHours = ChronoUnit.HOURS.between(dateTime, LocalDateTime.now());  
    String status = "";  
    if (getStatus() != 1 && passedHours < getHours()) {  
        status = "Estimated";  
    } else if (getStatus() != 1 && passedHours > getHours()) {  
        status = "Delayed";  
    } else if (getStatus() == 1) {  
        status = "Returned";  
    }  
    return status;  
}  
  
public long getDelayedHours() {  
    long passedHours = ChronoUnit.HOURS.between(dateTime, LocalDateTime.now());  
    return passedHours - hours;  
}
```

Metode getStatusToString untuk mendapatkan status transaksi dalam bentuk string berdasarkan waktu real dan status penyewaan, dan juga untuk mengembalikan status penyewaan. getDelayedHours digunakan untuk mengembalikan jumlah jam terlambat jika penyewaan yang melebihi batas waktu.

```
public void setStatus(int status) {  
    this.status = status;  
}
```

Metode untuk mengatur status penyewaan. Nilai 0 menunjukkan status penyewaan sedang berlangsung, dan nilai 1 untuk penyewaan yang telah dikembalikan.

3.1.1.12. User.java

```
public abstract class User {  
  
    private int ID;  
    private String firstName;  
    private String lastName;  
    private String email;  
    private String phoneNumber;  
    private String password;  
  
    //Type: 0 ==> Client  
    //      1 ==> Admin  
    //      2 ==> DeletedClientAccount  
    //      3 ==> DeletedAdminAccount  
  
    public User() {}
```

Deklarasi kelas untuk kelas abstrak yang memiliki beberapa variabel anggota untuk menyimpan informasi tentang pengguna seperti ID, nama depan-belakang, email, nomor telepon dan kata sandi. Kelas ini memiliki konstruktor kosong yang digunakan untuk membuat objek User menjadi kosong.

```

public int getID() {
    return ID;
}

public void setID(int ID) {
    this.ID = ID;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastname() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

```

Metode Akses dan Mutasi dari serangkaian metode getter dan setter untuk mengakses dan mengatur nilai dari variabel anggota kelas User.

```
public abstract void showList(Database database, JFrame f);
```

Deklarasi dari metode abstrak showList. Metode yang mengharuskan setiap kelas turunan untuk memberikan implementasi. Metode ini menerima dua paramater, objek database dan JFrame. Tujuannya untuk menampilkan daftar sesuai dengan tipe pengguna.

3.1.1.13. ps.java

File ps.java bertanggung jawab untuk menyimpan informasi tentang ps dan menyediakan metode untuk mengakses dan mengatur informasi.

```
public class ps {  
  
    private int ID;  
    private String brand;  
    private String model;  
    private String color;  
    private int year;  
    private double price;  
    private int available;  
  
    // 0 ==> available  
    // 1 ==> Rented  
    // 2 ==> Deleted  
  
    public ps() {  
    }  
}
```

Deklarasi kelas yang memiliki variabel anggota untuk menyimpan informasi tentang ps, seperti ID, merek, model, warna, tahun, harga, dan ketersediaan. Kelas ini juga memiliki konstruktor kosong yang digunakan untuk membuat objek ps yang kosong.

```
public int getID() {
    return ID;
}

public void setID(int ID) {
    this.ID = ID;
}

public String getBrand() {
    return brand;
}

public void setBrand(String brand) {
    this.brand = brand;
}

public String getModel() {
    return model;
}

public void setModel(String model) {
    this.model = model;
}

public String getColor() {
    return color;
}

public void setColor(String color) {
    this.color = color;
}
```

Metode akses dan mutasi yang terdiri dari serangkaian metode getter dan setter untuk mengakses dan mengatur nilai dari variabel anggota kelas ps.

```
public int isAvailable() {  
    return available;  
}
```

Metode isAvailable adalah metode yang digunakan untuk memeriksa ketersediaan barang. Jika barang tersedia maka akan mengembalikan nilai 0, dan sebaliknya jika barang tidak dikembalikan bernilai 1.

```
public void setAvailable(int available) {  
    this.available = available;  
}
```

Metode setter untuk mengatur ketersediaan barang. Dengan nilai yang sama dengan diatas.

3.2. PENGUJIAN

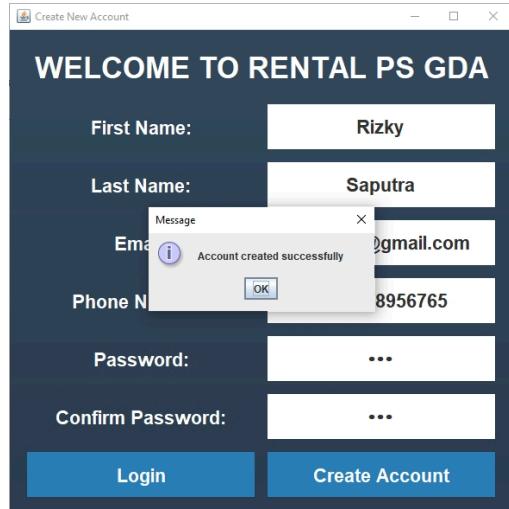
Pengujian program Java adalah proses penting dalam siklus pengembangan perangkat lunak untuk memastikan bahwa kode berjalan sesuai dengan yang diharapkan dan bebas dari bug.

3.2.1. Tampilan Pengujian

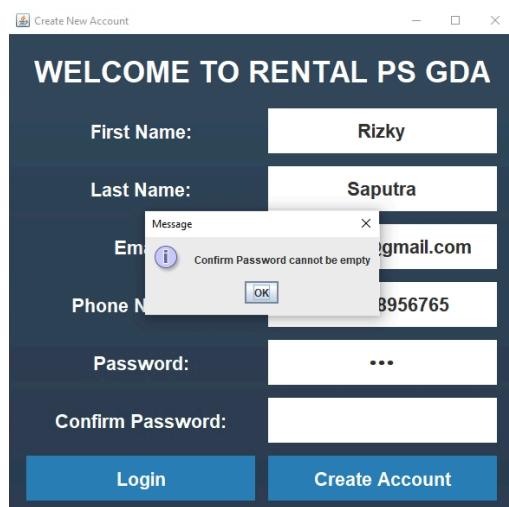
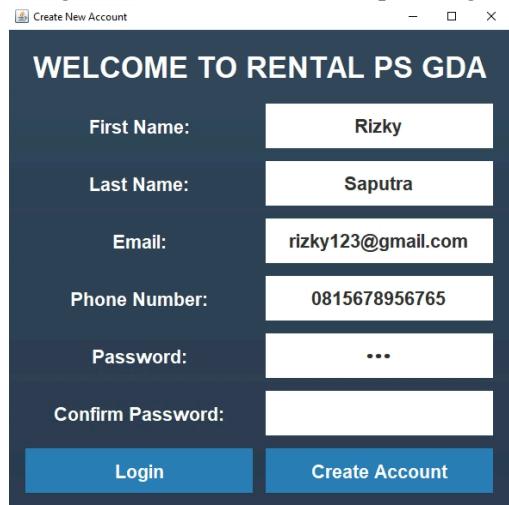
3.2.1.1. Registrasi

- Membuat akun baru sebagai customer

WELCOME TO RENTAL PS GDA	
First Name:	Rizky
Last Name:	Saputra
Email:	rizky123@gmail.com
Phone Number:	0815678956765
Password:	...
Confirm Password:	...
Login	Create Account

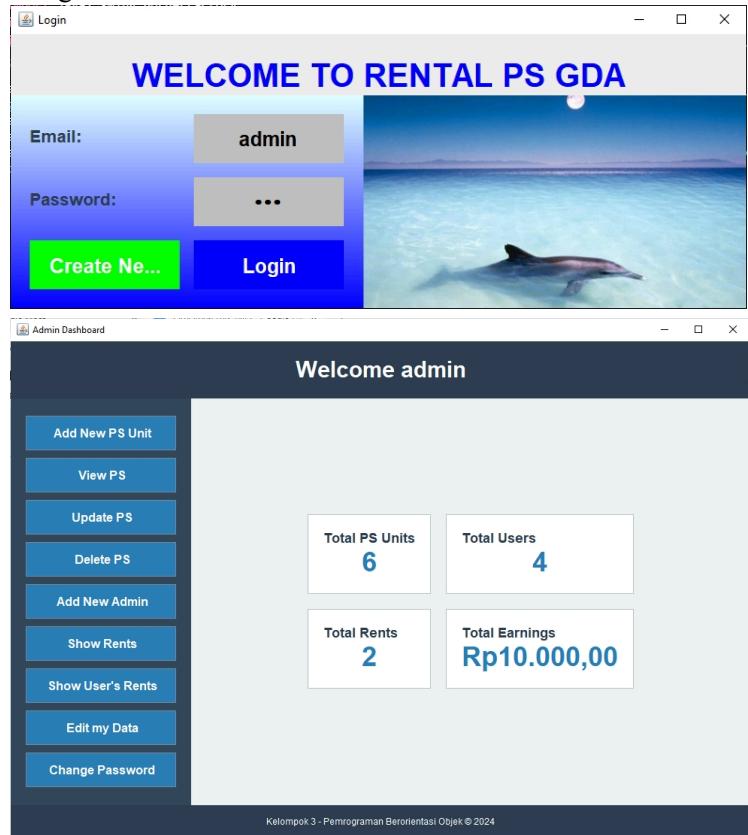


- b. Mengklik "Create Account" tanpa mengisi biodata.

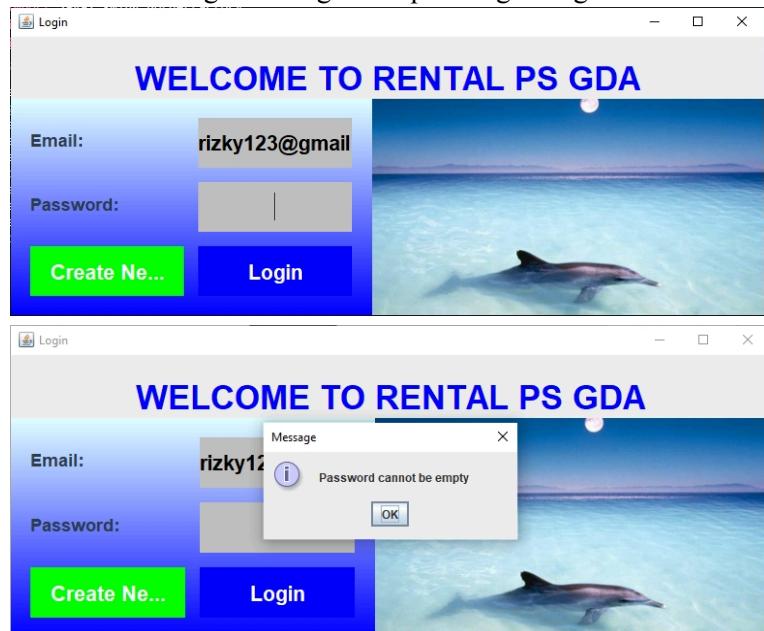


3.2.1.2. Login

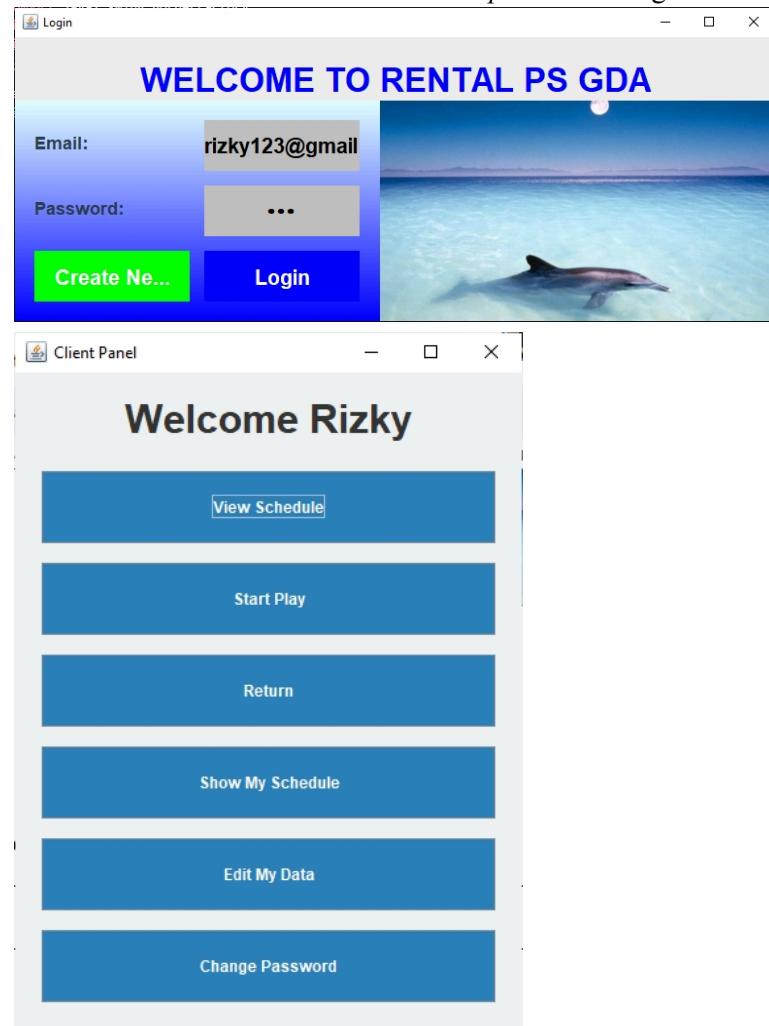
3.2.1.2.1. Memasukkan *username* sebagai Admin dan *password* sebagai Admin.



3.2.1.2.2. Mengklik "Login" tanpa mengisi bagian form.



3.2.1.2.3. Memasukkan *username* dan *password* dengan benar.

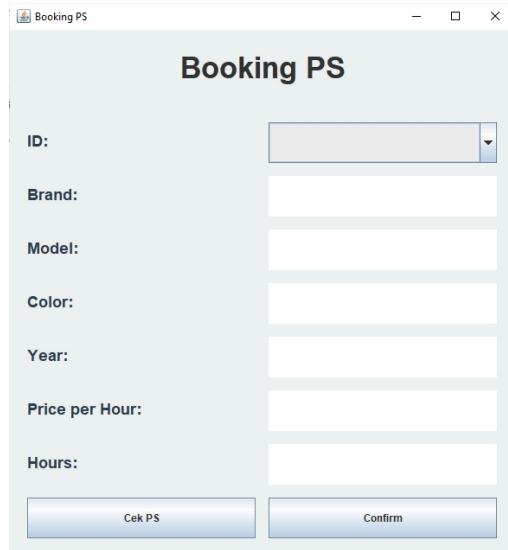


3.2.1.3. Dashboard

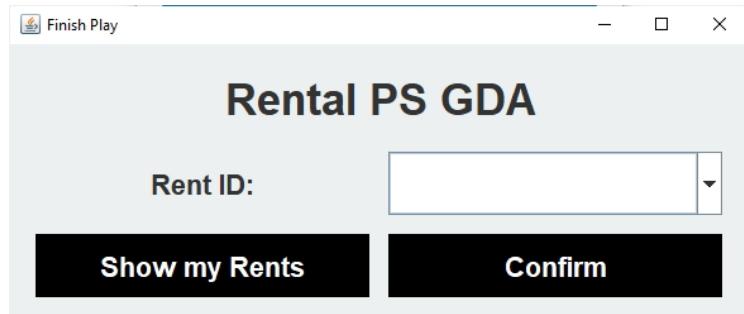
3.2.1.3.1. Mengklik menu “View PS”

All Playstations							
ID	Brand	Model	Color	Year	Price	Available	
0	Playstation	5	White	2023	Rp10000.0	Available	
1	Playstation	4	Black	2013	Rp5000.0	Available	
2	Playstation	3	Blue	2009	Rp3000.0	Available	
3	Playstation	2	Silver	2000	Rp2000.0	Available	
4	XBOX	Series X	Black	2016	Rp5000.0	Available	
5	XBOX	Series S	White	2019	Rp6000.0	Available	

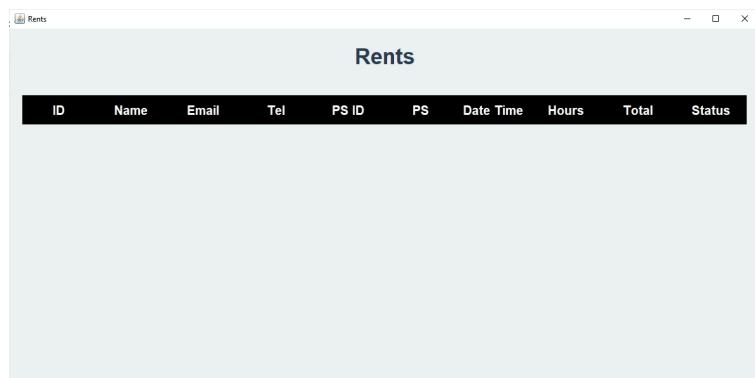
3.2.1.3.2. Mengklik menu “Start Play”



3.2.1.3.3. Mengklik menu “Return”



3.2.1.3.4. Mengklik menu “Show My Schedule”



3.2.1.3.5. Mengklik menu “Edit My Data”

The screenshot shows a window titled "Edit Data". It contains four input fields with the following data:

- First Name: Rizky
- Last Name: Saputra
- Email: rizky123@gmail.com
- Phone Number: 0815678956765

At the bottom are two buttons: a red "Cancel" button and a green "Confirm" button.

3.2.1.3.6. Mengklik menu “Change Password”

The screenshot shows a window titled "Change Password". It has three input fields:

- Old Password: (empty)
- New Password: (empty)
- Confirm Password: (empty)

At the bottom are two buttons: a red "Cancel" button and a green "Confirm" button.

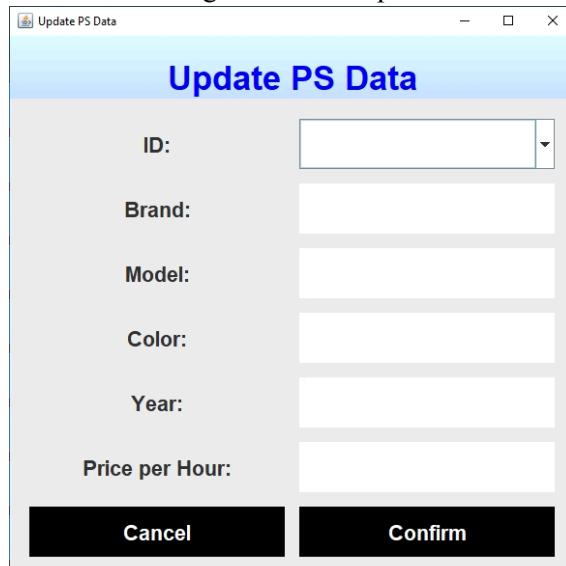
3.2.1.3.7. Mengklik menu “Add New PS”

The screenshot shows a window titled "Add New PS Unit". It has five input fields:

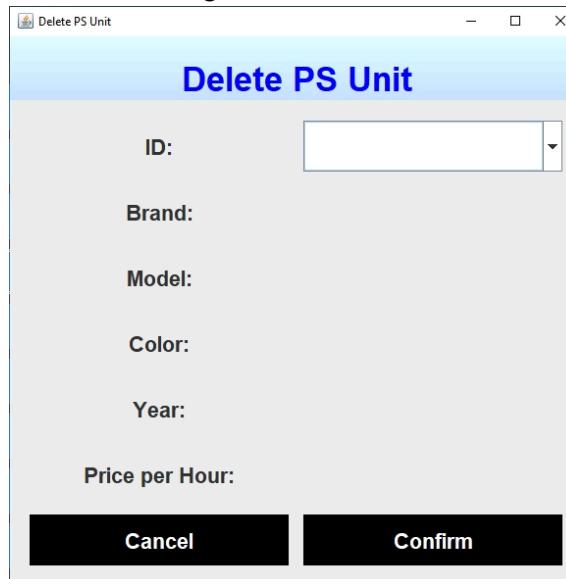
- Brand: (empty)
- Model: (empty)
- Color: (empty)
- Year: (empty)
- Price per Hour: (empty)

At the bottom are two buttons: a red "Cancel" button and a green "Confirm" button.

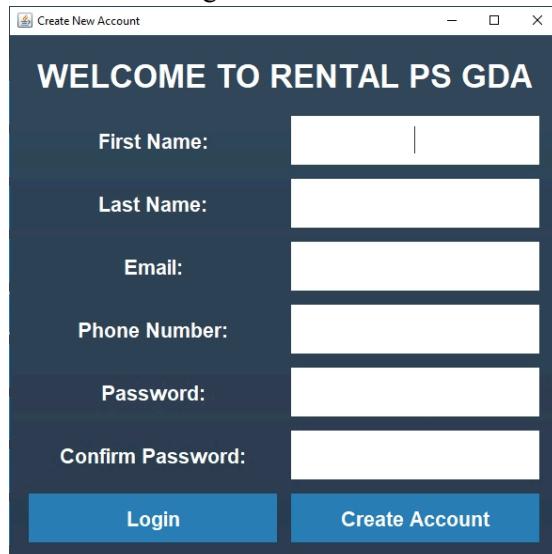
3.2.1.3.8. Mengklik menu “Update PS”



3.2.1.3.9. Mengklik menu “Delete PS”



3.2.1.3.10. Mengklik menu “New Admin”



3.2.1.3.11. Mengklik menu “Show Rents”

ID	Name	Email	Tel	ps ID	ps	Date Time	Hours	Total	Status
0	gege1 ge...	gege1	123	0	Playstatio...	2024-02-0...	2	Rp4000.0	Returned
1	gege1 ge...	gege1	123	3	Playstatio...	2024-02-0...	3	Rp6000.0	Returned

3.2.1.3.12. Mengklik menu “Show User’s Rents”

The window title is "Show User's Rents". It has a "User ID:" input field with a dropdown arrow. Below it are two buttons: "Show All Users" and "Confirm".

3.2.1.4. Penyewaan

- Customer melakukan penyewaan dengan klik ID Barang tanpa mengisi bagian *hours*

The window title is "Booking PS". It contains the following fields:
ID: 1
Brand: Playstation
Model: 4
Color: Black
Year: 2013
Price per Hour: Rp5000.0
Hours: (empty)
Buttons at the bottom: Cek PS and Confirm.



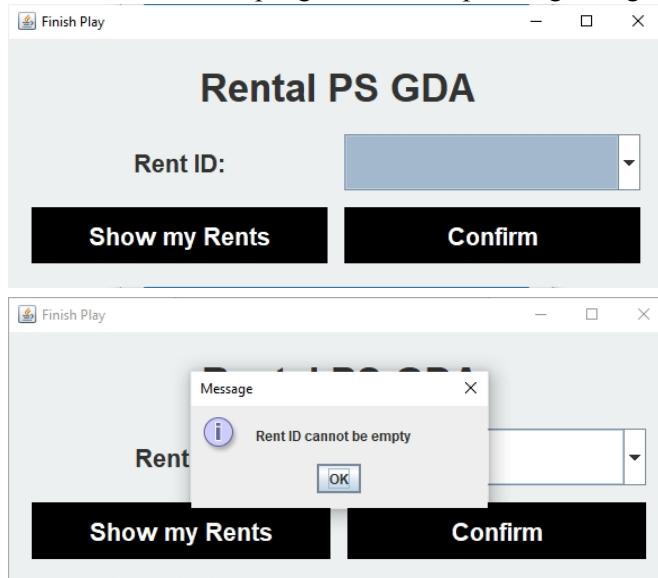
- b. Customer melakukan penyewaan dengan klik ID Barang dan mengisi bagian *hours* Melakukan penyewaan dengan mengklik button confirm.



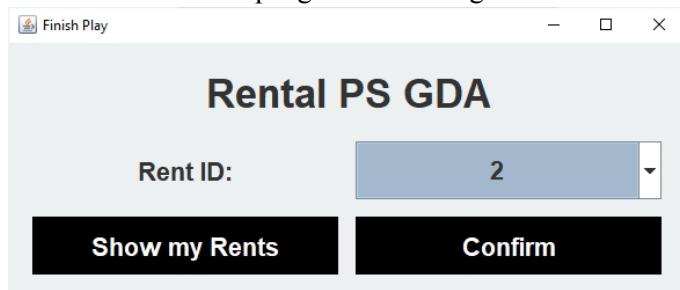


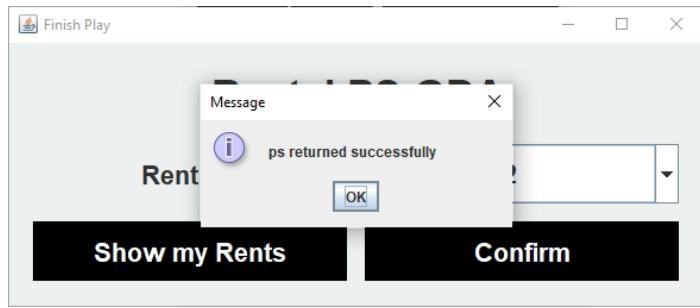
3.2.1.5. Pengembalian

- a. Customer melakukan pengembalian tanpa mengisi bagian Rent ID.



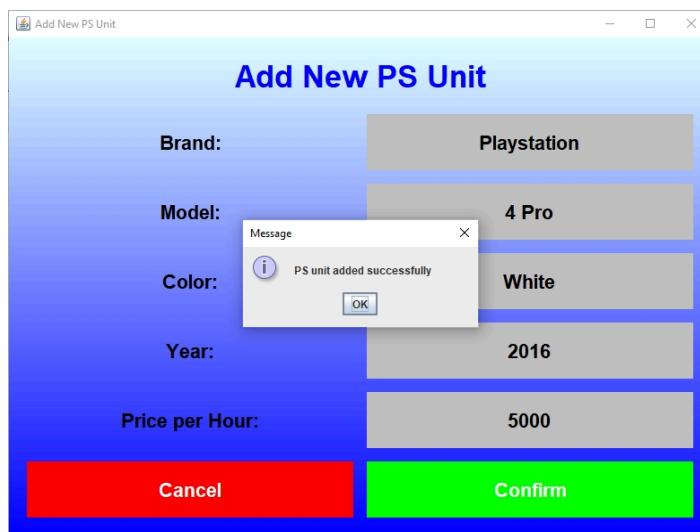
- b. Customer melakukan pengembalian dengan klik ID



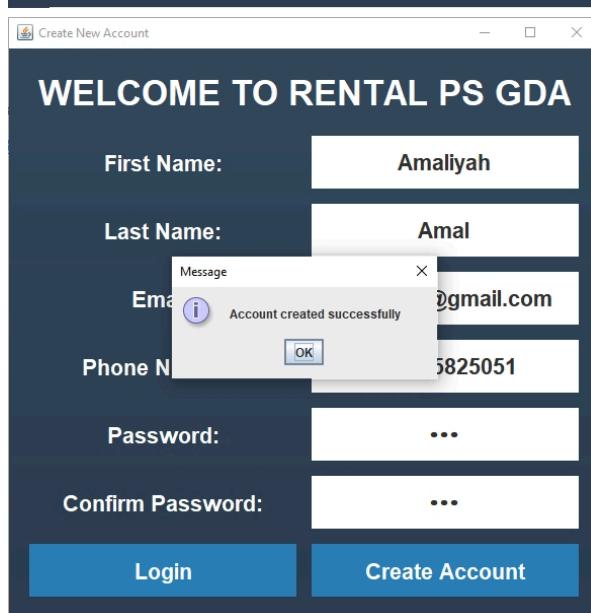
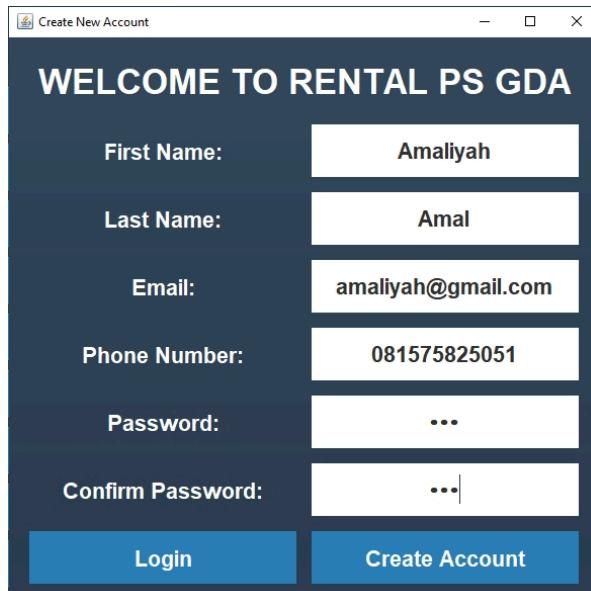


3.2.1.6. Input Data

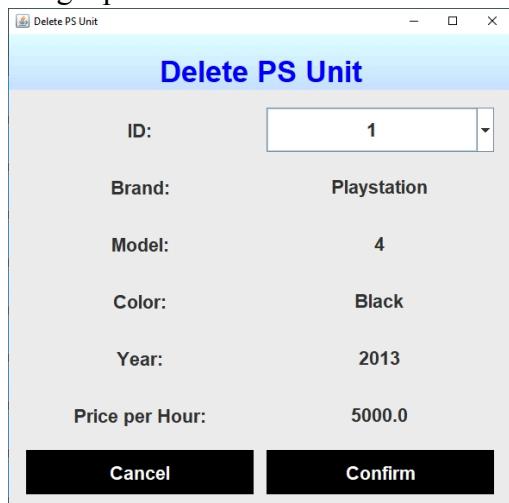
a. Menambahkan data PS

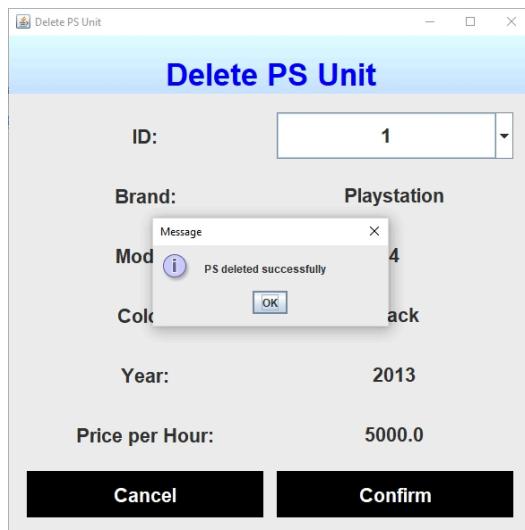


b. Menambahkan Akun Admin

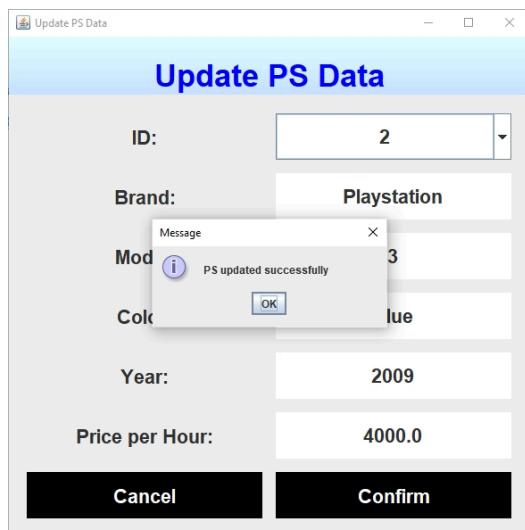
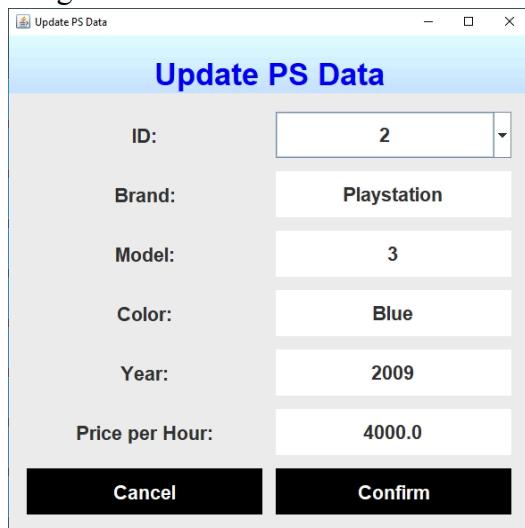


c. Menghapus data PS

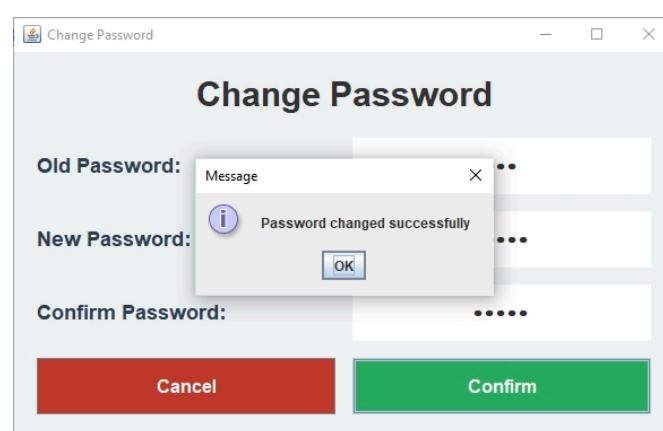
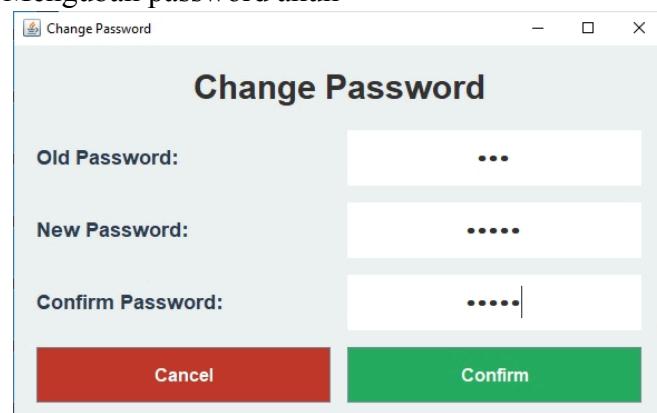




d. Mengedit data PS



e. Mengubah password akun



a. Hasil Pengujian

No.	Modul	Test	Hasil yang Diharapkan	Ket.
1.	<i>Registrasi</i>	Membuat akun baru sebagai customer	Menampilkan pesan berhasil, dan masuk ke bagian Dashboard	<i>Valid</i>
		Mengklik “Create Account” tanpa mengisi biodata	Menampilkan pesan “Cannot empty”	<i>Valid</i>
2.	<i>Login</i>	Memasukkan <i>username</i> sebagai Admin dan <i>password</i> sebagai Admin	Masuk ke menu <i>Dashboard</i>	<i>Valid</i>
		Mengklik “Login” tanpa mengisi bagian form	Menampilkan pesan “Cannot empty”	<i>Valid</i>
		Memasukkan <i>username</i> dan <i>password</i> dengan benar	Masuk ke menu <i>Dashboard</i>	<i>Valid</i>
3.	<i>Dashboard</i>	Mengklik menu “View PS”	Masuk ke frame data ps	<i>Valid</i>
		Mengklik menu “Start Play”	Masuk ke frame penyewaan	<i>Valid</i>
		Mengklik menu “Return PS”	Masuk ke frame pengembalian	<i>Valid</i>
		Mengklik menu “Show My Rents”	Masuk ke frame data customer yang sudah melakukan penyewaan	<i>Valid</i>
		Mengklik menu “Edit My Data”	Masuk ke frame Edit data	<i>Valid</i>
		Mengklik menu “Change Password”	Masuk ke frame Change Password	<i>Valid</i>
		Mengklik menu “Add New PS”	Masuk ke frame Add New PS	<i>Valid</i>
		Mengklik menu “Update PS”	Masuk ke frame Upadate PS	<i>Valid</i>
		Mengklik menu	Masuk ke frame	<i>Valid</i>

		“Delete PS”	Delete PS	
		Mengklik menu “New Admin”	Masuk ke frame New Admin	<i>Valid</i>
		Mengklik menu “Show Rents”	Masuk ke frame Show Rents	<i>Valid</i>
		Mengklik menu “Show User’s Rents”	Masuk ke frame Show User’s Rents	<i>Valid</i>
4.	Penyewaan	Customer melakukan penyewaan dengan klik ID Barang tanpa mengisi bagian <i>hours</i>	Menampilkan message “Cannot be empty”	<i>Valid</i>
		Customer melakukan penyewaan dengan klik ID Barang dan mengisi bagian <i>hours</i> Melakukan penyewaan dengan mengklik button confirm.	Menampilkan message Total biaya rental dan keluar dari frame penyewaan	<i>Valid</i>
5.	Pengembalian	Customer melakukan pengembalian tanpa mengisi bagian Rent ID	Menampilkan message “Cannot be empty”	<i>Valid</i>
		Customer melakukan pengembalian dengan klik ID	Menampilkan message “returned successfully” dan keluar dari frame Return/Pengembalian	<i>Valid</i>
6.	Input Data	Menambahkan data PS	Data berhasil ditambahkan dengan menampilkan message “added succesfully”	<i>Valid</i>
		Menambahkan Akun Admin	Akun baru berhasil ditambahkan dengan menampilkan message “created succesfully”	<i>Valid</i>
		Menghapus data PS	Data PS yang terpilih terhapus	<i>Valid</i>

		dengan menampilkan message “deleted successfully”	
	Mengedit data PS	Data berhasil diedit dengan menampilkan message “updated successfully”	<i>Valid</i>
	Mengedit data akun	Data berhasil diedit dengan menampilkan message “updated successfully”	<i>Valid</i>
	Mengubah password akun	Password berhasil diubah dengan menampilkan message “changed successfully”	<i>Valid</i>

BAB IV

KESIMPULAN

Dalam pengembangan aplikasi GUI Rental PS menggunakan bahasa pemrograman Java dengan pendekatan OOP, kelompok kami menyimpulkan bahwa implementasi konsep OOP memberikan banyak manfaat dan keuntungan. Konsep ini memudahkan dalam pengembangan dan pemeliharaan kode program dan pemisahan tugas serta tanggung jawab ke dalam objek-objek yang terpisah. Konsep *Encapsulation*, *Polymorphism*, *Package*, *Inheritance*, *Exception*, dan Tampilan Antarmuka juga memberikan fleksibilitas, mudah terpelihara dan membantu dalam pencapaian fungsionalitas serta kebutuhan pengguna yang lebih efektif.

Pada keseluruhan implementasi konsep OOP dalam pembuatan aplikasi GUI Rental PS menggunakan Java telah membawa manfaat yang signifikan. Konsep OOP memudahkan pengguna dalam pengembangannya yang lebih terstruktur, dan mudah diperluas. Selain itu, pendekatan OOP juga membantu mencapai tujuan fungsionalitas bagi pengguna dan pemilik rental.

Dengan menggunakan konsep OOP, pembuatan aplikasi GUI Rental PS menjadi lebih terorganisir dan efisien. Implementasi dari aplikasi ini juga membawa manfaat yang signifikan, termasuk dalam kemudahan pemeliharaan, fleksibilitas dalam pengaturan kode, dan fungsionalitas yang lebih efektif.

LAMPIRAN

Nama	NPM	Keterangan
Anisa Ayu Yandani	2210631250002	Survey Lapangan, Laporan, UML Usecase Diagram, Activity Diagram
Gerald Dustin Albert	2210631250011	Survey Lapangan, Implementasi kode bagian Admin, Folder Model, Database
Muhammad Rizky Saputra	2210631250021	Survey Lapangan, Implementasi kode bagian Client, Folder Controller
Amaliyah	2210631250042	Survey Lapangan, Laporan, UML Diagram Activity, Class Diagram