CREDIT CARD PROCESSING SYSTEM

AIM:

To design credit card processing system.

SOFTWARE SYSTEM:

The project entails the development of a Java-based "Credit Card Processing System" software, designed to facilitate secure and efficient online transaction processing

This system aims to offer reliability, affordability, security, and scalability, catering to the needs of both customers and business merchant banks

It enables seamless acceptance of credit card payments over the internet, ensuring a smooth and trustworthy transaction experience.

SOFTWARE REQUIREMENT SPECIFICATION:

- When a consumer initiates a purchase, the merchant's commerce application prompts for credit card details, often alongside other pertinent information like shipping addresses.
- The consumer securely inputs payment information either through a form protected by Secure Socket Layer (SSL) protocol or via compliant applications such as internet browsers adhering to the Secure Electronic Transaction specification. SSL ensures encryption of payment data within the form.
- Utilizing payment software integrated into the web server, the merchant transmits the encrypted transaction to the acquiring processor for authorization.
- The acquiring processor evaluates the transaction and either approves it for a specific amount, which reduces the available credit limit without immediate charge, or declines it.
- Upon authorization, the next step involves "capturing" the transaction, which confirms and solidifies the transaction's processing.
- In cases of cancellations or returns, a "void" transaction is generated, canceling any associated credit or charges.
- The final step involves "settling" the transaction, facilitating the exchange of funds between the merchant and acquiring processor.

USE CASE MODEL:

Actors:

Customer: The individual who initiates a credit card transaction.

Banking System: Represents the bank's system which is responsible for processing the payment.

Use Cases:

Initiate Transaction:

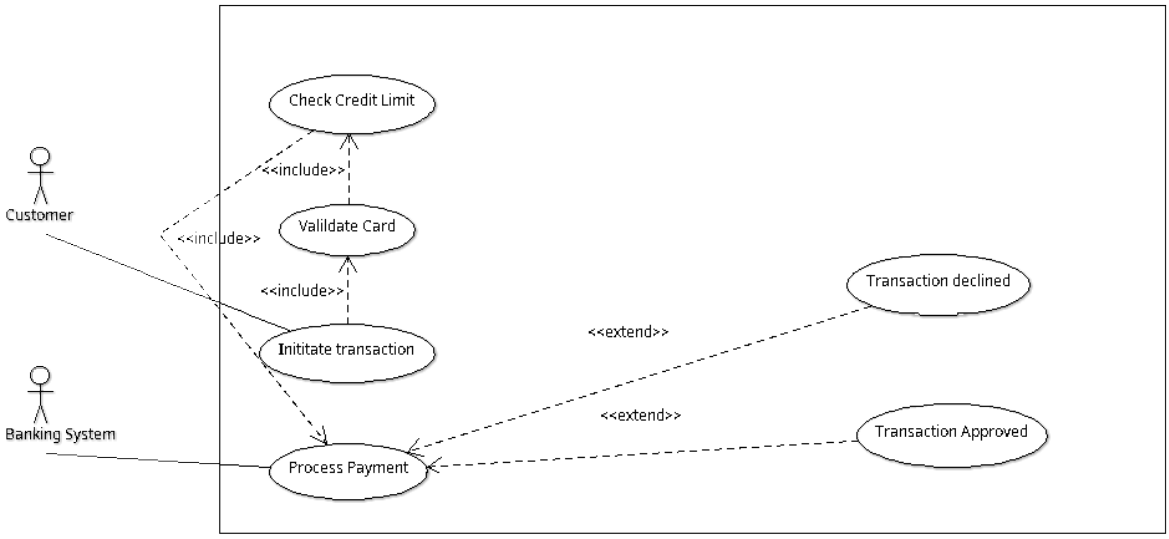
Validate Card: Once the transaction is initiated, the system will include a step to validate the credit card details.

Check Credit Limit: After validating the card, the system includes a check to ensure that the credit limit has not been exceeded.

Process Payment: This is the core function where the actual processing of the payment takes place.

Transaction Approved: An extension of the "Process Payment" use case, it signifies a successful payment transaction.

Transaction Declined: This is another extension of the "Process Payment" that denotes a failed payment transaction due to various reasons such as exceeding the credit limit, invalid card details, etc.



UML CLASS DIAGRAM:

Customer: Represents the customer who owns the credit card and initiates transactions.

- Attributes include the customer's name and credit card information.
- Behavior includes initiating a transaction.

Credit Card: Encapsulates the credit card details.

- Attributes include the card number, expiration date, and security code.
- Behavior includes validation of the card and checking the credit limit for a given transaction amount.

Transaction: Represents a payment transaction with credit card.

- Attributes include the transaction amount, date, and status.

Banking System: Represents the external banking system that processes payments.

- It has a behavior to process a payment, updating the transaction's status accordingly.

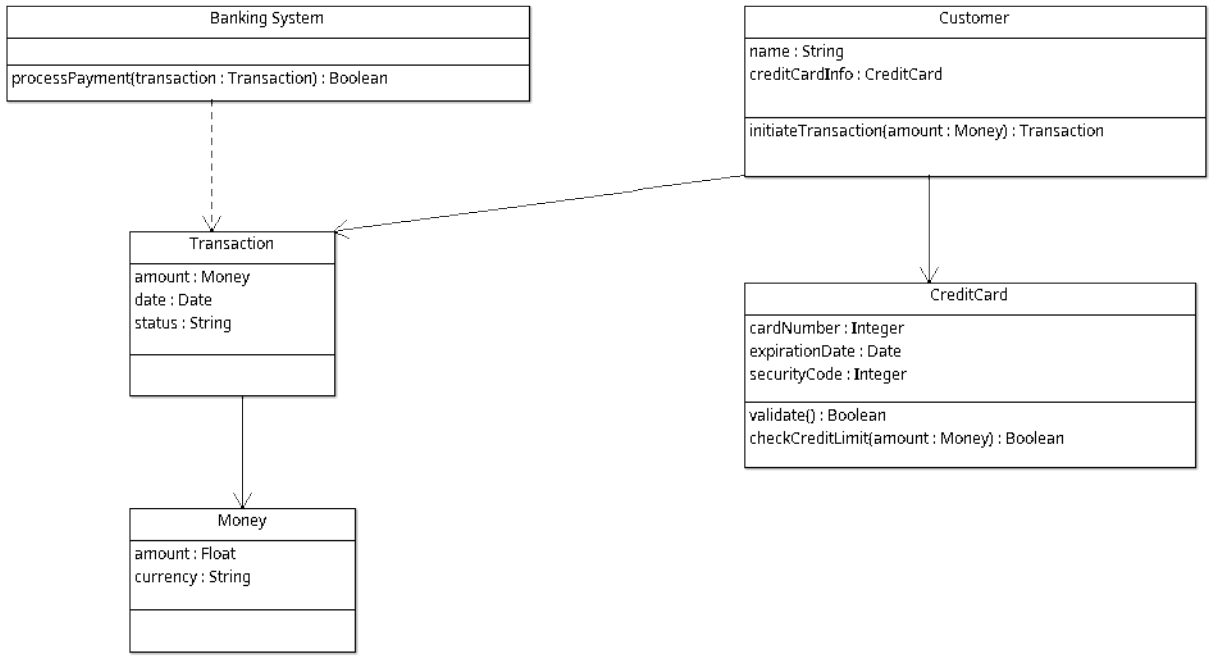
Money: Represents a monetary value associated with a transaction.

- Attributes include the amount and currency.

A Customer is associated with a CreditCard and can have multiple Transaction instances.

A Transaction is associated with Money, indicating the amount of money involved in the transaction.

The BankingSystem can process Transaction instances, changing their status based on whether the payment is approved or declined.

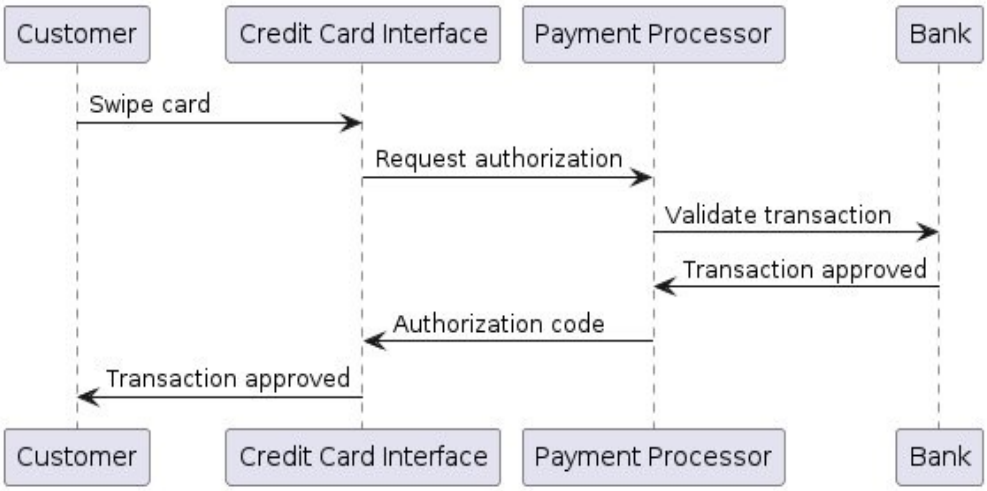


UML SEQUENCE DIAGRAM:

Actors and Participants:

- **Customer:** The person who wishes to make a payment using a credit card.
- **Credit Card:** The credit card entity that will be validated and checked for credit limit.
- **Banking System:** The system that processes the payment.
- **Transaction:** Represents a payment transaction attempt.

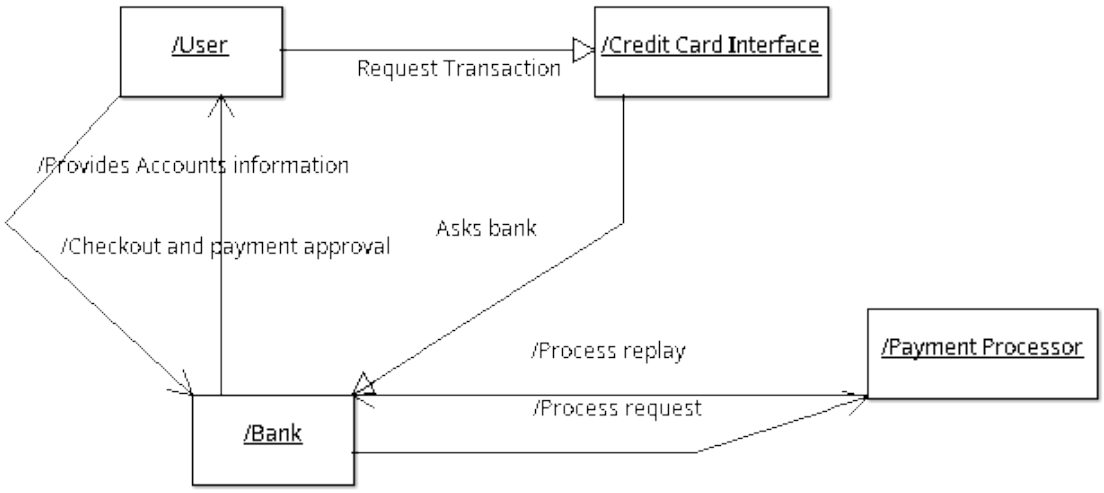
1. The Customer initiates the payment process by entering credit card details which are sent to the Credit Card (CC) entity to validate.
2. The Credit Card entity performs the Validate() operation. If validation is successful, it sends a positive response back to the Customer.
3. On successful validation, the Customer makes a request to the Banking System (BS) to process the payment.
4. The Banking System then creates a new Transaction with the transaction details.
5. Next, the Banking System checks with the Credit Card entity to verify if the credit limit is sufficient.
6. The Credit Card returns the credit limit status to the Banking System.



UML COLLABORATION DIAGRAM:

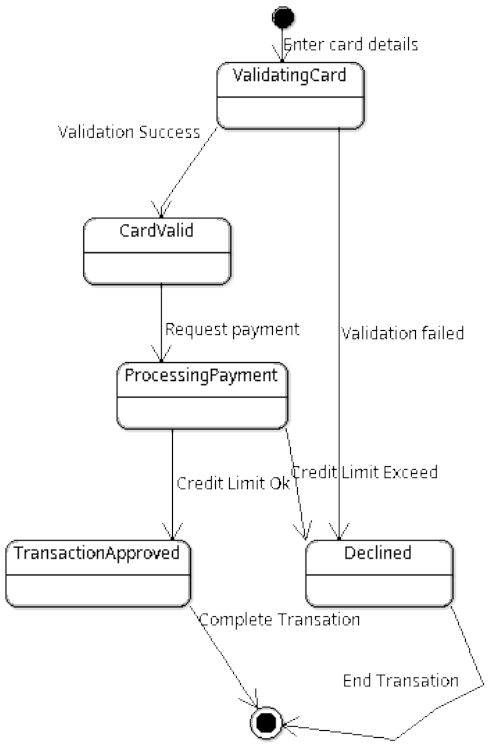
- **Customer:** The person who intends to make a payment using a credit card.
- **Credit Card Interface:** The system or service where the customer swipes their credit card to initiate the transaction.
- **Payment Processor:** The service that processes credit card transactions by communicating with the bank to authorize payments.
- **Bank:** The financial institution that issues the credit card and authorizes the transaction.

1. The Customer uses the Credit Card Interface to swipe their card, indicated by the "Swipe card" message.
2. The Credit Card Interface then requests authorization from the Payment Processor through a "Request authorization" message.
3. Next, the Payment Processor sends a "Validate transaction" message to the Bank to check the validity and approve the transaction.
4. The Bank processes the request and, if the transaction is valid, sends a "Transaction approved" message back to the Payment Processor.
5. The Payment Processor then generates an authorization code and sends this "Authorization code" message to the Credit Card Interface.
6. Finally, the Credit Card Interface informs the Customer that the transaction is approved.

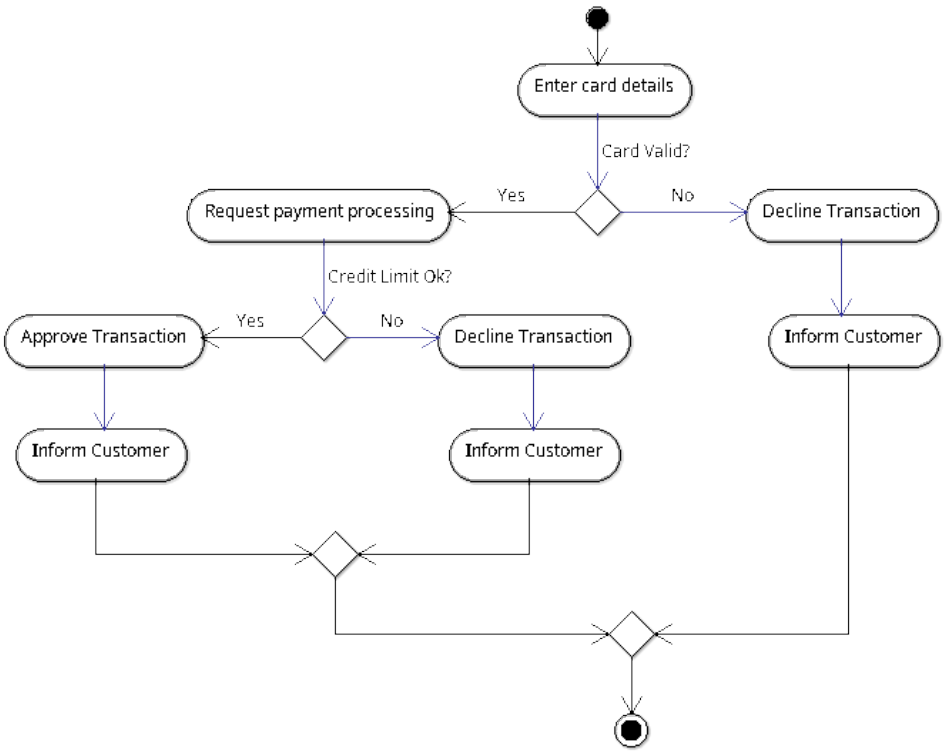


STATE CHART DIAGRAM:

1. **States:** These are represented as rounded rectangles and reflect the various conditions of the system. For example, in a credit card processing system, you might have states such as "Idle", "Card Swiped", "Authorizing", "Approved", "Declined", etc.
2. **Transitions:** Depicted by arrows, transitions show the movement from one state to another and are typically triggered by an event. For instance, a "Card Swiped" event might result in a transition from "Idle" to "Authorizing".
3. **Events:** These are the triggers that cause the transitions between states. An "Authorization Response" could be an event that triggers a transition from "Authorizing" to either "Approved" or "Declined".
4. **Actions:** Actions may occur as a result of entering or exiting a state. For example, when entering the "Approved" state, the action might be "Issue Receipt".
5. **Initial State:** Indicated by a filled black circle, this marks the starting point of the system's behavior.
6. **Final State:** Represented by a circle surrounding a smaller filled circle, indicating where the behavior terminates.



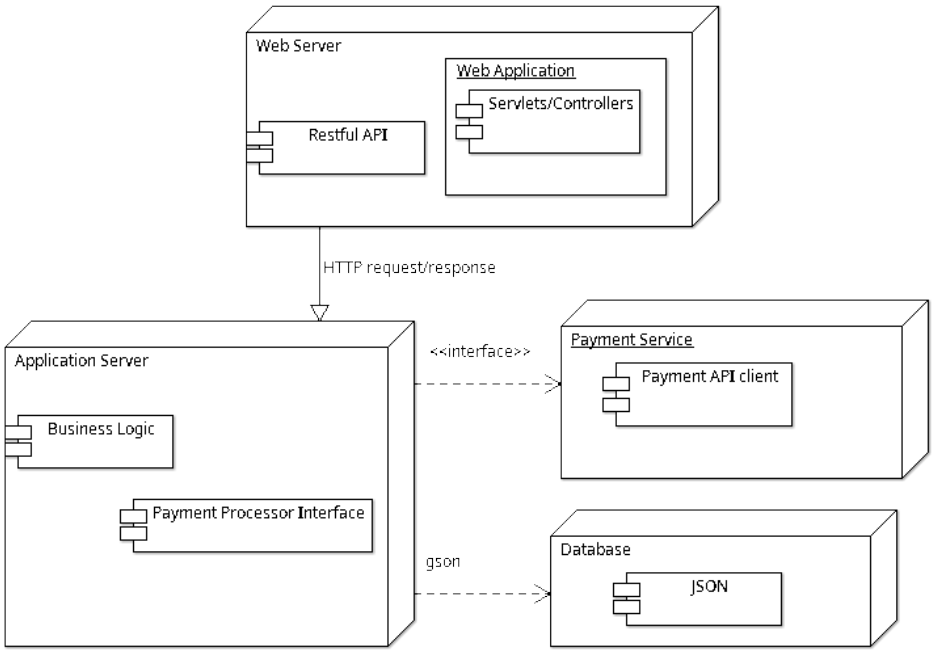
UML ACTIVITY DIAGRAM:



Explanation:

- **Start (Circle symbol):** Represents the starting point of the process.
- **Action state "Enter card details":** Shows the customer entering their credit card information into the system.
- **Decision node "Card Valid?":** A conditional check to determine if the entered credit card details are valid.
  - If the card is **valid**, the process moves to "Request payment processing."
  - If the card is **invalid**, it leads to an action state "Decline transaction" and then to "Inform Customer," where the customer is notified of the declined transaction. The process ends here for invalid cards.
- **Action state "Request payment processing":** Indicates the system processing the payment request.
- **Decision node "Credit Limit OK?":** A conditional check for the credit limit on the card. A conditional check for the credit limit on the card.
  - If the credit limit is sufficient, the activity moves to "Approve transaction" followed by "Inform Customer," where the system informs the customer of the approved transaction.
  - If the credit limit is insufficient, it goes to "Decline transaction" and then "Inform Customer," notifying the customer of the declined transaction.
- **End (Filled circle with border):** Represents the end of the transaction process.

UML DEPLOYMENT DIAGRAM FOR PROJECT:



- Web Server contains the Web Application with Servlets/Controllers that handle HTTP requests and responses, as well as a RESTful API artifact for the API end-points.
- Application Server includes the Business Logic, which can be Java classes that implement the system's business rules, and a Payment Processor Interface that defines how the application communicates with external payment services.
- Database Server hosts the Transaction Database with components like JPA/Hibernate Entities representing the tables and relationships in the database.
- External Payment Gateway shows an external node, potentially provided by a third-party service, with a Payment Service that includes a component for a Payment API Client (typically a Java class that interacts with the external payment gateway's API).

RESULT:

Thus credit card processing system UML diagrams were designed and code was generated successfully.