**1. Design class for "PaymentConfig"**

**Table 1: Attribute design**

| # | Name | Data type | Default value | Description |
|---|---|---|---|---|
| 1 | payUrl | string | null | URL endpoint for payment processing |
| 2 | returnUrl | string | null | URL for redirect after payment completion |
| 3 | version | string | null | API version identifier |
| 4 | secretKey | string | null | Secret key for secure communication |
| 5 | tmnCode | string | null | Payment connection identifier |

**Table 2: Operation design**

| # | Name | Return type | Description (purpose) |
|---|---|---|---|
| 1 | PaymentConfig() | PaymentConfig | Constructor to initialize a new payment configuration object |

**1.1 Method: PaymentConfig()**

- Parameters:
    - None
- Exceptions:
    - None
- How attributes and parameters are to be implemented and used:
    - Initializes a new empty PaymentConfig object with null values for all attributes
- Algorithm to implement operation:
    - Create a new instance of PaymentConfig
    - Set default values for all attributes

**2. Design class for "VNPaySubsystemController"**

**Table 2: Operation design**

| # | Name | Return type | Description (purpose) |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1 | payOrder | PaymentTransaction | Process a payment order and return transaction details |
| 2 | processResponse | void | Process the response from payment gateway |

**2.1 Method: payOrder**

- Parameters:
    - amount: double -> Payment amount to be processed
    - transactionContent: string -> Description of the transaction
    - orderID: string -> Unique identifier for the order
    - paymentConfig: PaymentConfig -> Configuration settings for payment processing
- Exceptions:
    - PaymentException if payment validation fails
- How attributes and parameters are to be implemented and used:
    - Uses provided parameters to initiate payment transaction
- Algorithm to implement operation:
    - Validate order parameters using checkValidOrder
    - Create payment request with provided parameters
    - Initialize payment transaction
    - Return PaymentTransaction object with transaction details

**2.2 Method: processResponse**

- Parameters:
    - vpnReturnUrl: string -> Return URL data from payment gateway
- Exceptions:
    - None explicitly defined
- How attributes and parameters are to be implemented and used:
    - Processes return URL data to determine payment result
- Algorithm to implement operation:

- Parse query string from vpnReturnUrl
- Create Response object
- Handle any error codes
- Create PaymentTransaction with response data
-

**3. Design class for "VNPayScreen"**

**Table 2: Operation design**

| # | Name | Return type | Description (purpose) |
|---|---|---|---|
| 1 | VNPayScreen | void | Constructor for payment screen |
| 2 | handleURLChanged | void | Handle URL changes during payment process |

**3.1 Method: VNPayScreen**

- Parameters: None
- Exceptions: None explicitly defined
- How attributes and parameters are to be implemented and used:
  - Initializes the payment screen interface
- Algorithm to implement operation:
  - Initialize web view component
  - Set up URL handlers
  - Configure screen display settings

**3.2 Method: handleURLChanged**

- Parameters: None
- Exceptions: None
- How attributes and parameters are to be implemented and used:
  - Handles URL change events during payment flow
- Algorithm to implement operation:
  - Parse new URL

- Check if URL contains payment response data
- Process response data if applicable
- Update UI based on payment status

**4. Design class for "PaymentScreen"**

**Table 2: Operation design**

| # | Name | Return type | Description (purpose) |
|---|---|---|---|
| 1 | WebView | void | Initialize web view component |
| 2 | getEngine | void | Get web engine instance |
| 3 | handleURLChanged | void | Handle URL change events |

**4.1 Method: WebView**

- Parameters: None
- Exceptions: None
- How attributes and parameters are to be implemented and used:
  - Sets up web view interface for payment processing
- Algorithm to implement operation:
  - Initialize web view component
  - Configure web view settings
  - Set default loading page

**4.2 Method: getEngine**

- Parameters: None
- Exceptions: None
- How attributes and parameters are to be implemented and used:
  - Retrieves the web engine instance for rendering
- Algorithm to implement operation:
  - Return reference to the web engine object

**4.3 Method: handleURLChanged**

- Parameters: None
- Exceptions: None
- How attributes and parameters are to be implemented and used:
    - Processes URL change events
- Algorithm to implement operation:
    - Detect URL changes
    - Parse URL for payment status
    - Invoke appropriate handlers based on URL content

**5. Design class for "Request"**

**Table 2: Operation design**

| # | Name | Return type | Description (purpose) |
|---|---|---|---|
| 1 | buildQueryURL | string | Build URL query string for payment request |
| 2 | Request | void | Constructor for payment request |

**5.1 Method: buildQueryURL**

- Parameters: None
- Exceptions: None
- How attributes and parameters are to be implemented and used:
    - Constructs URL query string from request parameters
- Algorithm to implement operation:
    - Format all request parameters according to API requirements
    - Encode parameters as URL query string
    - Return formatted query string

**5.2 Method: Request**

- Parameters:
    - amount: double -> Payment amount
    - orderID: string -> Order identifier

- transactionContent: string -> Transaction description
- paymentConfig: PaymentConfig -> Payment configuration

- Exceptions: None
- How attributes and parameters are to be implemented and used:
  - Creates a new payment request with specified parameters
- Algorithm to implement operation:
  - Initialize request object
  - Set all parameters for the request
  - Validate request parameters

**6. Design class for "Response"**

**Table 2: Operation design**

| # | Name | Return type | Description (purpose) |
|---|---|---|---|
| 1 | parseQueryString | void | Parse query string from response URL |
| 2 | Response | void | Constructor for response object |
| 3 | handleErrorCode | void | Handle error codes in response |

**6.1 Method: parseQueryString**

- Parameters:
  - queryString: string -> Query string to parse
- Exceptions:
  - None explicitly defined
- How attributes and parameters are to be implemented and used:
  - Extracts parameters from query string
- Algorithm to implement operation:
  - Split query string by delimiters
  - Parse key-value pairs
  - Store parsed parameters in response object

### 6.2 Method: Response

- Parameters:
    - vnpReturnURL : string -> Response URL
- Exceptions: None
- How attributes and parameters are to be implemented and used:
    - Initializes a new response object
- Algorithm to implement operation:
    - Create response object
    - Initialize response properties

### 6.3 Method: handleErrorCode

- Parameters: None
- Exceptions: None
- How attributes and parameters are to be implemented and used:
    - Processes error codes in payment response
- Algorithm to implement operation:
    - Identify error code in response
    - Map error code to appropriate exception or message
    - Process error handling logic

## 7. Design class for "PaymentTransaction"

**Table 1: Attribute design**

| # | Name | Data type | Default value | Description |
|---|---|---|---|---|
| 1 | transactionID | int | null | Unique identifier for transaction |
| 2 | amount | double | 0.0 | Transaction amount |
| 3 | orderID | string | null | Order identifier |
| 4 | bankCode | string | null | Code of processing bank |

| | | | | |
|---|---|---|---|---|
| 5 | bankTransactionID | string | null | Bank's transaction identifier |
| 6 | cardType | string | null | Type of card used |
| 7 | payDate | string | null | Date and time of payment |
| 8 | transactionContent | string | null | Description of transaction |

**Table 2: Operation design**

| # | **Name** | **Return type** | **Description (purpose)** |
|---|---|---|---|
| 1 | savePaymentTransaction | void | Save transaction details to database |
| 2 | PaymentTransaction | void | Constructor for transaction object |

**7.1 Method: savePaymentTransaction**

- Parameters: None
- Exceptions: None
- How attributes and parameters are to be implemented and used:
    - Stores transaction data in persistent storage
- Algorithm to implement operation:
    - Format transaction data for storage
    - Execute database operation to save transaction
    - Handle any storage errors

**7.2 Method: PaymentTransaction**

- Parameters: None
- Exceptions: None
- How attributes and parameters are to be implemented and used:
    - Creates a new transaction object
- Algorithm to implement operation:
    - Initialize transaction object
    - Set default values for all attributes