Smart ATM Application using Java Swing

1. **Problem Statement**
    - ATMs allow customers to deposit, withdraw, check balance, and manage accounts.
    - This project simulates an ATM system with a **graphical user interface** (GUI) instead of console.
    - It provides **user features** (deposit, withdraw, balance check, PIN change, history) and **admin features** (block/unblock, delete users, view history).

2. **Technologies Used**
    - **Java Swing** – For GUI components (JFrame, JButton, JTextField, etc.).
    - **AWT & Event Handling** – For layout and handling button actions.
    - **Serialization** – To store account data in a file (accounts.db).
    - **Collections (HashMap, List)** – For managing multiple accounts and their history.

3. **Features Implemented**

    User Features

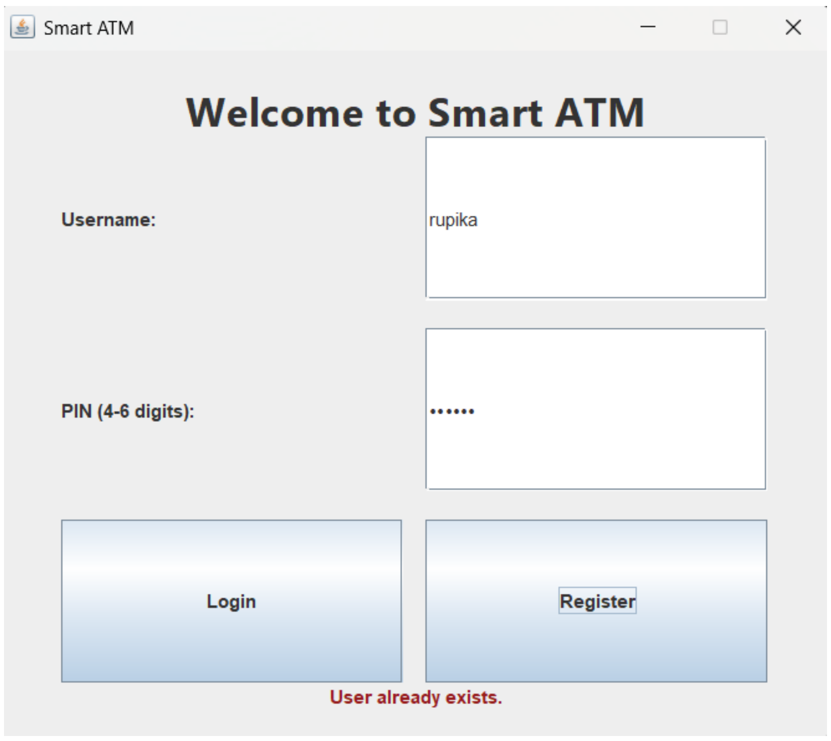
    - Register a new account with username & PIN.
    - Login using username and PIN.
    - Deposit and Withdraw money.
    - Check current balance.
    - View last 30 transactions in history.
    - Change PIN securely.
    - Delete account with PIN confirmation.

Admin Features

- Admin login (username: admin, PIN: 0000).
- View all user accounts.
- View transaction history of users.
- Block or unblock accounts.
- Delete user accounts.

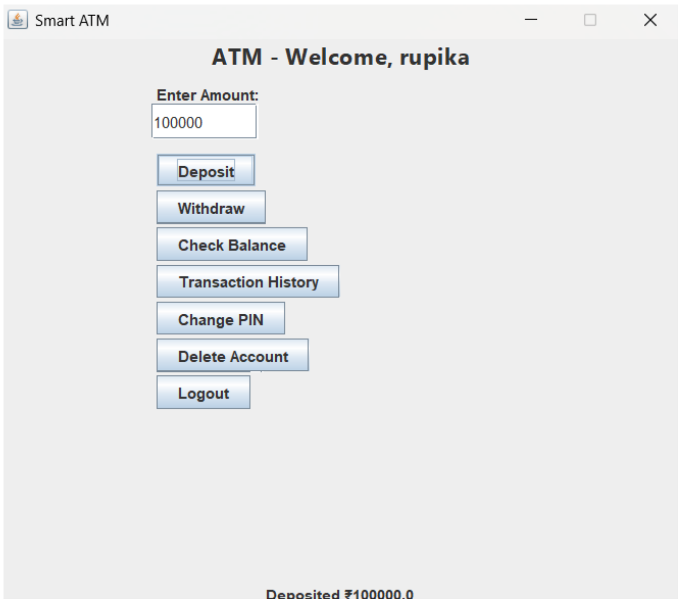
4. Data Handling

- Account details (username, PIN, balance, history, block status) are stored in a serialized HashMap inside accounts.db.
- This ensures data is persistent even if the app is closed.
- Each account keeps a transaction history with timestamps (max 30 entries).



User already exists.

Deposit

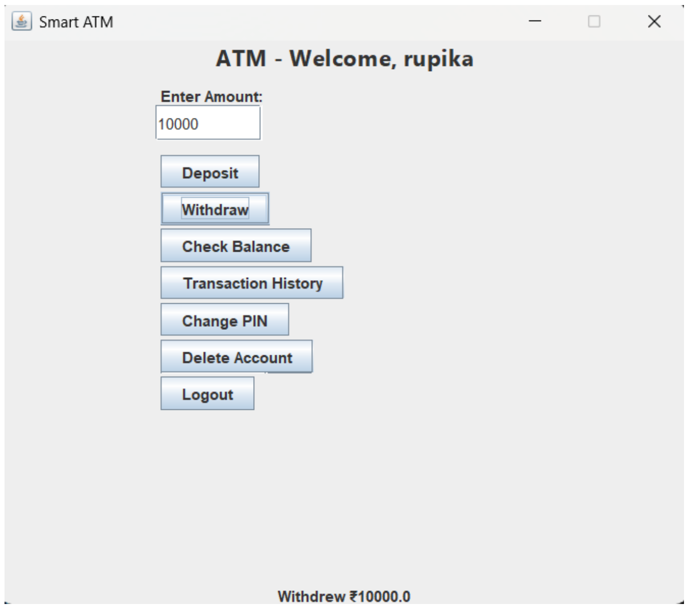


ATM - Welcome, rupika

Enter Amount: 100000

Deposited ₹100000.0

Withdraw

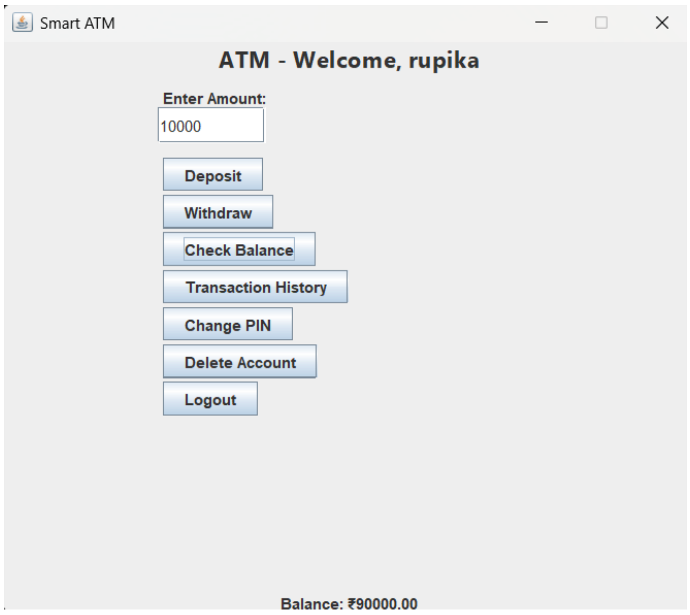


ATM - Welcome, rupika

Enter Amount: 10000

Withdrew ₹10000.0

Check Balance

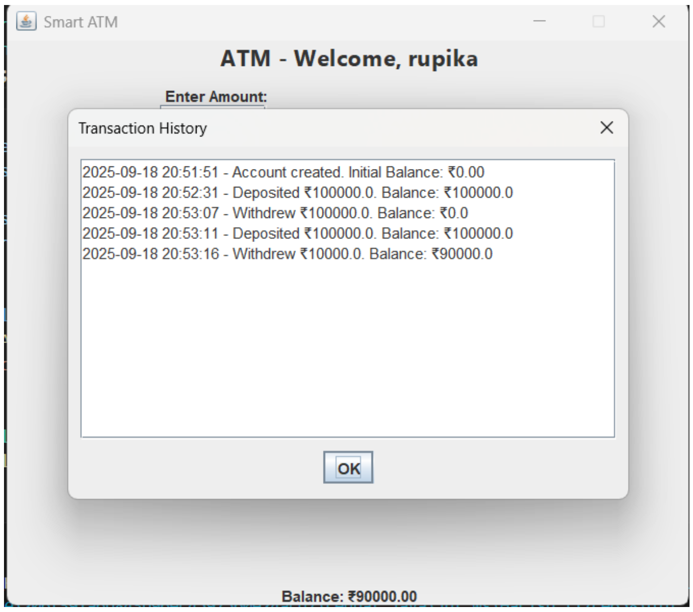


ATM - Welcome, rupika

Enter Amount: 10000

Balance: ₹90000.00

Transcation History



ATM - Welcome, rupika

Enter Amount:

```
2025-09-18 20:51:51 - Account created. Initial Balance: ₹0.00
2025-09-18 20:52:31 - Deposited ₹100000.0. Balance: ₹100000.0
2025-09-18 20:53:07 - Withdrew ₹100000.0. Balance: ₹0.0
2025-09-18 20:53:11 - Deposited ₹100000.0. Balance: ₹100000.0
2025-09-18 20:53:16 - Withdrew ₹10000.0. Balance: ₹90000.0
```

Balance: ₹90000.00

Code:

```
import javax.swing.*;
import java.awt.*;

import java.io.*;

import java.text.SimpleDateFormat;
import java.util.*;
import java.util.List;

class Account implements Serializable {
    private static final long serialVersionUID = 1L;
    private String username;
    private String pin;
    private double balance;
    private boolean isBlocked;
    private List<String> history;

    public Account(String username, String pin) {
        this.username = username;
        this.pin = pin;
        this.balance = 0.0;
        this.isBlocked = false;
        this.history = new ArrayList<>();
        addHistory("Account created. Initial Balance: ₹0.00");
    }

    public synchronized boolean validatePin(String inputPin) {
        return Objects.equals(pin, inputPin);
    }
```

```
public String getUsername() {
    return username;
}

public synchronized double getBalance() {
    return balance;
}

public synchronized boolean deposit(double amt) {
    if (amt > 0) {
        balance += amt;
        addHistory("Deposited ₹" + amt + ". Balance: ₹" + balance);
        return true;
    }
    return false;
}

public synchronized boolean withdraw(double amt) {
    if (amt > 0 && amt <= balance) {
        balance -= amt;
        addHistory("Withdrew ₹" + amt + ". Balance: ₹" + balance);
        return true;
    }
    return false;
}

public synchronized void addHistory(String desc) {
    String ts = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
    history.add(ts + " - " + desc);
    if (history.size() > 30)
```

```
            history = history.subList(history.size() - 30, history.size());
        }

        public List<String> getHistory() {
            return new ArrayList<>(history);
        }

        public synchronized boolean changePin(String old, String newPin) {
            if (validatePin(old) && newPin.matches("\\d{4,6}")) {
                this.pin = newPin;
                addHistory("PIN changed.");
                return true;
            }
            return false;
        }

        public synchronized void block() {
            isBlocked = true;
            addHistory("Account blocked by admin.");
        }

        public synchronized void unblock() {
            isBlocked = false;
            addHistory("Account unblocked.");
        }

        public synchronized boolean isBlocked() {
            return isBlocked;
        }
    }
```

```java
// A utility class for saving and loading account data
class AccountStorage {
    private static final String FILE = "accounts.db";

    @SuppressWarnings("unchecked")
    public static HashMap<String, Account> loadAccounts() {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE))) {
            Object obj = in.readObject();
            if (obj instanceof HashMap<?, ?>) {
                // Ensure the map is of the correct generic type
                HashMap<?, ?> rawMap = (HashMap<?, ?>) obj;
                boolean valid = rawMap.keySet().stream().allMatch(k -> k instanceof String)
                        && rawMap.values().stream().allMatch(v -> v instanceof Account);
                if (valid) {
                    return (HashMap<String, Account>) rawMap;
                }
            }
            return new HashMap<>();
        } catch (Exception e) {
            return new HashMap<>();
        }
    }

    public static void saveAccounts(HashMap<String, Account> accounts) {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(FILE))) {
            out.writeObject(accounts);
        } catch (Exception e) {
            // ignore
        }
    }
}
```

```
public class SmartATMApp extends JFrame {
    private static final String ADMIN_USER = "admin";
    private static final String ADMIN_PIN = "0000";
    private HashMap<String, Account> accounts;
    private Account current;
    private String currentRole = "user"; // or 'admin'

    // Input fields
    private JTextField userField;
    private JPasswordField pinField;
    private JPasswordField pin2Field, newPinField, oldPinField;
    private JTextField amountField;
    private JLabel statusLabel;

    public SmartATMApp() {
        accounts = AccountStorage.loadAccounts();
        // Ensure admin account exists
        if (!accounts.containsKey(ADMIN_USER)) {
            Account admin = new Account(ADMIN_USER, ADMIN_PIN);
            accounts.put(ADMIN_USER, admin);
            AccountStorage.saveAccounts(accounts);
        }

        setTitle("Smart ATM");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(550, 480);
        setLocationRelativeTo(null);
        setResizable(false);
        showWelcomeScreen();
        setVisible(true);
```

```
}

private void showWelcomeScreen() {
    JPanel p = new JPanel(new BorderLayout());
    p.setBorder(BorderFactory.createEmptyBorder(20, 40, 20, 40));
    JLabel title = new JLabel("Welcome to Smart ATM", SwingConstants.CENTER);
    title.setFont(new Font("Segoe UI", Font.BOLD, 26));
    p.add(title, BorderLayout.NORTH);

    JPanel form = new JPanel(new GridLayout(3, 2, 15, 18));
    userField = new JTextField();
    pinField = new JPasswordField();
    JButton loginBtn = new JButton("Login");
    JButton regBtn = new JButton("Register");

    form.add(new JLabel("Username:"));
    form.add(userField);
    form.add(new JLabel("PIN (4-6 digits):"));
    form.add(pinField);
    form.add(loginBtn);
    form.add(regBtn);

    p.add(form, BorderLayout.CENTER);
    statusLabel = new JLabel(" ", SwingConstants.CENTER);
    statusLabel.setForeground(new Color(150, 30, 30));
    p.add(statusLabel, BorderLayout.SOUTH);
    setContentPane(p);
    revalidate();

    loginBtn.addActionListener(e -> login());
    regBtn.addActionListener(e -> register());
```

```
}

private void login() {
   String u = userField.getText().trim();
   String p = String.valueOf(pinField.getPassword()).trim();
   if (u.isEmpty() || p.isEmpty()) {
      statusLabel.setText("Fill all fields.");
      return;
   }
   Account acc = accounts.get(u);
   if (acc != null && acc.validatePin(p)) {
      if (u.equals(ADMIN_USER)) {
         currentRole = "admin";
         showAdminMenu();
      } else if (acc.isBlocked()) {
         statusLabel.setText("Account is blocked.");
      } else {
         currentRole = "user";
         current = acc;
         showATMMenu();
      }
   } else {
      statusLabel.setText("Invalid username or PIN.");
   }
}

private void register() {
   String u = userField.getText().trim();
   String p = String.valueOf(pinField.getPassword()).trim();
   if (!u.matches("[a-zA-Z0-9_]{3,12}")) {
      statusLabel.setText("Username: 3-12 chars, letters/numbers/_ only.");
```

```
            return;
        }
        if (!p.matches("\\d{4,6}")) {
            statusLabel.setText("PIN must be 4-6 digits.");
            return;
        }
        if (accounts.containsKey(u)) {
            statusLabel.setText("User already exists.");
            return;
        }
        Account acc = new Account(u, p);
        accounts.put(u, acc);
        AccountStorage.saveAccounts(accounts);
        statusLabel.setText("Registered! Please login.");
    }

    private void showATMMenu() {
        JPanel p = new JPanel(new BorderLayout());
        JLabel head = new JLabel("ATM - Welcome, " + current.getUsername(),
SwingConstants.CENTER);
        head.setFont(new Font("Segoe UI", Font.BOLD, 18));
        p.add(head, BorderLayout.NORTH);

        JPanel center = new JPanel();
        center.setLayout(new BoxLayout(center, BoxLayout.Y_AXIS));
        center.setBorder(BorderFactory.createEmptyBorder(10, 120, 10, 120));

        amountField = new JTextField();
        amountField.setMaximumSize(new Dimension(160, 28));

        JButton dep = new JButton("Deposit");
        JButton wdr = new JButton("Withdraw");

        JButton bal = new JButton("Check Balance");

        JButton hist = new JButton("Transaction History");

        JButton chgPinBtn = new JButton("Change PIN");

        JButton delBtn = new JButton("Delete Account");

        JButton logout = new JButton("Logout");


        statusLabel = new JLabel("Status: Ready", SwingConstants.CENTER);


        dep.addActionListener(e -> {

            performDeposit();

        });

        wdr.addActionListener(e -> {

            performWithdraw();

        });

        bal.addActionListener(e -> statusLabel.setText("Balance: ₹" + String.format("%.2f",
current.getBalance())));

        hist.addActionListener(e -> showHistory());

        chgPinBtn.addActionListener(e -> showChangePinDialog());

        delBtn.addActionListener(e -> showDeleteAccountDialog());

        logout.addActionListener(e -> {

            current = null;

            showWelcomeScreen();

        });


        center.add(new JLabel("Enter Amount:"));

        center.add(amountField);

        center.add(Box.createRigidArea(new Dimension(0, 11)));

        center.add(dep);

        center.add(Box.createRigidArea(new Dimension(0, 3)));

        center.add(wdr);
```

```
    center.add(Box.createRigidArea(new Dimension(0, 3)));
    center.add(bal);
    center.add(Box.createRigidArea(new Dimension(0, 3)));
    center.add(hist);
    center.add(Box.createRigidArea(new Dimension(0, 3)));
    center.add(chgPinBtn);
    center.add(Box.createRigidArea(new Dimension(0, 3)));
    center.add(delBtn);
    center.add(Box.createRigidArea(new Dimension(0, 3)));
    center.add(logout);

    p.add(center, BorderLayout.CENTER);
    p.add(statusLabel, BorderLayout.SOUTH);

    setContentPane(p);
    revalidate();
  }

  private void performDeposit() {
    try {
      double amt = Double.parseDouble(amountField.getText());
      if (amt <= 0) {
        statusLabel.setText("Enter a positive amount.");
        return;
      }
      if (current.deposit(amt)) {
        statusLabel.setText("Deposited ₹" + amt);
        AccountStorage.saveAccounts(accounts);
      } else {
        statusLabel.setText("Deposit failed.");
      }
```

```
        } catch (NumberFormatException ex) {
            statusLabel.setText("Invalid amount.");
        }
    }

    private void performWithdraw() {
        try {
            double amt = Double.parseDouble(amountField.getText());
            if (amt <= 0) {
                statusLabel.setText("Enter a positive amount.");
                return;
            }
            if (amt > current.getBalance()) {
                statusLabel.setText("Insufficient balance.");
                return;
            }
            if (current.withdraw(amt)) {
                statusLabel.setText("Withdrew ₹" + amt);
                AccountStorage.saveAccounts(accounts);
            } else {
                statusLabel.setText("Withdraw failed.");
            }
        } catch (NumberFormatException ex) {
            statusLabel.setText("Invalid amount.");
        }
    }

    private void showHistory() {
        List<String> logs = current.getHistory();
        JTextArea ta = new JTextArea();
        logs.forEach(x -> ta.append(x + "\n"));
        ta.setEditable(false);
        JScrollPane scroll = new JScrollPane(ta);
        scroll.setPreferredSize(new Dimension(420, 220));
        JOptionPane.showMessageDialog(this, scroll, "Transaction History",
JOptionPane.PLAIN_MESSAGE);
    }


    private void showChangePinDialog() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 2, 10, 12));
        oldPinField = new JPasswordField();
        newPinField = new JPasswordField();
        panel.add(new JLabel("Old PIN:"));
        panel.add(oldPinField);
        panel.add(new JLabel("New PIN:"));
        panel.add(newPinField);
        int result = JOptionPane.showConfirmDialog(this, panel, "Change PIN",
JOptionPane.OK_CANCEL_OPTION);
        if (result == JOptionPane.OK_OPTION) {
            String oldPin = new String(oldPinField.getPassword()).trim();
            String newPin = new String(newPinField.getPassword()).trim();
            if (!newPin.matches("\\d{4,6}")) {
                JOptionPane.showMessageDialog(this, "PIN must be 4-6 digits.");
                return;
            }
            if (current.changePin(oldPin, newPin)) {
                AccountStorage.saveAccounts(accounts);
                JOptionPane.showMessageDialog(this, "PIN changed successfully.");
            } else {
                JOptionPane.showMessageDialog(this, "Old PIN incorrect.");
            }
        }
```

```
}

private void showDeleteAccountDialog() {
   JPasswordField pf = new JPasswordField();
   int okCxl = JOptionPane.showConfirmDialog(this, pf,
         "Enter PIN to confirm account deletion.", JOptionPane.OK_CANCEL_OPTION);
   if (okCxl == JOptionPane.OK_OPTION) {
      String pin = new String(pf.getPassword()).trim();
      if (current.validatePin(pin)) {
         accounts.remove(current.getUsername());
         AccountStorage.saveAccounts(accounts);
         JOptionPane.showMessageDialog(this, "Account deleted. Bye!");
         current = null;
         showWelcomeScreen();
      } else {
         JOptionPane.showMessageDialog(this, "PIN incorrect!");
      }
   }
}

// =====Admin area ====
private void showAdminMenu() {
   JPanel p = new JPanel(new BorderLayout());
   JLabel head = new JLabel("ADMIN DASHBOARD", SwingConstants.CENTER);
   head.setFont(new Font("Segoe UI", Font.BOLD, 18));
   p.add(head, BorderLayout.NORTH);

   DefaultListModel<String> model = new DefaultListModel<>();
   JList<String> userList = new JList<>(model);

   accounts.entrySet().stream()
```

```
        .filter(e -> !e.getKey().equals(ADMIN_USER))
        .forEach(e -> {
            Account a = e.getValue();
            String label = e.getKey() + (a.isBlocked() ? " (BLOCKED)" : "");
            model.addElement(label);
        });

userList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
JScrollPane scroll = new JScrollPane(userList);
scroll.setPreferredSize(new Dimension(340, 240));

JButton vhistBtn = new JButton("View History");
JButton blockBtn = new JButton("Block/Unblock");
JButton delBtn = new JButton("Delete Account");
JButton logoutBtn = new JButton("Logout");

JPanel bottom = new JPanel();
bottom.add(vhistBtn);
bottom.add(blockBtn);
bottom.add(delBtn);
bottom.add(logoutBtn);

p.add(scroll, BorderLayout.CENTER);
p.add(bottom, BorderLayout.SOUTH);

vhistBtn.addActionListener(e -> adminViewHistory(userList));
blockBtn.addActionListener(e -> adminBlockUnblock(userList));
delBtn.addActionListener(e -> adminDelete(userList));
logoutBtn.addActionListener(e -> {
    currentRole = "user";
    showWelcomeScreen();
```

```
        });

        setContentPane(p);
        revalidate();
    }

    private void adminViewHistory(JList<String> userList) {
        String sel = userList.getSelectedValue();
        if (sel == null)
            return;
        String user = sel.replace(" (BLOCKED)", "");
        Account acc = accounts.get(user);
        if (acc == null)
            return;

        List<String> logs = acc.getHistory();
        JTextArea ta = new JTextArea();
        logs.forEach(x -> ta.append(x + "\n"));
        ta.setEditable(false);
        JScrollPane scroll = new JScrollPane(ta);
        scroll.setPreferredSize(new Dimension(420, 220));
        JOptionPane.showMessageDialog(this, scroll, "History of " + user,
JOptionPane.PLAIN_MESSAGE);
    }

    private void adminBlockUnblock(JList<String> userList) {
        String sel = userList.getSelectedValue();
        if (sel == null)
            return;
        String user = sel.replace(" (BLOCKED)", "");
        Account acc = accounts.get(user);
        if (acc == null)
            return;
        acc.block();
        if (sel.contains("BLOCKED")) {
            acc.unblock();
        } else {
            acc.block();
        }
        AccountStorage.saveAccounts(accounts);
        showAdminMenu();
    }

    private void adminDelete(JList<String> userList) {
        String sel = userList.getSelectedValue();
        if (sel == null)
            return;
        String user = sel.replace(" (BLOCKED)", "");
        int ok = JOptionPane.showConfirmDialog(this, "Delete user " + user + "?", "Confirm",
JOptionPane.YES_NO_OPTION);
        if (ok == JOptionPane.YES_OPTION) {
            accounts.remove(user);
            AccountStorage.saveAccounts(accounts);
            showAdminMenu();
        }
    }

    // MAIN
    public static void main(String[] args) {
        SwingUtilities.invokeLater(SmartATMApp::new);
    }
}
```