PseudoCode for Cart Class Methods

```
// Pseudo-code for Cart.changePaymentMethod()
FUNCTION changePaymentMethod(payementMethod: newPaymentMethod)
  // This method would allow a user to change their payment method for the
cart
  IF newPaymentMethod IS VALID
    Cart.PaymentMethod = newPaymentMethod
    RETURN "Payment method updated."
  ELSE
    RETURN "Invalid payment method."
  ENDIF
END FUNCTION



// Pseudo-code for Cart.changeDeliveryOption()
FUNCTION changeDeliveryOption(deliveryOption: newDeliveryOption)
  // This method would let a user change their delivery option for the
items in the cart
  Cart.DeliveryOption = newDeliveryOption
  RETURN "Delivery option updated."
END FUNCTION

// Pseudo-code for Order.buyNow()
FUNCTION buyNow()
  // This method would be responsible for processing the purchase of items
in the cart
  IF Cart.SaleItems IS EMPTY
    RETURN "Your cart is empty."
  ELSE
    // Calculate total cost from SaleItems in Cart
    totalCost = SUM(Cart.SaleItems.price * Cart.SaleItems.quantity)
    // Process payment
    IF PaymentMethod.verifyPayment(totalCost) IS SUCCESSFUL
      // Set delivery option
      DELIVERY_METHOD = Cart.DeliveryOption
      IF DELIVERY_METHOD IS pickup
        Cart.pickUpOrder()
      ELSE
        DELIVERY_METHOD.acceptRequest()
      // Create a new order
      Order = NEW Order(Cart.SaleItems, totalCost, DELIVERY_METHOD)
      // Add the new order to the OrderPage
      OrderPage.Orders.ADD(Order)
      RETURN "Purchase successful. Order placed."
    ELSE
      RETURN "Payment failed. Please try again."
```

```
    ENDIF
  ENDIF
END FUNCTION
```

PseudoCode for deliveryOption Class

```
//Linked to changeDeliveryOption method from Cart
// Pseudo-code for DeliveryOption.acceptRequest()
FUNCTION acceptRequest()
  // This method would handle the acceptance of a delivery request by
assigning a driver
  availableDriver = FIND an available driver in the system
  IF availableDriver IS NOT NULL
    ASSIGN availableDriver to DeliveryOption
    RETURN "Delivery accepted. Driver assigned."
  ELSE
    RETURN "No available drivers. Please try again later."
  ENDIF
END FUNCTION

// Pseudo-code for DeliveryOption.declineRequest()
FUNCTION declineRequest()
  // This method would handle the rejection of a delivery request, perhaps
if no driver is available or for other reasons
  IF DeliveryOption.driver IS NOT NULL
    // Assuming there's a way to de-assign the driver
    DEASSIGN DeliveryOption.driver
    DeliveryOption.driver = NULL
  ENDIF

  // Assuming there's a field in the DeliveryOption that stores the status
of the delivery
  DeliveryOption.status = "Declined"

  // Log the decline action for record-keeping and auditing purposes
  LOG "Delivery request declined for DeliveryOption ID: " +
DeliveryOption.id

  // Here we might also want to notify the relevant parties, such as the
restaurant or the customer, that the delivery has been declined
  NOTIFY parties "Delivery has been declined for order associated with
DeliveryOption ID: " + DeliveryOption.id

  // The method could return a status indicating the delivery has been
declined successfully
  RETURN "Delivery option declined successfully."
END FUNCTION
```

- **DEASSIGN** is a placeholder for the actual logic to remove driver from delivery
- **NOTIFY** is an abstracted action that would involve sending notification to inform about the change in delivery status

PseudoCode for Reviews Class

```
// Pseudo-code for Reviews.likeReview()
FUNCTION likeReview(review: reviewId, reviewer: userId)
  // This method would increment the like count for a review
  review = GET review by reviewId
  IF review IS NULL
    RETURN "Review not found."
  ENDIF

  user = GET user by userId
  IF user HAS ALREADY liked review
    RETURN "You have already liked this review."
  ELSE
    review.likes = review.likes + 1
    LOG like action with userId and reviewId
    SAVE review changes
    RETURN "Review liked successfully."
  ENDIF
END FUNCTION

// Pseudo-code for Reviews.dislikeReview()
FUNCTION dislikeReview(reviewId, reviewer: userId)
  // This method would increment the dislike count for a review
  review = GET review by reviewId
  IF review IS NULL
    RETURN "Review not found."
  ENDIF

  user = GET user by userId
  IF user HAS ALREADY disliked review
    RETURN "You have already disliked this review."
  ELSE
    review.dislikes = review.dislikes + 1
    LOG dislike action with userId and reviewId
    SAVE review changes
    RETURN "Review disliked successfully."
  ENDIF
END FUNCTION
```

```
// Pseudo-code for Reviews.reportReview()
FUNCTION reportReview(reviewId, reviewer: userId)
  // This method would flag a review for inspection by
moderators/administrators
  review = GET review by reviewId
  IF review IS NULL
    RETURN "Review not found."
  ENDIF

  user = GET user by userId
  IF user HAS ALREADY reported review
    RETURN "You have already reported this review."
  ELSE
    review.flaggedForReview = TRUE
    LOG report action with userId and reviewId
    SAVE review changes
    NOTIFY moderators to inspect the flagged review
    RETURN "Review reported successfully."
  ENDIF
END FUNCTION
```

- **GET review by reviewID and GET user by userID are placeholders for operations that would retrieve review and user objects from a data store**

PseudoCode for financialInstitution class

```
// Pseudo-code for FinancialInstitution.verifyPayment()
FUNCTION verifyPayment(paymentMethod: paymentForm)
  // This method checks if the payment method is valid and can be charged
  IF paymentMethod IS VALID AND paymentMethod.hasSufficientFunds()
    RETURN TRUE  // The payment method is valid and has sufficient funds
  ELSE
    RETURN FALSE // The payment method is invalid or does not have
sufficient funds
  ENDIF
END FUNCTION

// Pseudo-code for FinancialInstitution.chargeUser()
FUNCTION chargeUser(paymentMethod: paymentForm, Integer: amount)
  // This method charges the user's account with the specified amount
  IF verifyPayment(paymentMethod) IS FALSE
    RETURN "Payment verification failed. Transaction not completed."
  ELSE
    // Assuming the paymentMethod object has a method to charge the amount
    transactionStatus = paymentMethod.chargeAmount(amount)
    IF transactionStatus IS SUCCESSFUL
      LOG "Account charged successfully for amount: " + amount
```

```
      RETURN "Transaction completed successfully."
    ELSE
      LOG "Failed to charge account for amount: " + amount
      RETURN "Transaction failed."
    ENDIF
  ENDIF
END FUNCTION
```

- **LOG** is an abstracted action that record the transaction's outcome in a system's logs
- Both methods abstract away the specific details of interfacing with an institution's payment processing API

PseudoCode for saleItem class

```
// Pseudo-code for SaleItem.changeQuantity()
FUNCTION changeQuantity(newQuantity)
  // This method updates the quantity of the SaleItem
  IF newQuantity IS LESS THAN 1
    RETURN "Quantity must be at least 1."
  ELSE
    this.quantity = newQuantity
    RETURN "Quantity updated to " + newQuantity + "."
  ENDIF
END FUNCTION

// Pseudo-code for SaleItem.addToCart(cart)
FUNCTION addToCart(cart: cartObject)
  // This method adds the SaleItem to a Cart
  IF cartObject CONTAINS this
    RETURN "Item already in cart."
  ELSE
    cartObject.SaleItems.ADD(this)
    RETURN "Item added to cart."
  ENDIF
END FUNCTION
```

- cartObject.SaleItems is assumed to be a list of 'SaleItem' objects within the Cart class

PseudoCode for Receipt Class

```
// Pseudo-code for Receipt.download()
FUNCTION download(account: userId)
  // This method generates a downloadable file for the receipt
  IF VERIFY user access rights with userId FOR this receipt
    receiptFile = GENERATE receipt file with details from the Receipt
object
```

```
    IF receiptFile IS SUCCESSFULLY created
      PROVIDE download link for receiptFile
      RETURN "Receipt download initiated."
    ELSE
      LOG "Receipt generation failed for user: " + userId
      RETURN "Unable to generate receipt for download."
    ENDIF
  ELSE
    RETURN "User does not have the rights to download this receipt."
  ENDIF
END FUNCTION
```

- **VERIFY** user access rights with userId FOR this receipt is an abstract check to ensure that the user requesting the download is authorized to access this receipt.
- **GENERATE** receipt file with details from the Receipt object is a placeholder for the operation that would format the receipt details into a downloadable file. The actual implementation would likely involve creating a PDF or another file format.

- **'PROVIDE download link for receiptFile'** represents the action of giving the user a way to download the generated receipt file, perhaps by sending a link to their email or starting a file download in the web browser.
- **LOG** is an action that would record the event in the system's logs, particularly in the case of a failure to generate the receipt.