CS514 - Fall 2025

HW/Assignment 3: My Payment System

Due Date: Wednesday, Oct 15 @9pm SF time

Instructor: Nancy Stevens

OVERVIEW

Design and implement a payment tracking system called ***MyPaymentSystem***. The program manages contacts (friends or family), their preferred payment systems (like PayPal, Venmo), and payment records. It should support basic operations like paying a contact, updating payment preferences, and generating reports.

Note: This is a simulation - you will not be connecting to any live payment systems.

Learning Objectives:

This project reinforces key object-oriented programming concepts:

- **Inheritance**: Creating class hierarchies with shared and specialized behavior
- **Polymorphism**: Using parent class references to interact with child class objects
- **Interfaces**: Defining contracts that multiple classes implement
- File I/O and date/time handling

Requirements:

Main Menu

Create a program called `MyPaymentSystem.java` that presents the following menu:

```
1. Pay Contact
2. Print all Payments
3. Update Preferred Payment System
4. Print Contact Information
0. Exit
```

Feature 1: Pay Contact (Menu Option 1)

1. Prompt user to search for a contact using a 10-digit phone number
2. Display the contact's information and preferred payment system
3. Prompt for payment amount
4. Confirm payment details with user
5. Process payment through the contact's preferred payment system

6. Append transaction to **`payments.txt`**
7. Return to main menu

Feature 2: Print All Payments (Menu Option 2)

1. Read and display all payments from **`payments.txt`**
2. Format output clearly showing: recipient, amount, payment system, and timestamp
3. Return to main menu

Feature 3: Update Preferred Payment System (Menu Option 3)

1. Prompt user to search for a contact
2. Display current payment system
3. Prompt for new payment system name
4. Update **`contacts.txt`** with the new preference
5. Confirm the change
6. Return to main menu

Feature 4: Print Contact Information (Menu Option 4)

1. Prompt user to search for a contact
2. Display the contact's full details including contact type (Friend/Family)
3. Demonstrate polymorphic behavior by calling type-specific methods
4. Return to main menu

Object-Oriented Design Requirements

1. Create a Base Class: Person

Fields:

`firstName, lastName, phoneNumber, preferredPaymentSystem, contactType`

Constructor: Initializes all fields

Methods:

- Getters and setters
- `getFullName()` - returns first and last name
- `toString()` - returns formatted string representation
- `getContactType()` - returns the type of contact ("Friend" or "Family")
- `getRelationshipMessage()` - returns a generic message that can be overridden

Purpose: Provides common functionality that both Friend and Family contacts share.

2. Subclasses: `Friend` and `Family`

- **Friend class**:
    - Extends **`Person`**
    - Constructor calls **`super()`** and sets **`contactType`** to "Friend"
    - **Overrides `getRelationshipMessage()`** to return a friendship-related message
    - Add a field for **`friendshipYears`** (optional enhancement)

- **Family class**:
    - Extends **`Person`**
    - Constructor calls **`super()`** and sets **`contactType`** to "Family"
    - **Overrides `getRelationshipMessage()`** to return a family-related message
    - Add a field for **`relationship`** (e.g., "Mother", "Brother") (optional enhancement)

Purpose: Demonstrates inheritance and method overriding. Each subclass provides specific implementations of inherited methods.

3. Interface: PaymentSystem

Review the following interface: PaymentSystem.java

Purpose: Defines a contract that all payment system implementations must follow.

4. Payment System Implementations

Create **at least three** concrete classes that implement **`PaymentSystem`**:

- **VenmoPayment**
    - **`getSystemName()`** returns "Venmo"
    - **`getTransactionFee()`** returns 0.00 (free)
    - **`pay()`** simulates instant payment with confirmation message
- **PayPalPayment**
    - **`getSystemName()`** returns "PayPal"
    - **`getTransactionFee()`** 2.5% for amounts over $10
    - **`pay()`** simulates payment with fee calculation
- **AppleCashPayment** (or CashAppPayment)
    - **`getSystemName()`** returns "AppleCash"
    - **`getTransactionFee()`** returns 0.00 (free)
    - **`pay()`** simulates payment with processing message

Purpose: Demonstrates interface implementation and polymorphism - the program can work with any PaymentSystem through the interface reference.

5. Custom Exception: `PaymentFailedException`

Review the the custom exception that extends **`Exception`**:

- Constructor accepts a message string
- Should be thrown by **`pay()`** method when payment fails (e.g., invalid amount)

6. Payment Record Class

Create a **`Payment`** class to represent individual transactions:

- Fields: **`recipientName, amount, paymentSystem, timestamp`** (LocalDateTime)
- Constructor and getters
- **`toString()`** method for formatted output
- Static method to parse payment from file line

7. Update the `Main` Class: MyPaymentSystem

- Contains **`main()`** method
- Manages menu loop
- **Demonstrates polymorphism**: Store contacts in a **`ArrayList<Person>`** treating Friend and Family objects uniformly
- **Demonstrates interface usage**: Use **`PaymentSystem`** interface references to call payment methods
- Handles file I/O for reading/writing contacts and payments
- Manages user input validation

Provided Files

1) **contacts.txt** - Contact information with format

```
ContactType,FirstName LastName,PhoneNumber,PreferredPaymentSystem
```

Example:

```
Friend,John Doe,4157771111,Venmo
Family,Alice Smith,9876154321,PayPal
Family,Jane Smith,41526001111,AppleCash
```

This program requires the use of `File` class to read and write date to the following files:

2) payments.txt

- The first name and last name of the contact paid
- The amount of money paid to the contact (note the 2 digits after the decimal)
- The payment system used
- the timestamp of when the payment was made

```
John Doe,25.00,Venmo,2024-06-20T15:30:00
Alice Smith,100.00,PayPal,2024-06-21T12:00:00
Alice Smith,3.00,PayPal,2024-06-25T11:23:12
```

Submission Details

Print exactly what is requested. Put useful comments into your code. Design and organize your code so that it is easy for others to understand. Please ensure that your main method is in **MyPaymentSystem.java**.

Put your files into the hw3 directory of your myWork directory, and remember to push your code to GitHub before the deadline. This assignment must be turned in before Wednesday, Oct 15th at 9pm.

Expected Output

```
=== MyPaymentSystem ===
1. Pay Contact
2. Print all Payments
3. Update Preferred Payment System
4. Print Contact Information
0. Exit
Choose an option: 1

Enter contact phone number: 9876154321
Contact found: Alice Smith (Family) - Preferred: PayPal
Enter payment amount: 50.00

Confirm payment of $50.00 to Alice Smith using PayPal? (y/n): y
Processing payment through PayPal...
Transaction fee: $1.45
Total charged: $51.45
Payment successful!
```

```
=== MyPaymentSystem ===
[menu repeats]
```

GRADING RUBRIC

This homework will be graded using specifications grading. There are a sequence of tasks to complete. Grades will be assigned as follows:

Grade Level	Tasks to Complete Per Level
C	Compiles + inheritance + basic menu + file reading + core payment functionality
B	All C requirements + interface implementation + polymorphism + 4 menu items + exception handling
A	All B requirements + 3 payment systems + comprehensive validation + 1 enhancement + exceptional code quality

To earn a C, your program must:

- Compile and run without errors
- Implement the **`Person`** base class with all required fields and methods
- Implement **`Friend`** and **`Family`** subclasses that extend **`Person`**
- Read contacts from **`contacts.txt`** successfully
- Implement Menu Option #1 (Pay Contact) - basic functionality working
- Implement Menu Option #2 (Print all Payments) - reads and displays from file

To earn a B, your program must meet all C requirements PLUS:

- Review the **`PaymentSystem`** interface correctly - ensure that all functions are present
- Create at least TWO **`PaymentSystem`** implementations (e.g., VenmoPayment, PayPalPayment)
- Implement Menu Option #3 (Update Preferred Payment System) - fully functional
- Implement Menu Option #4 (Print Contact Information) - displays contact details
- Demonstrate **polymorphism**: Store Friend and Family in an **`ArrayList<Person>`**
- Friend and Family classes **override** at least one method from Person (e.g., **`getRelationshipMessage()`**)

- Write payments to **`payments.txt`** correctly with timestamp
- Uses **`PaymentFailedException`** and throw it appropriately
- Basic input validation (e.g., non-empty names, positive payment amounts)
- Code is organized with reasonable methods (not one giant main method)

To earn an A, your program must meet all B requirements PLUS:

- Create at least THREE **`PaymentSystem`** implementations with different behaviors
- Implement different fee structures for different payment systems (demonstrated in transaction)
- Comprehensive input validation:
  - Phone numbers: exactly 10 digits, numeric only
  - Names: cannot be empty or whitespace only
  - Payment amounts: must be positive
  - Payment system names: must match available systems
- Use **`PaymentSystem`** interface references to call different implementations
- Clear demonstration that **`Friend`** and **`Family`** behave differently (through overridden methods)
- Complete at least ONE enhancement feature (see Enhancement Options below)
- Well-organized, readable code with meaningful comments explaining:
  - How inheritance is used
  - Where polymorphism occurs
  - Why interfaces are beneficial

Enhancement Features (Choose at least ONE sub-option for A Grade)

1. **Payment Reports**:
   - Print all payments to a specific person
   - Print all payments using a specific payment system
   - Calculate and display total amount paid (with transaction fees)
2. **Contact Management**:
   - Add new contacts through the menu
   - Delete existing contacts
   - Sort contacts by name or by total amount paid
3. **Payment System Features**:
   - Implement a 4th payment system with unique fee structure
   - Add payment limits (e.g., Venmo has a $2,999.99 weekly limit)
   - Display transaction fee breakdown before payment
4. **Enhanced Reporting**:
   - Generate a summary report showing total paid per person and per payment system
   - Create a CSV export feature for payments