BIT 4524 Project

University Retail Food Service Vendor Accounts Payable System

Kabir Singh

Supreet Pannu

Ryan Whalen

Sheharyar Gondal

10th December, 2019

Table of Contents

Executive Summary

Our consulting team has been developing a system that will replace the current system to increase the efficiency of processing invoices and generating reports (to aid in the management of the system). The current system, which involves many time-consuming and monotonous tasks, is outdated and is directly causing the University to lose money from late invoice payments and excessive hiring costs. Since the high turnover rate and many of the current business problems stem from using outdated information technology, the new system will utilize modern information technology to address and eliminate problems with duplication of payments, multiple entry of data, the inability to rapidly assess the status of accounts payable, and the inability to generate reports on the number of invoices by vendor, account type, or time period.

The system is cross platform and uses modern computer equipment connected to a secure server that has access to a local area network and internet, thus enabling full network capability between the University Center, Accounts Payable office, and the Check Issuance office at a minimal cost. Synergizing the Accounts Payable office and the Check Issuance office will minimize the time required for forms, reports, and data to be shared between offices. Not only will the implementation of modern information technology increase data communication speeds between offices, it will also greatly reduce the repetitive and monotonous tasks that were previously causing accounts payable clerks to quit their job at the accounts payable office. Clerks will be able to scan information from the invoice and save it into the system which will enable clerks to upload the saved information into the Payment Report and Request Issuance form. Furthermore, the clerks will be able to easily generate monthly reports in a fraction of the time it currently takes to generate a report. This will greatly decrease the turnover rate of employees by reducing both the monotony of daily routine processing and also the stress associated with requests for summary information about the status of accounts payable and the creation of summary reports which are common complaints of employees leaving these positions.

Requirements Definition

Nonfunctional requirements

1. Operational requirements
    a. The system will be cross platform.
    b. The system will save an electronic copy of each payment report and request issuance for future reference.
    c. Two copies of the payment report are printed. One copy is kept in the accounts payable files, and the other copy is sent to the university check issuance office.
    d. Will combine the functions of excel (for the payment report) and the stand-alone program that is used to enter information for the request issuance.
    e. The system will include drop down boxes or some caching mechanism to store frequently entered items when filling out the payment report and request issuance form.
    f. System is connected to the internet and local area network
2. Performance requirements
    a. The system should be able to store/save payment reports and request issuances in 2 seconds
    b. The system should be able to retrieve payment reports and request issuances in 2 seconds for matching with processed request issuances and for future reference
    c. Process payments within 10 days
3. Security requirements
    a. Only designated employees should have access to using the system and network.
    b. Only clerks within the check issuance can receive documents from the accounts payable office and vice versa.
    c. Login required to join and use network
4. Cultural and political requirements
    a. No special cultural or political requirements are anticipated

Functional requirements

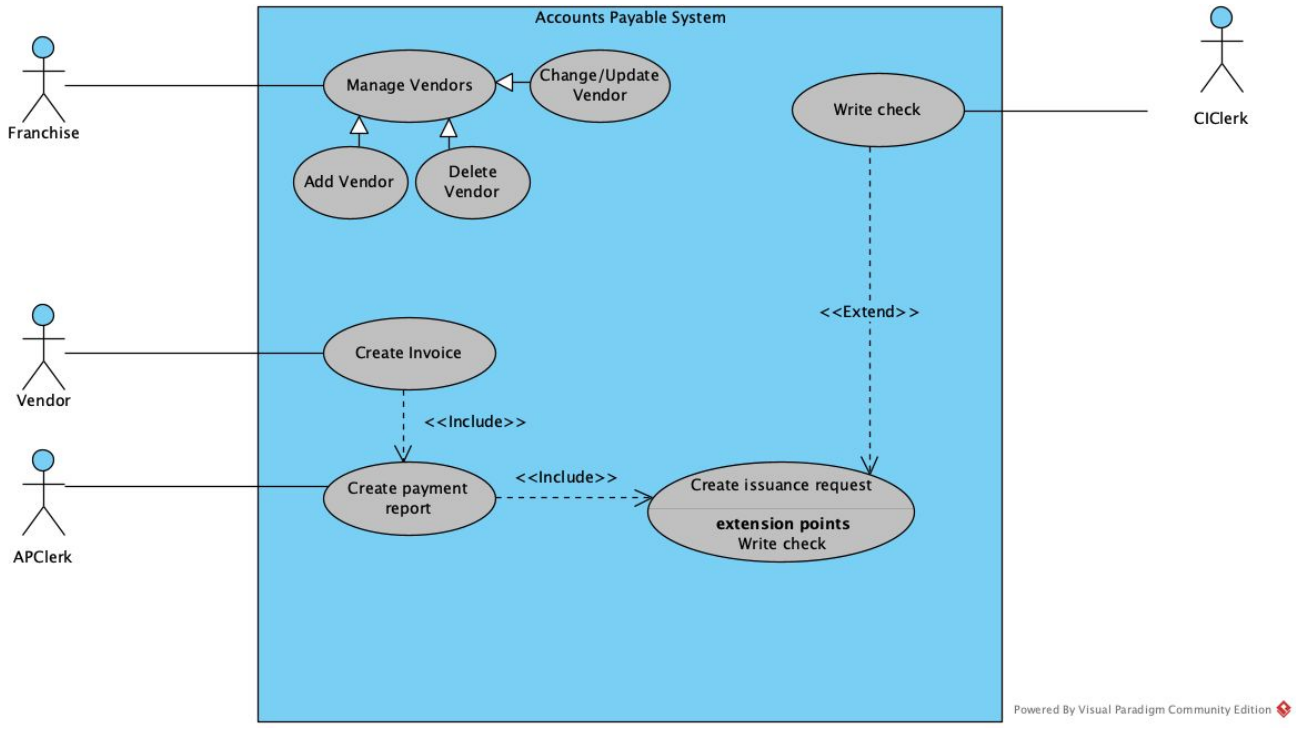
1. Manage vendors
    a. Add vendors to franchise's preapproved vendor list
    b. Update information for a vendor on franchise's preapproved vendor list
    c. Delete vendors from franchise's preapproved vendor list
2. Retail vendors accounts office or AP office
    a. Combine excel and issuance request software
    b. Vendor info will simultaneously update templates with recurring information as excel payment report is filled
    c. Create payment report
        i. Two printed one in A/P files and other sent to check issuance office
        ii. Store online
    d. Create issuance request

    e. Send to check issuance office for payment
    f. Tracking request issuances
        i. The accounts payable office sends a request issuance to the check issuance office
        ii. The request will be tracked to ensure it is returned and paid prior to 10 days
    g. Track invoices per vendor and any franchise its associated with; send to check issuance to gen one check
    h. Gen monthly data Report: total paid out by each franchise, franchise total broken down by university account, university detail and summary by unv acct, vendor totals, total for each acct
3. University Check Issuance office
    a. Generate Check and send it to vendor
    b. Process request issuance
        i. Send to AP with check num, amt, and date it was paid
    c. Match electronic payment report and electronic version of original invoice to processed request issuance form
        i. Store docs

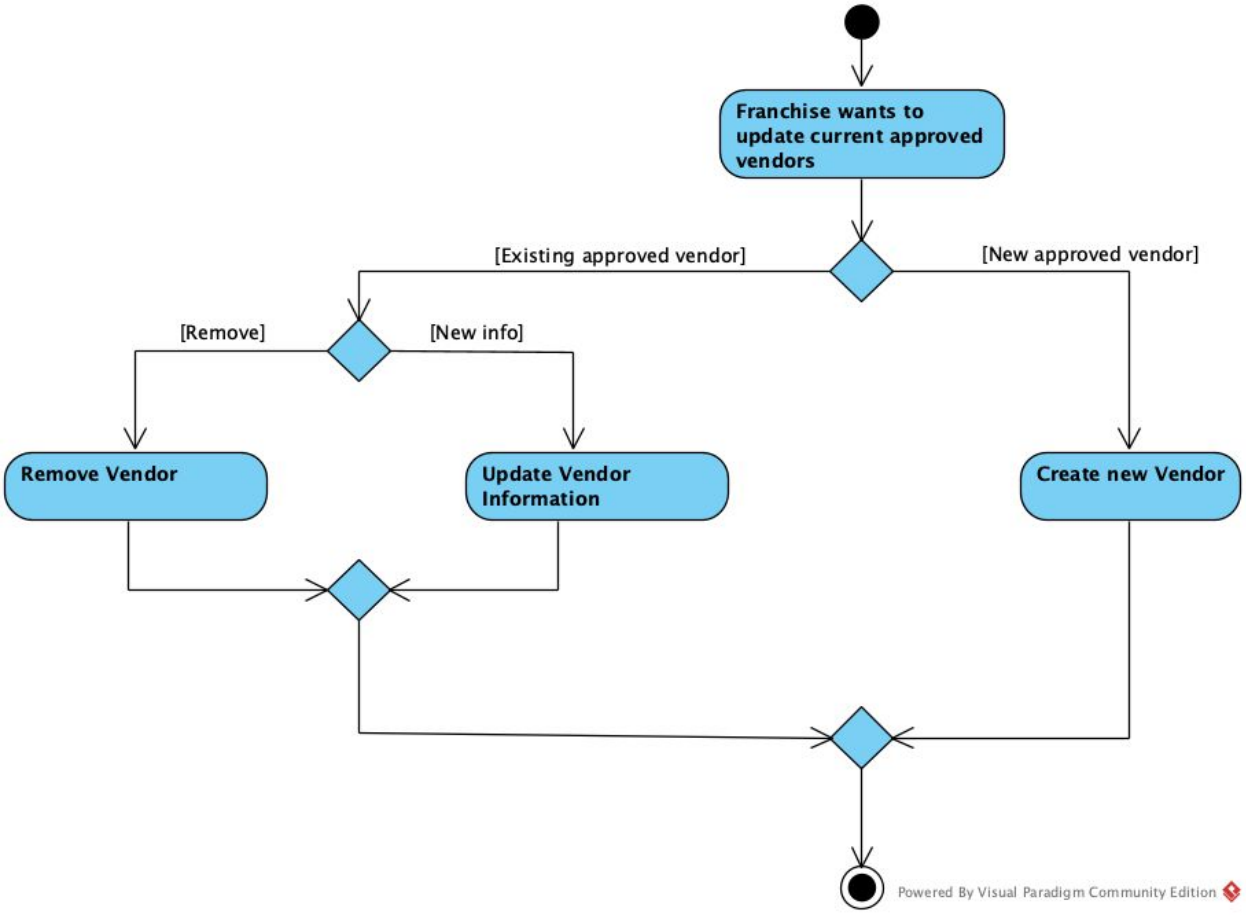
User Stories

1. As a vendor I want to be able to easily deliver invoices to the university center and receive the full payment within 30 days.
2. As a clerk I do not like repeatedly entering the same info regarding vendors, I wish the necessary info could be saved
3. As a clerk I do not like having to search for a specific document in a crowded file cabinet
4. As the head of the department I only want those with proper credentials to have access to invoice related files
5. As a clerk I want to be able to quickly access reports
6. As the head of financial operations I want to pay our invoices within 10 days of receiving them to qualify for a 2% discount
7. As head of financial operations I want to pay our invoices within 30 days of receiving them to avoid a 2% late fee
8. As head of financial operations I want to reduce employee costs and maintain high retention rate
9. As a clerk I do not like using two different softwares, I want to only use one and store info across reports

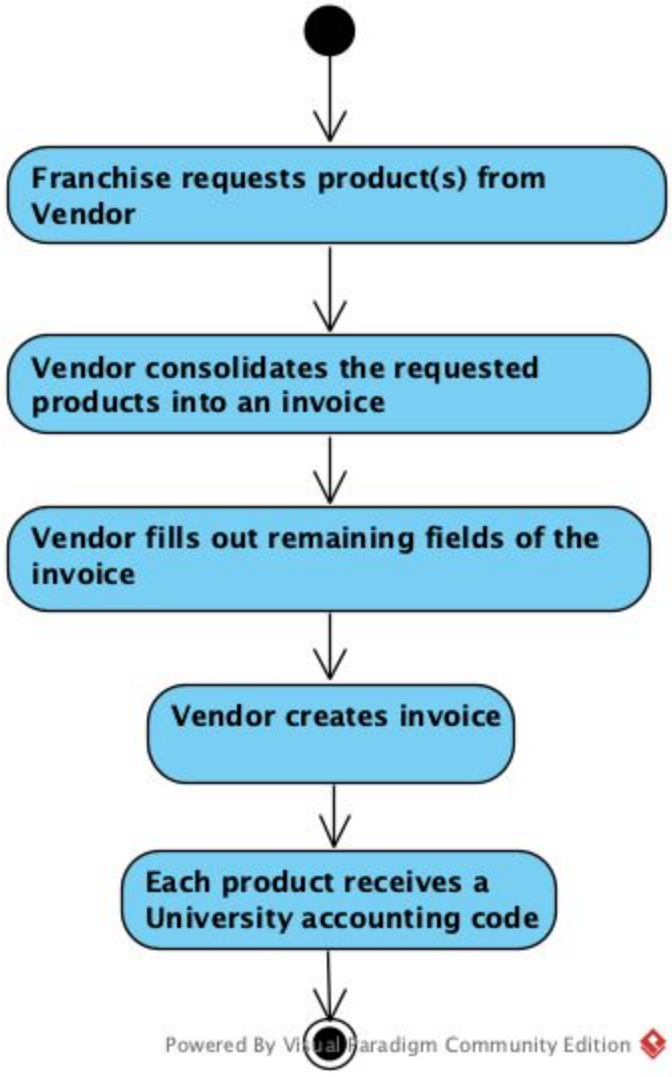
Use Case Diagram



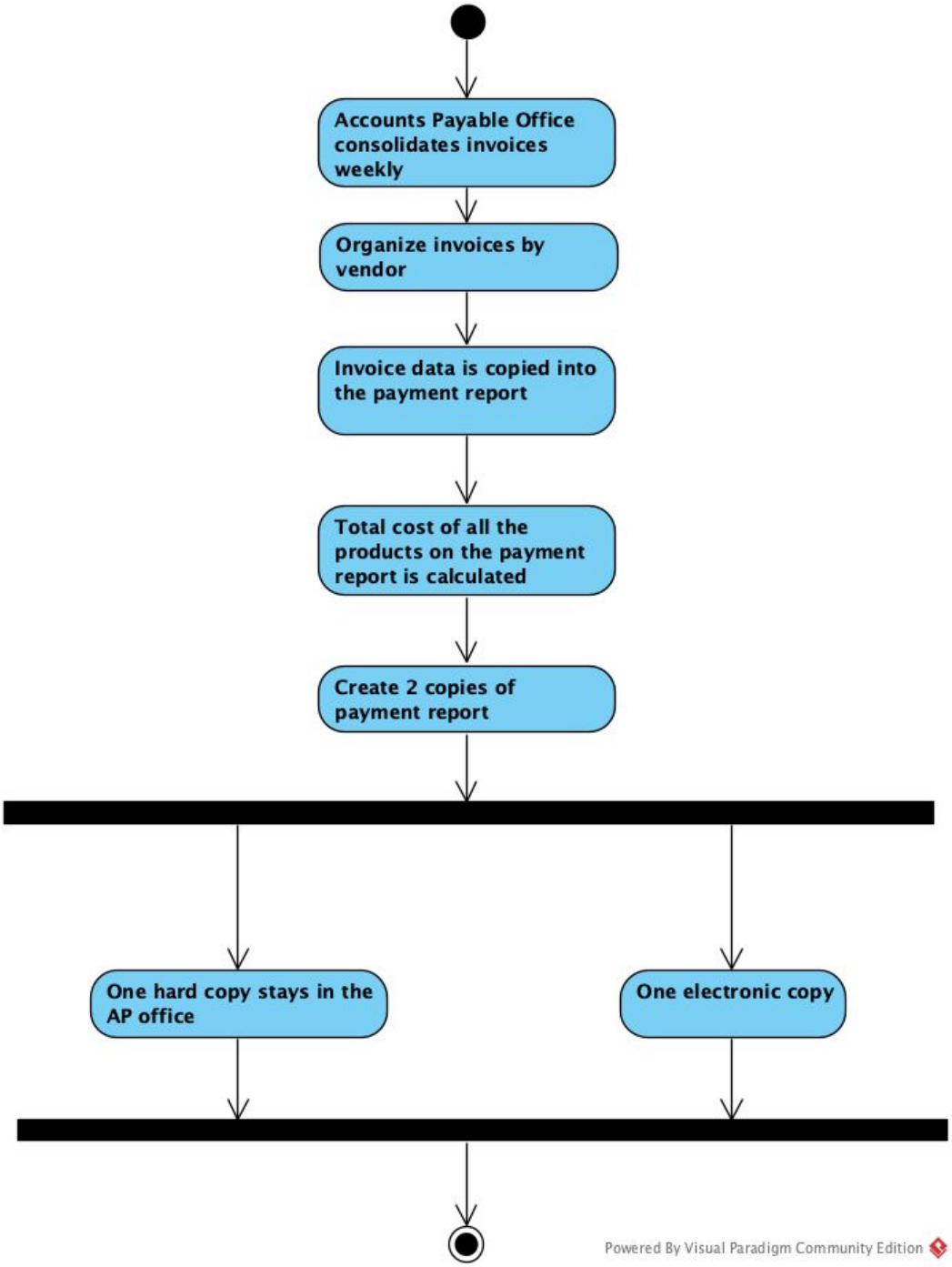
Manage Vendors Activity Diagram



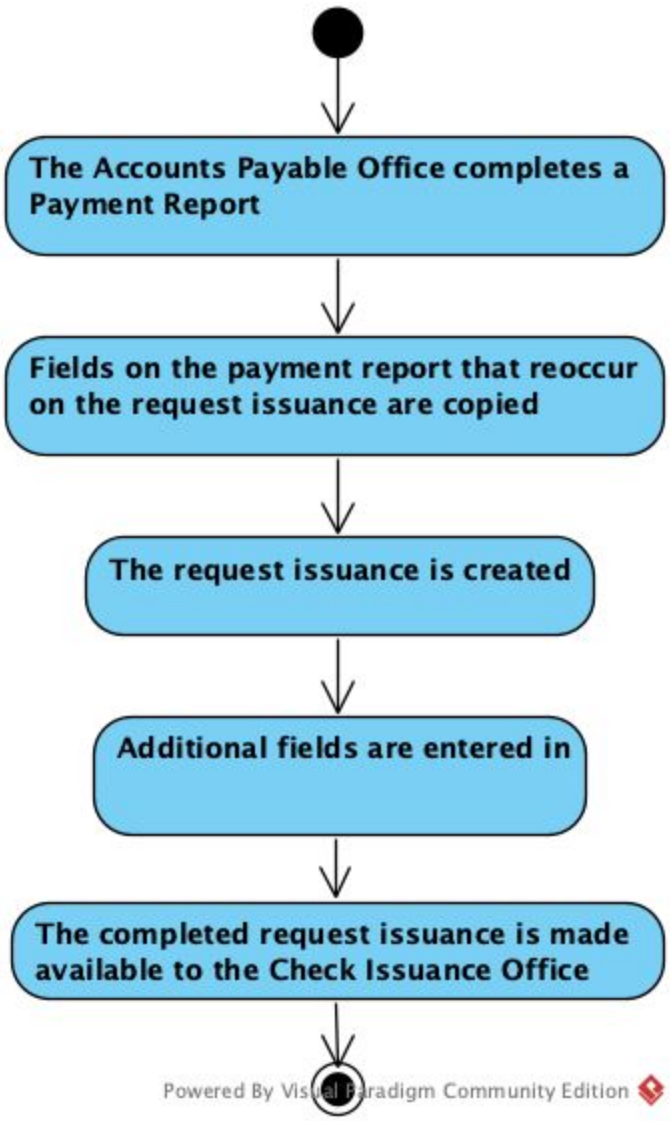
Create Invoice Activity Diagram



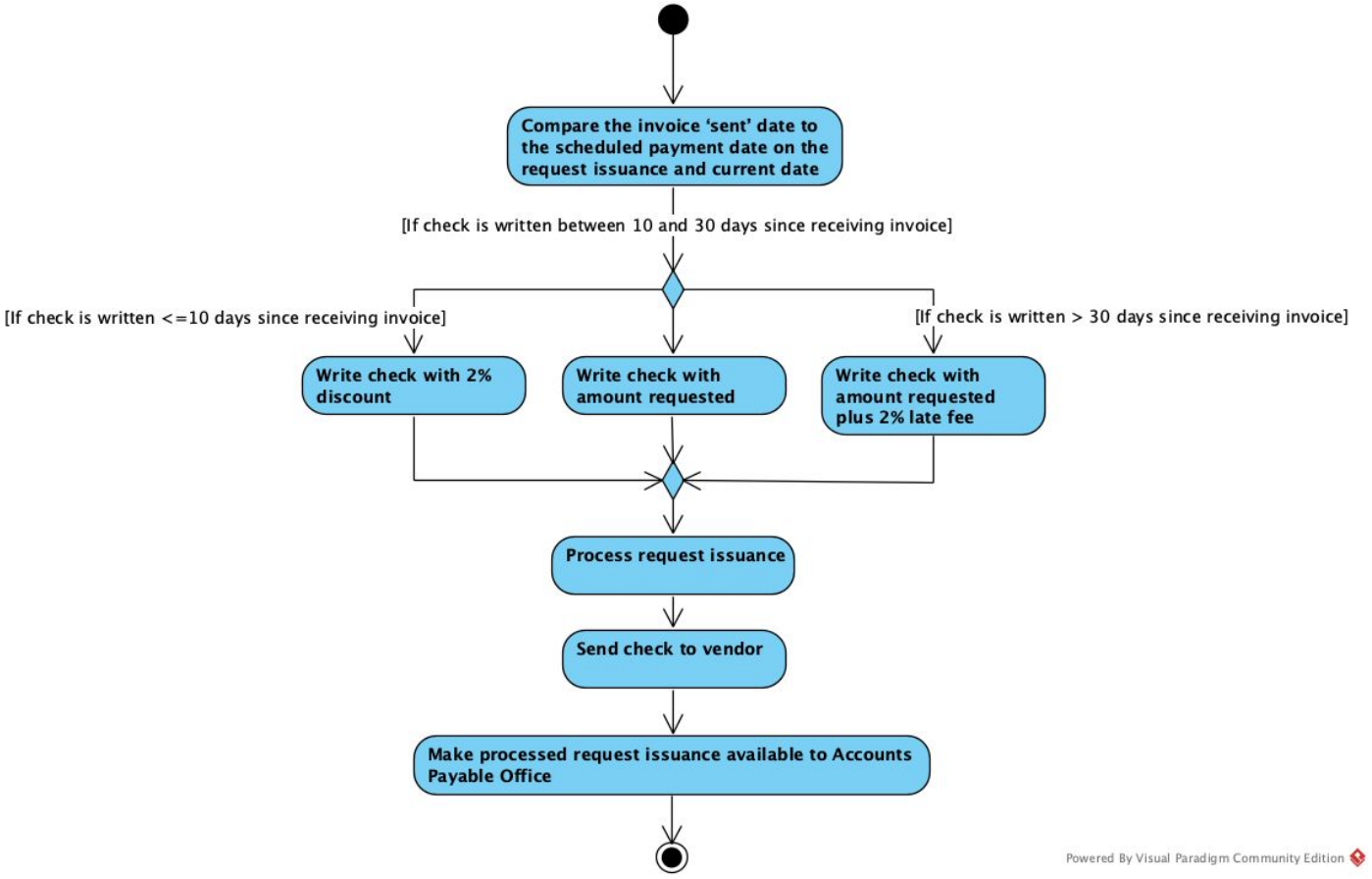
Create Payment Report Activity Diagram



Create Request Issuance Activity Diagram



Write Check Activity Diagram



Use Case Descriptions

Use Case Name: Create Invoice	ID: 1	Importance Level: High
Primary Actor: Vendor	Use Case Type: Detail, Essential	
Stakeholders and Interests: - Franchise: Wants to receive supplies - Vendor: The Vendor needs to be paid for providing the university with supplies		
Brief Description: Vendors fulfill requests and creates invoice		
Trigger: Franchise submits request for products to vendor Type: External		
Relationships: Association: Vendor Include: Create payment Report Extend: None Generalization: None		
Normal Flow of Events: 1. Franchise submits request for products 2. Vendor fulfills the request 3. Vendor generates invoice based on products requested 4. The products on an invoice then receive accounting codes 5. Call Create Payment Report		
SubFlows: None		
Alternate/Exceptional Flows: None		

Use Case Name: Create Payment Report	ID: 2	Importance Level: High
Primary Actor: APClerk	Use Case Type: Detail, Essential	
Stakeholders and Interests: - Accounts Payable Office: They need to make the payment report available to the Check Issuance Office so it can write a check to pay for the supplies - Check issuance Office: The Check Issuance Office needs the details in the request issuance so that they can write a check to the vendor for supplies ordered - Vendors: The Vendor needs to be paid for providing the university with supplies		
Brief Description: Vendors fulfill requests and generate invoice		
Trigger: Franchise submits request for products to vendor Type: External		
Relationships: Association: AP Office Clerk Include: Create Request Issuance Extend: None Generalization: Invoice		
Normal Flow of Events: 1. Consolidate invoices weekly 2. Organize invoices by vendor 4. Invoice data is copied into payment report 5. Total Cost is calculated for invoices 6. Create one electronic copy and one hard copy of payment report 7. Call Create Request Issuance		
SubFlows: None		
Alternate/Exceptional Flows: None		

Use Case Name: Create Request Issuance	ID: 3	Importance Level: High
Primary Actor: APClerk	Use Case Type: Detail, Essential	
Stakeholders and Interests: - Accounts Payable Office: They need to make the request issuance available to the Check Issuance Office so that they can write a check to pay for the supplies - Check Issuance Office: the Check Issuance Office needs the details in the request issuance so that they can write a check to the vendor for supplies ordered - Vendor: The Vendor needs to be paid for providing the university with supplies		
Brief Description: The Accounts Payable Office creates and sends a formal request issuance that details the specifics of a payment to the check issuance office.		
Trigger: The Accounts Payable Office creates a payment report Type: External		
Relationships: Association: AP Office Include: Create Payment Report Extend: None Generalization: None		
Normal Flow of Events: 1. The Accounts Payable Office completes a Payment Report 2. Fields on the payment report that reoccur on the request issuance are copied 3. Additional fields are filled in 4. The completed request issuance is made available to the Check Issuance Office for processing		
SubFlows: None		
Alternate/Exceptional Flows: None		

Use Case Name: Write Check	ID: 4	Importance Level: High
Primary Actor: Check Office Employee	Use Case Type: Detail, Essential	

Stakeholders and Interests:
- Accounts Payable Office: They need the check in order to fulfill the completion of the invoice(s)
- Vendor: The vendor needs to be paid for providing supplies.

Brief Description: This use case describes how the check office generates a check to pay an invoice(s) after receiving the necessary documentation from the Accounts Payable Office.

Trigger: AP office sends a payment report along with an accompanying request issuance
Type: External

Relationships:
Association: Check Office Staff
Include: None
Extend: None
Generalization: None

Normal Flow of Events:
1. The Check Office receives a payment report along with an accompanying request issuance from the AP office
2. The Check office creates a check
3. The check office compares the scheduled payment date with the current date
    a. If the current date is 10 days or less after the invoice has been sent, the S-1: write check with discount subflow is performed
    b. If the current date is between 10 and 30 days after the invoice has been sent, the S-2: write check subflow is performed
    c. If the current date is more than 30 days after the invoice has been sent, the S-3: write check with late fee subflow is performed
4. The request issuance is processed by filling in the check number, the amount on the check, and the date it was paid
5. The processed request issuance is sent back to the Accounts Payable Office so that it can be matched with the corresponding payment report and invoices
6. The check is sent to the Vendor

SubFlows:
S-1: Write check with discount
- The Check Issuance Office writes the check to the Vendor with a 2% discount on the original amount for paying early

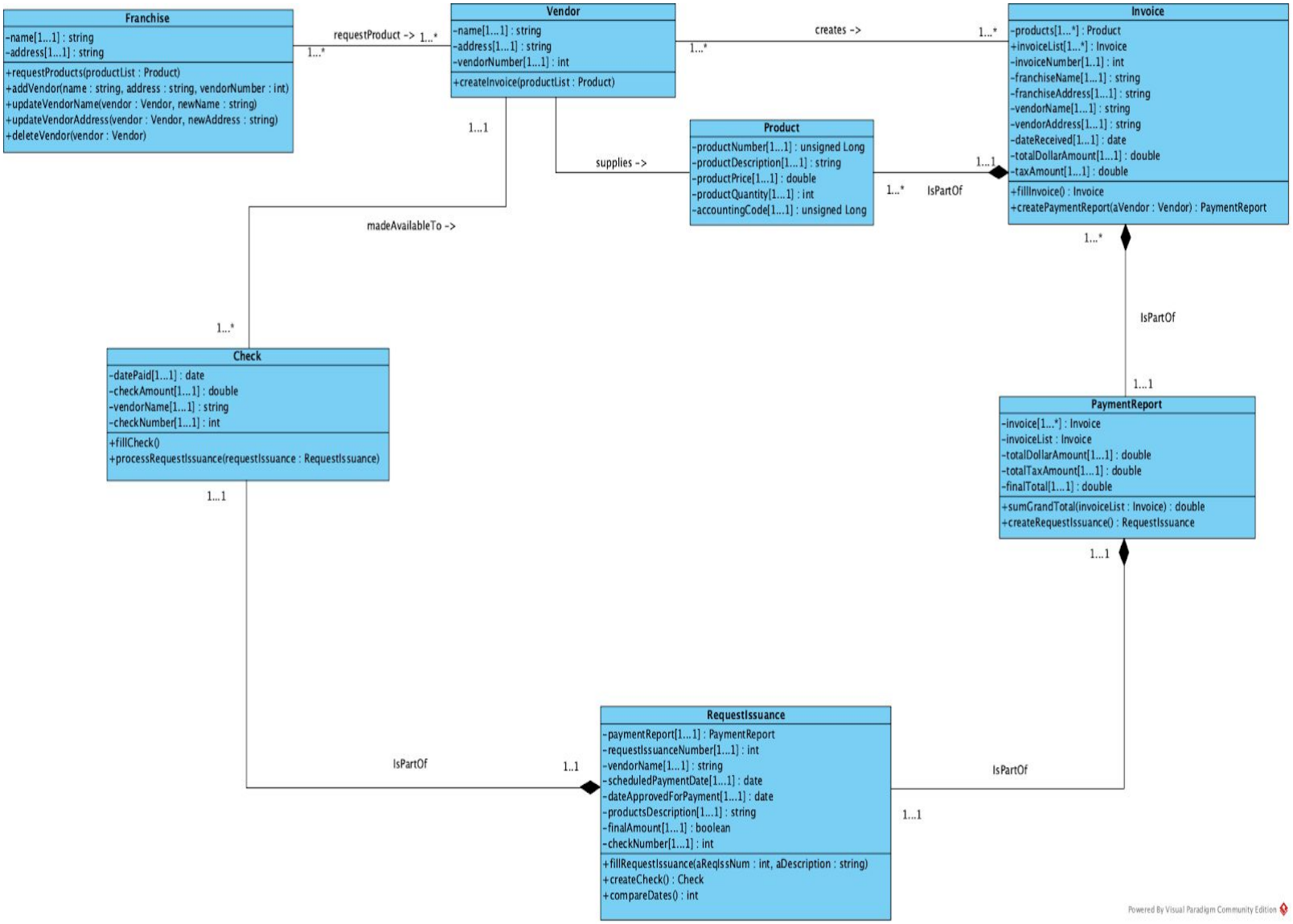
S-2: Write check
- The Check Issuance Office writes the check to the Vendor with the amount that is requested

S-3: Write check with late fee
- The Check Issuance Office writes the check to the Vendor with a 2% late fee added to the original amount for paying late

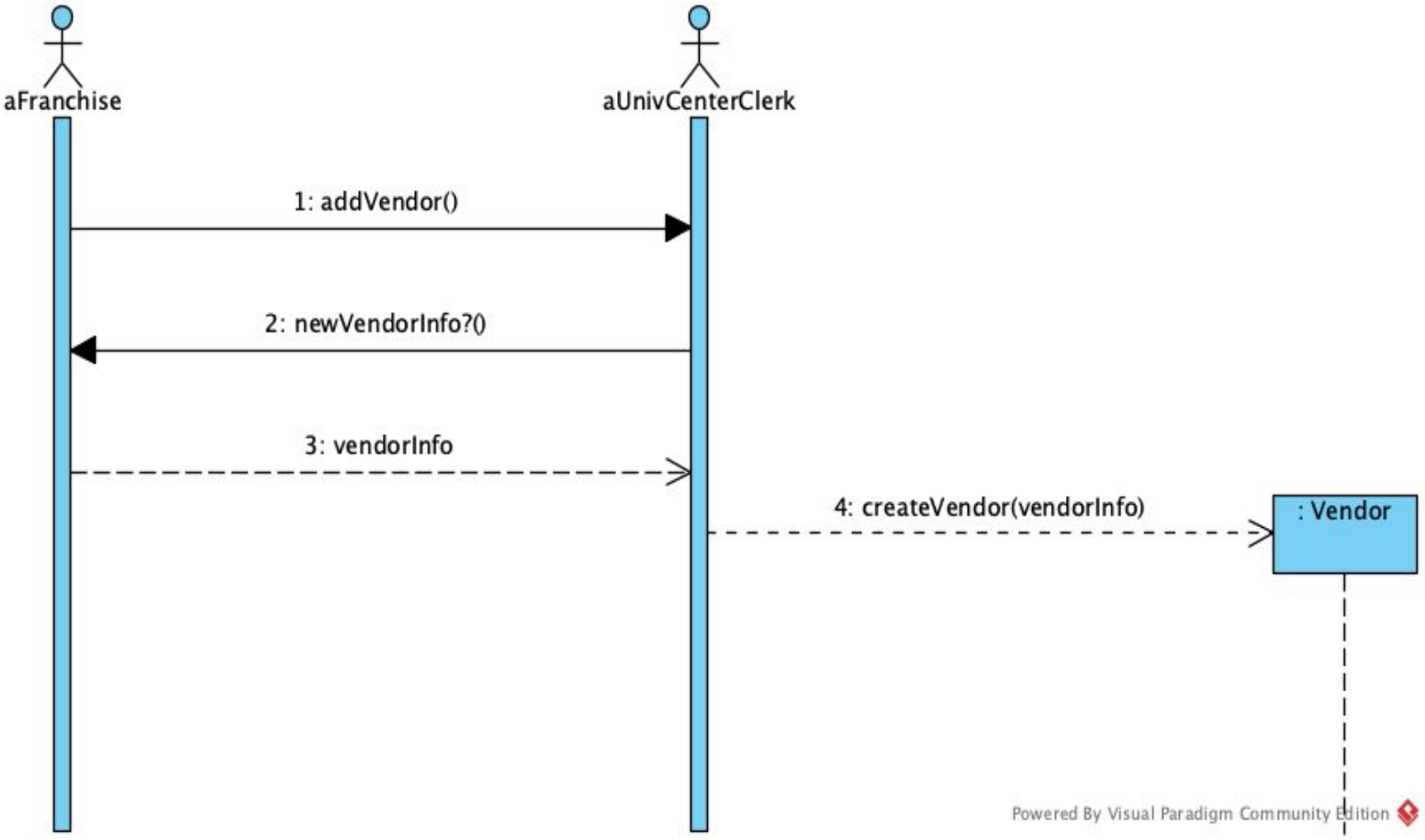
Alternate/Exceptional Flows: None

Use Case Name: Manage Vendors	ID: 5	Importance Level: Low
Primary Actor: Franchise	Use Case Type: Detail, Essential	
Stakeholders and Interests: - Vendor: The current vendor(s) information is updated - Franchise: Does not want to use a specific vendor and wants the university to drop them.		
Brief Description: Franchise updates current approved Vendor(s)		
Trigger: New Approved Vendor, New Unapproved Vendor, Vendor Info Changed Type: External		
Relationships: Association: Franchise Actor Include: None Extend: None Generalization: New, Delete, Update		
Normal Flow of Events: 1. Franchise wants to update current approved vendors a. If the franchise wishes to add a new approved vendor, the S-1: add vendor subflow is performed b. If the franchise wishes to update an existing approved vendor, the S-2: update vendor subflow is performed c. If the franchise wishes to remove an existing approved vendor, the S-3: remove vendor subflow is performed		
SubFlows: S-1: Add Vendor - The franchise adds a new approved vendor to the current approved vendors S-2: Remove Vendor - The franchise removes an already approved vendor S-3: Update Vendor - The franchise updates the attributes of an existing approved vendor		
Alternate/Exceptional Flows: None		

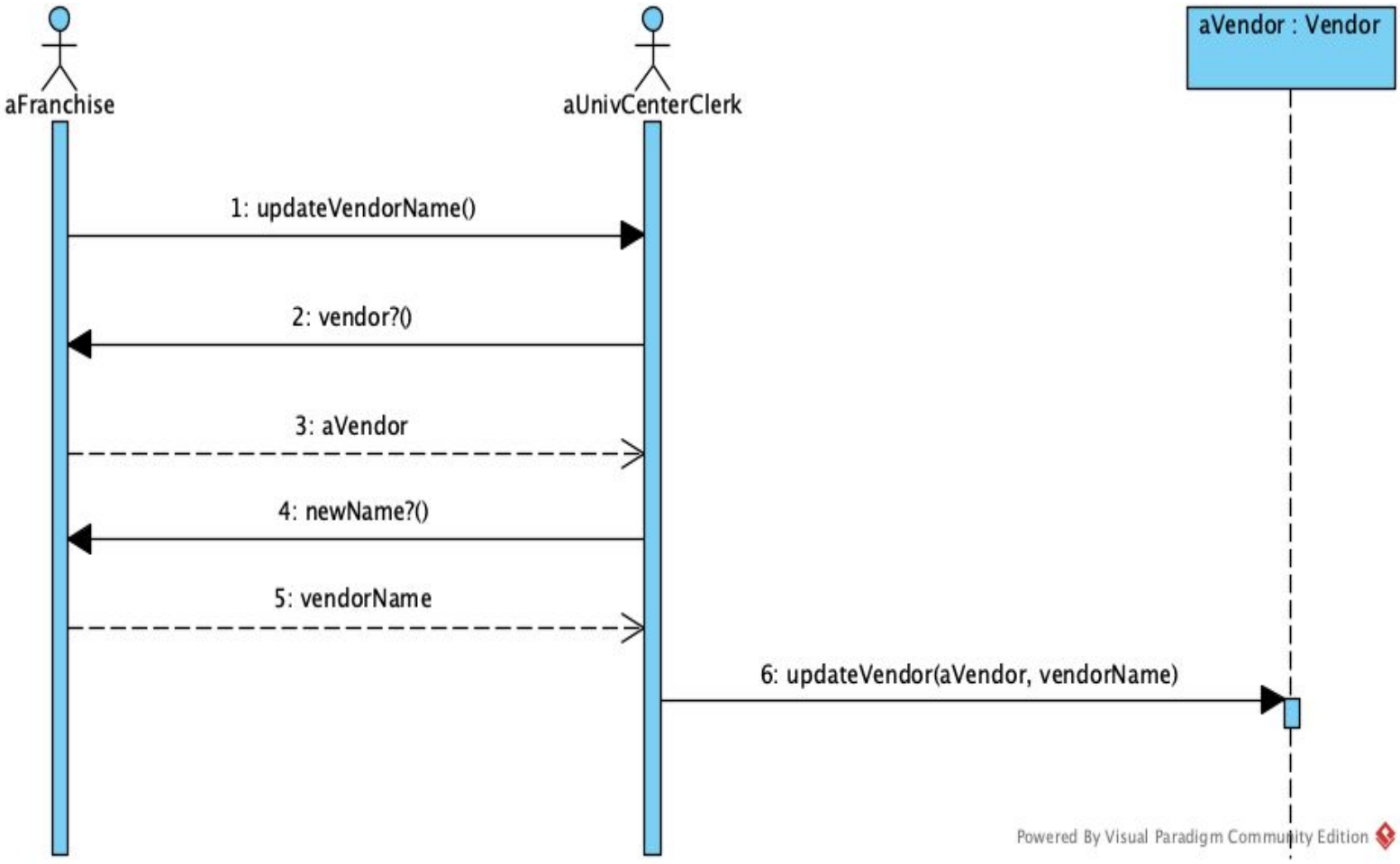
Class Diagram



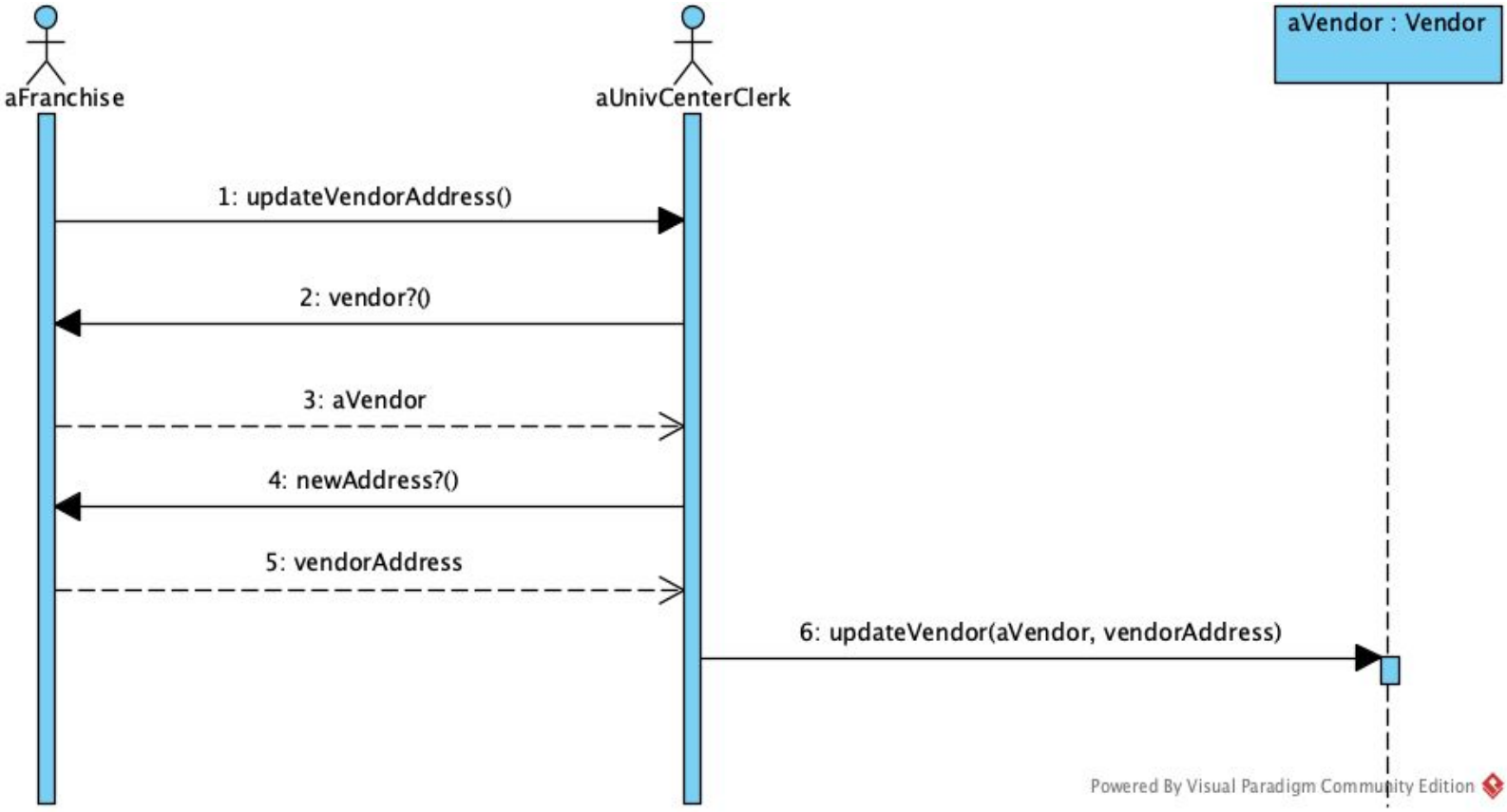
Add Vendor Sequence Diagram



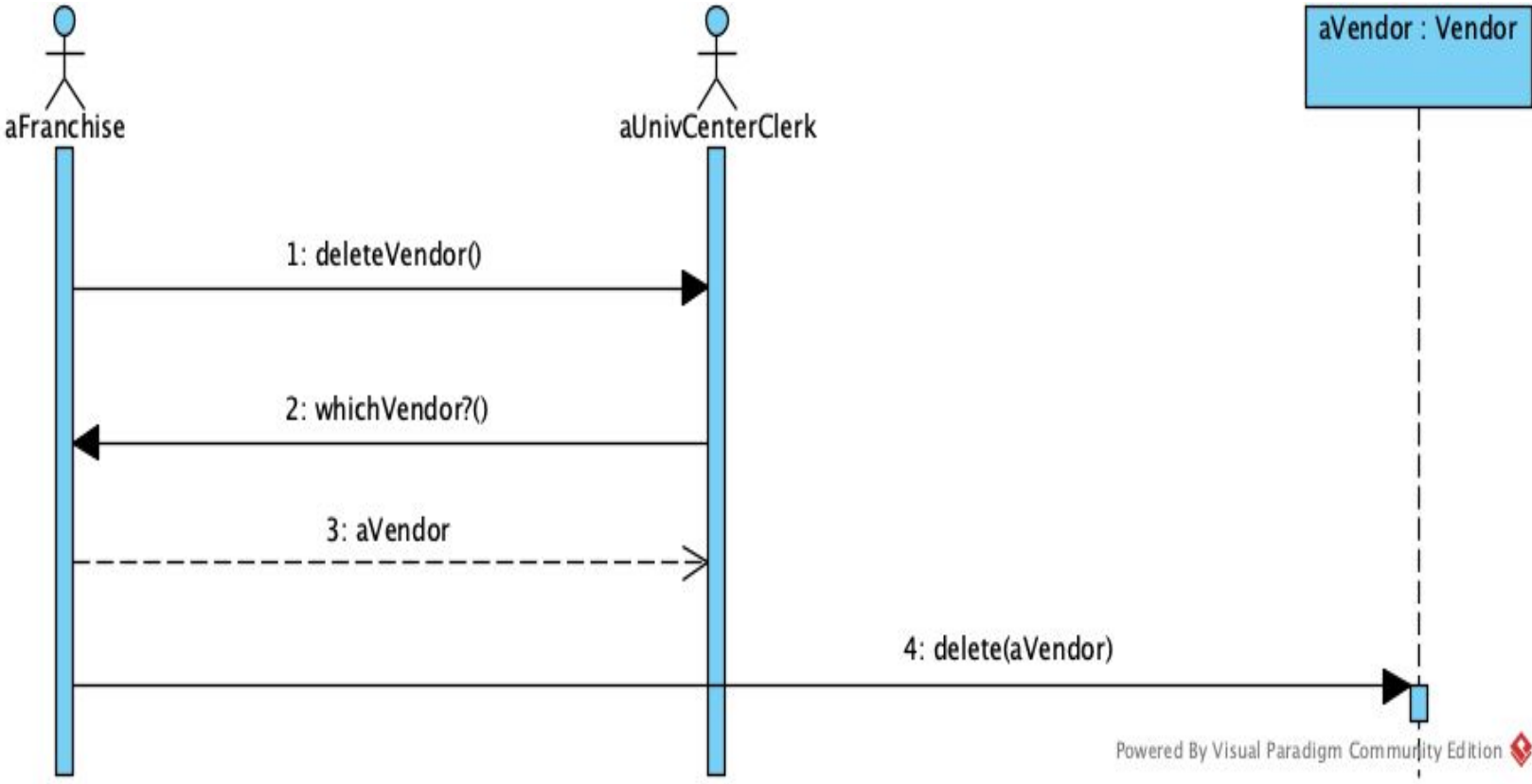
Update Vendor Name Sequence Diagram



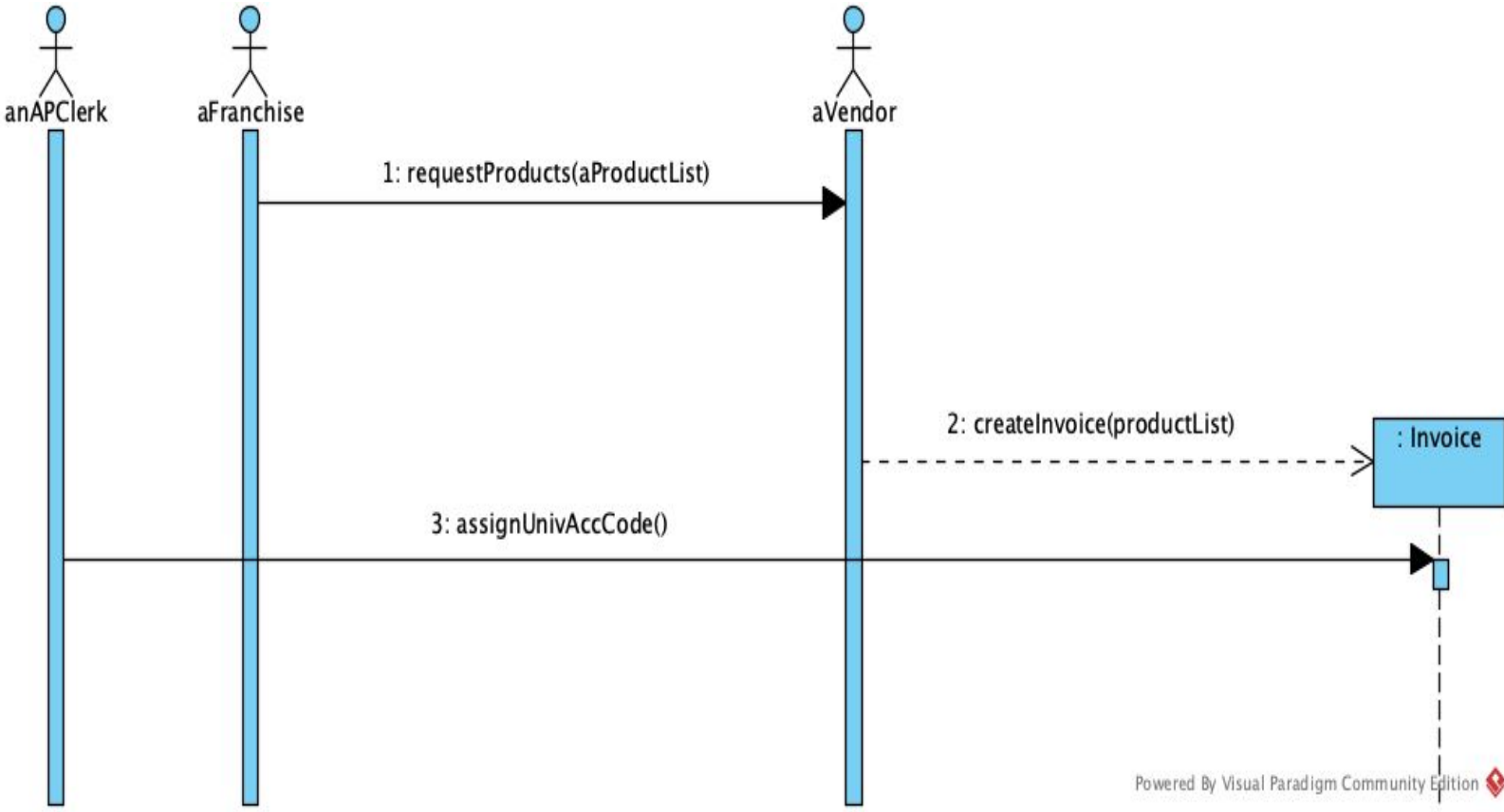
Update Vendor Address Sequence Diagram



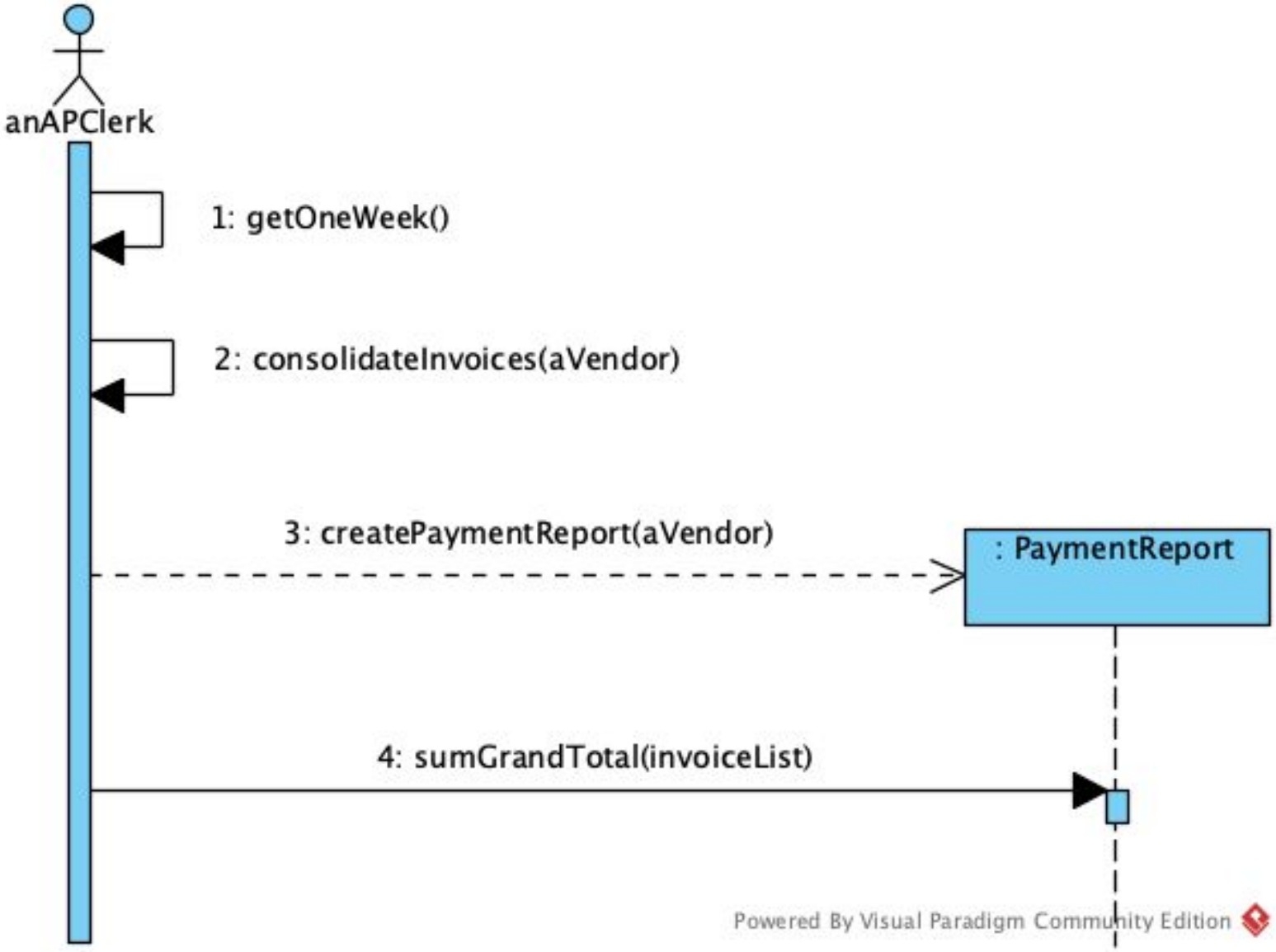
Delete Vendor Sequence Diagram



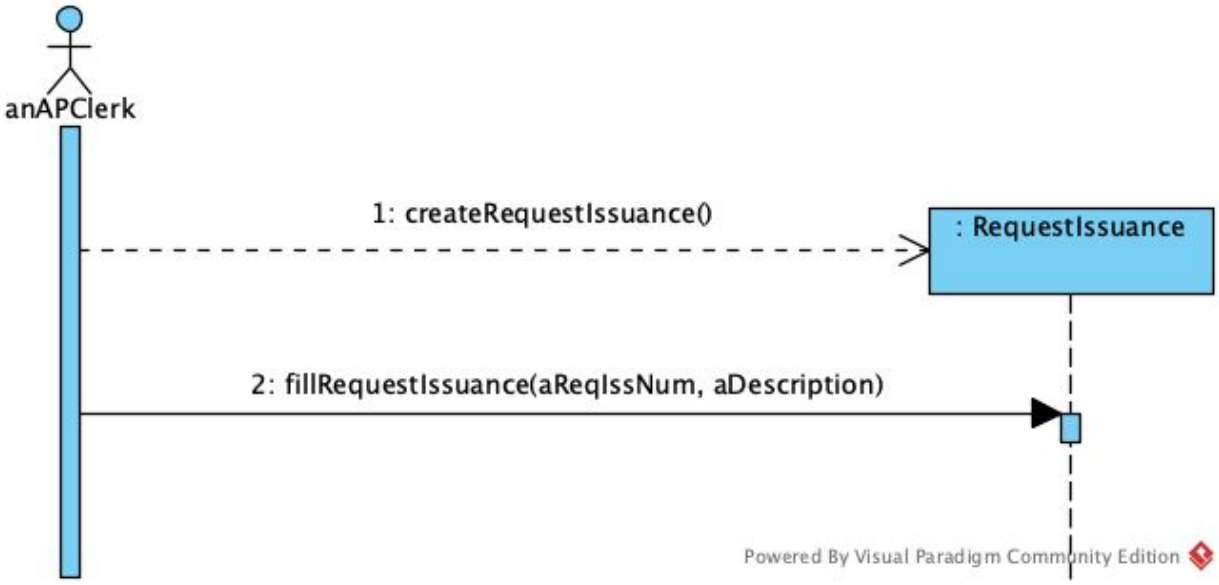
Create Invoice Sequence Diagram



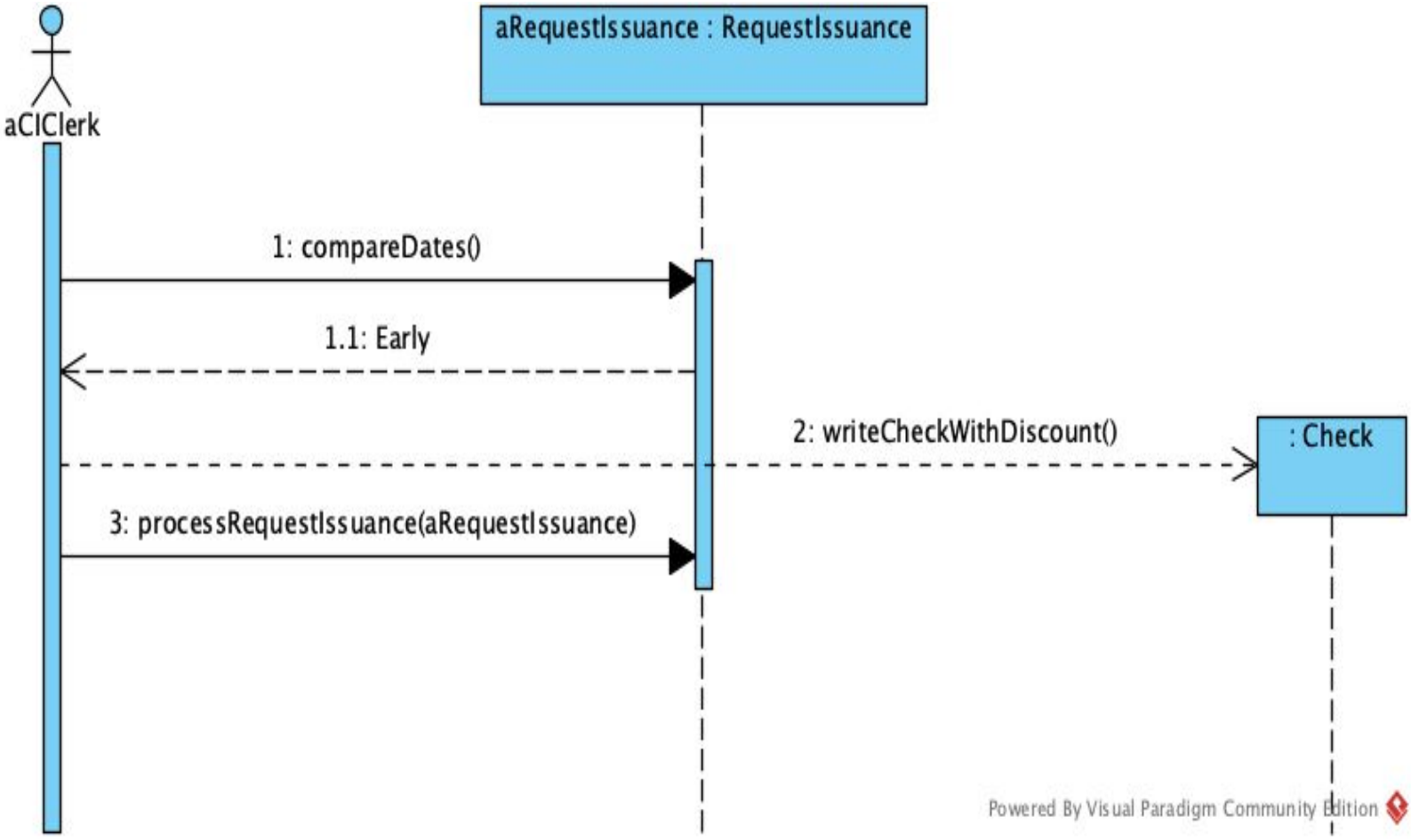
Create Payment Report Sequence Diagram



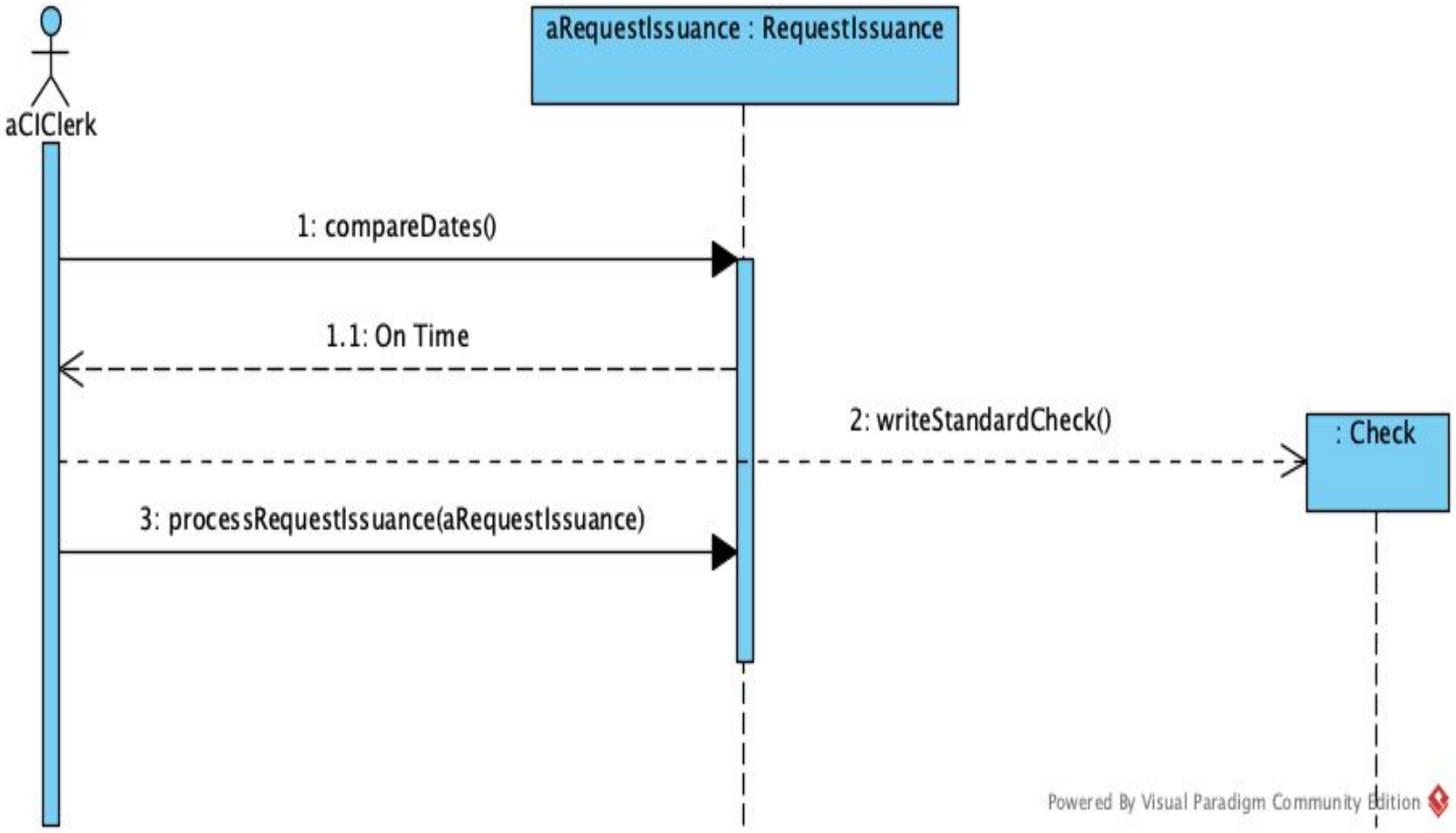
Create Request Issuance Sequence Diagram



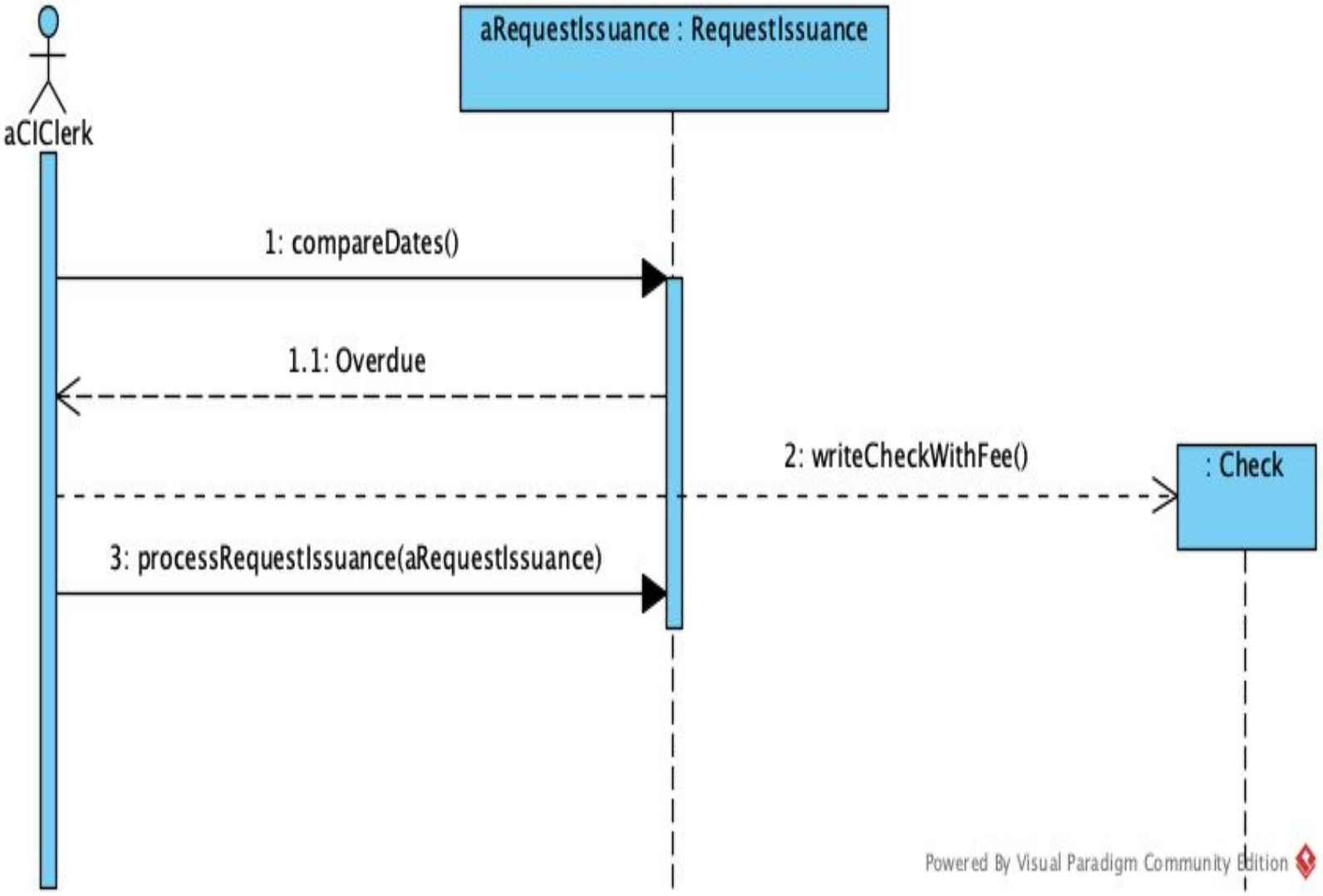
Write Check w/ Discount Sequence Diagram



Write Standard Check Sequence Diagram



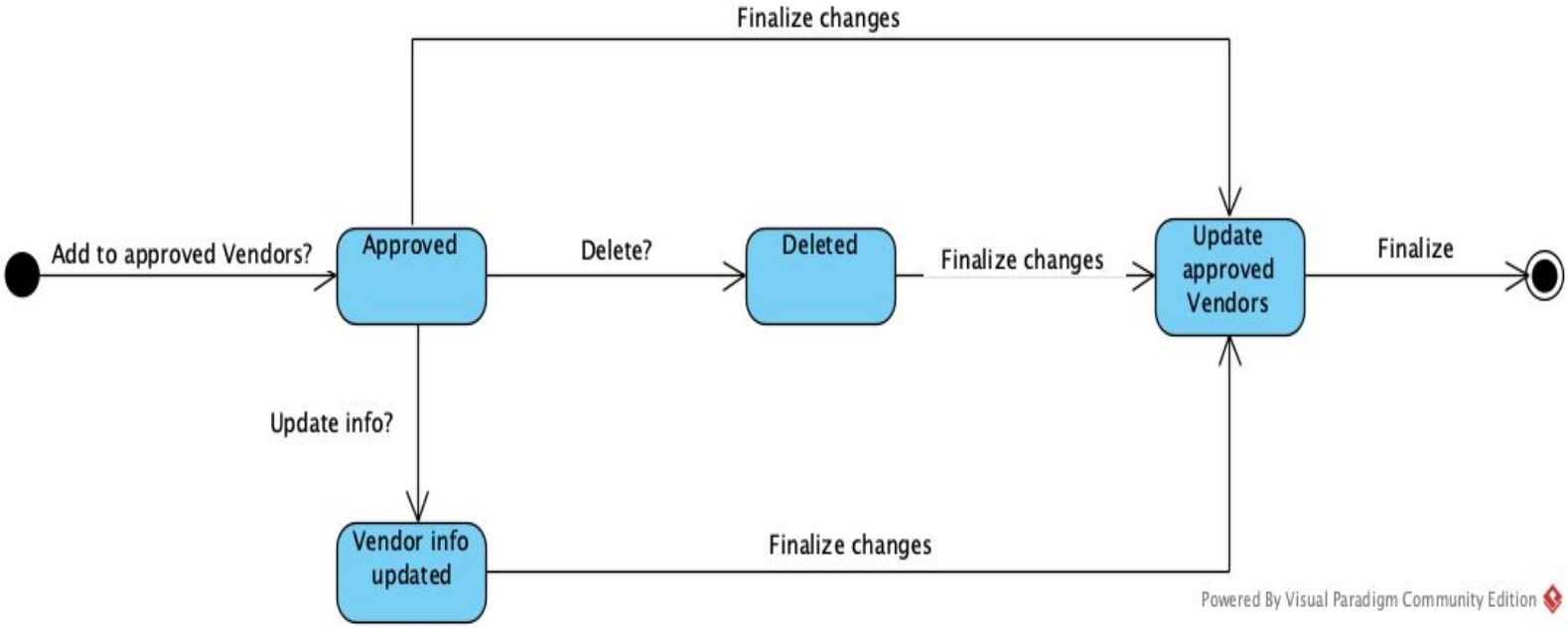
Write Check w/ Fee Sequence Diagram



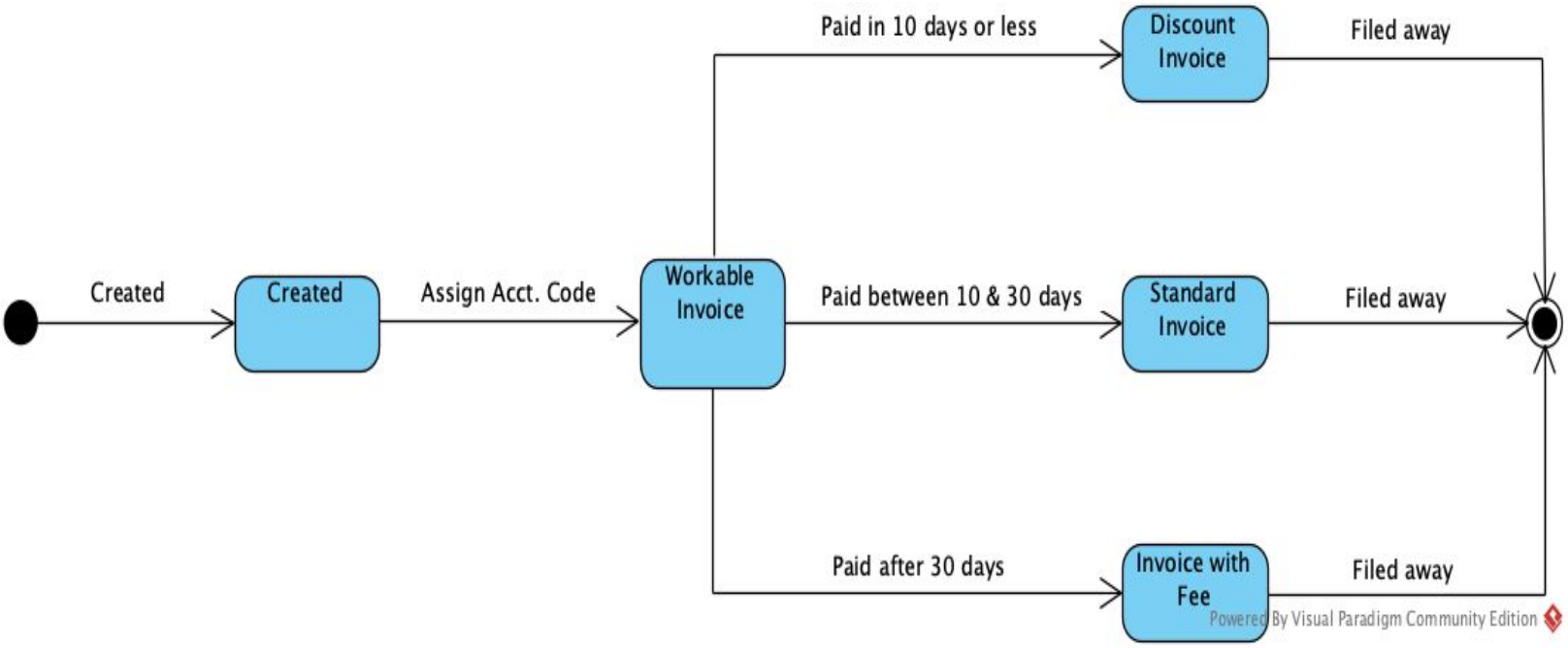
CRUDE Matrix

	University Center Ac	Vendor Actor	Franchise Actor	AP Office Actor (Accoun	Check Iss Office (Actor)	Product Class	Franchise Class	Invoice Class	Payment Report Class	Request Issuance Class	Check Class	Vendor Class
University Center Actor												CRUD
Vendor Actor						CRUD		CRUD			R	
Franchise/Store Actor												
AP Office Actor (Accountant)							R	RU	CRUD	CRUD	R	
Check Iss Office (Actor)							R	R	R	RU	CRUD	
Product Class												
Franchise/Store Class												
Invoice Class						R						
Payment Report Class												
Request Issuance Class												
Check Class												
Vendor Class												

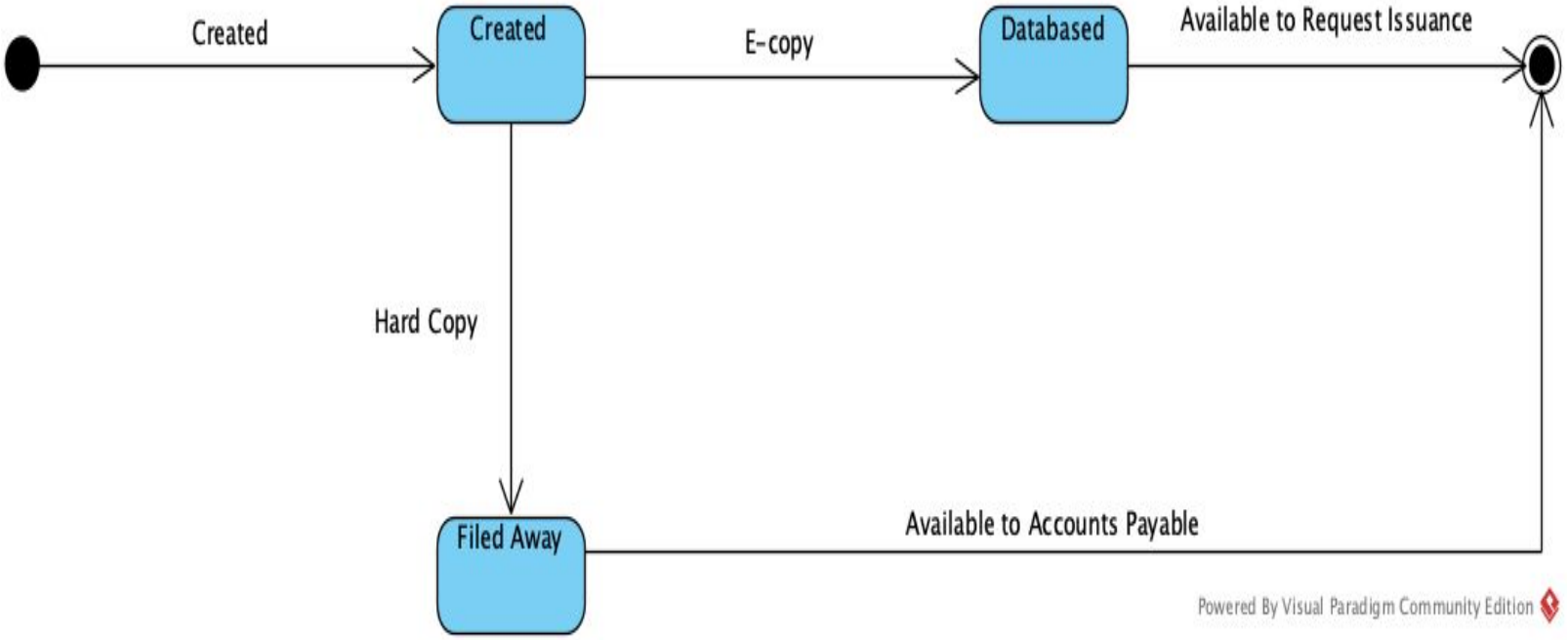
Vendor Behavioral State Machines



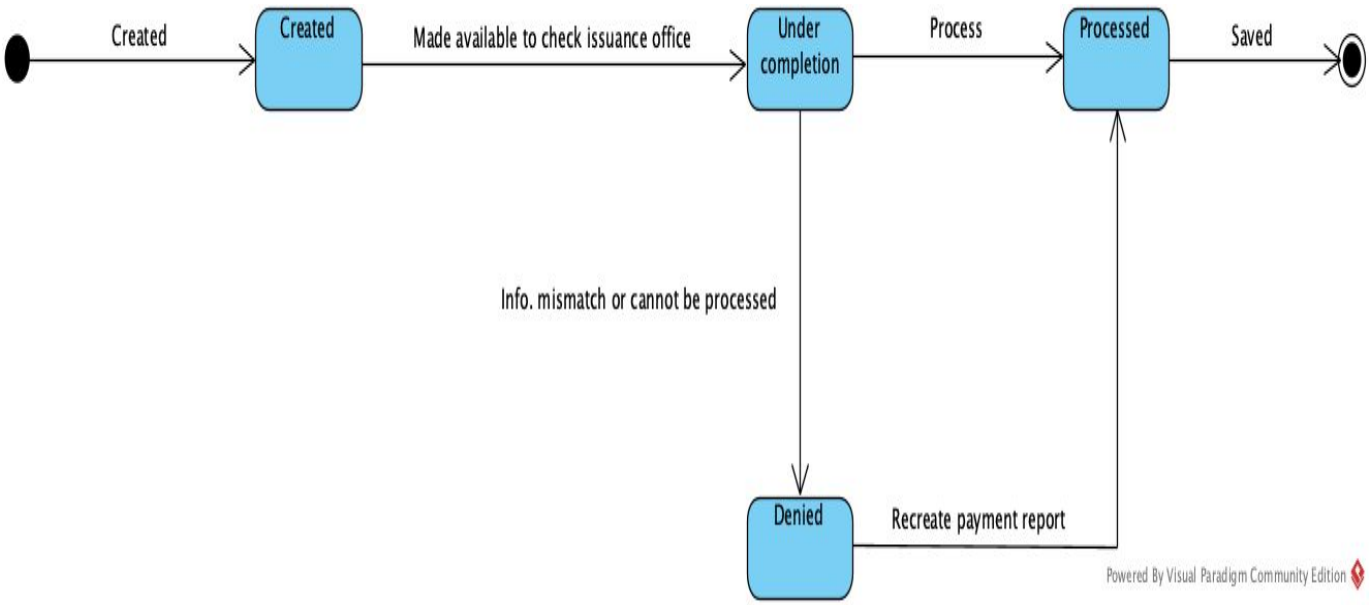
Invoice Behavioral State Machine



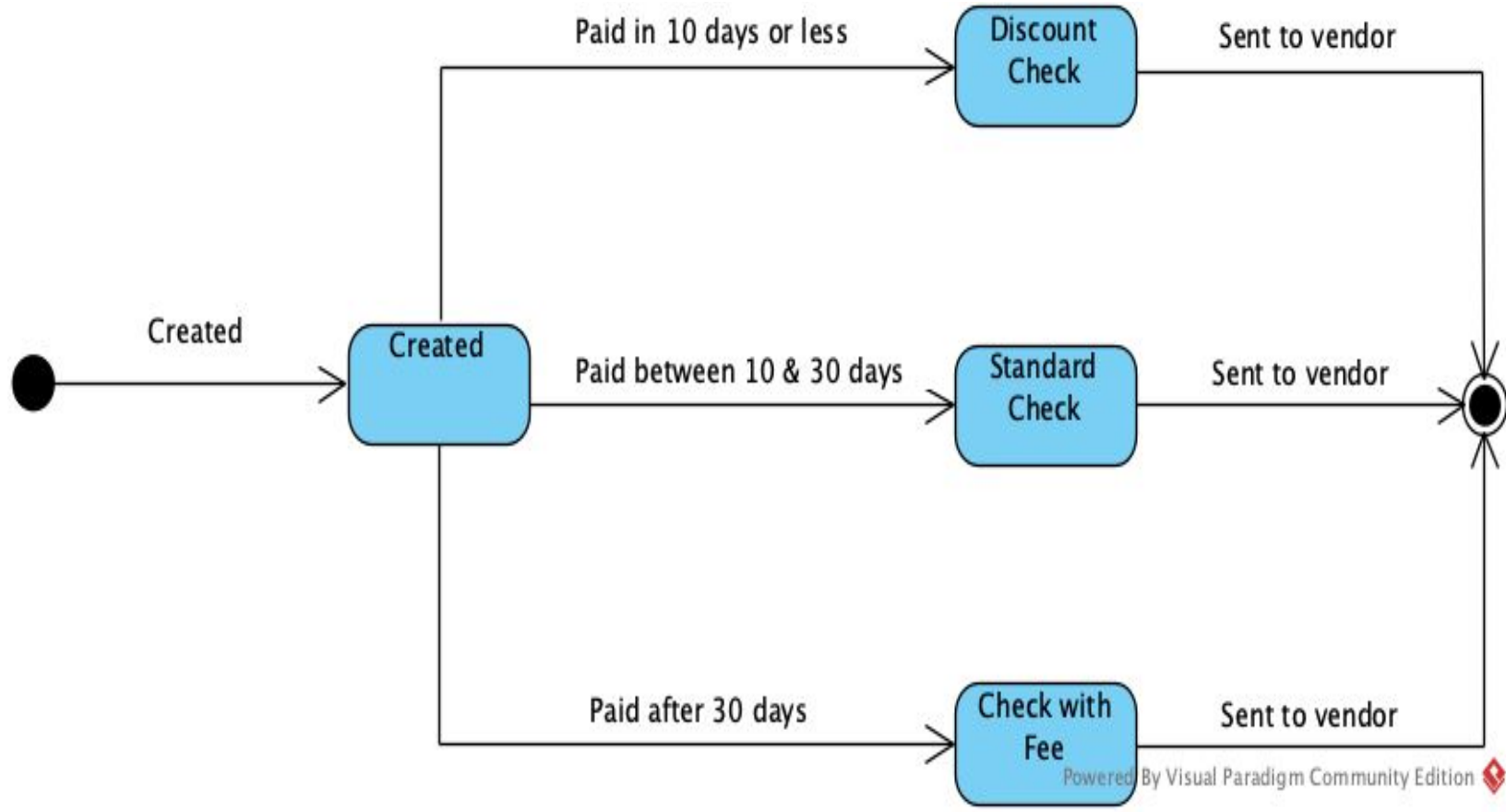
Payment Report Behavioral State Machine



Request Issuance Behavioral State Machine



Check Behavioral State Machine



Invariants

Invoice Class Invariants franchiseName = Franchise.getName() franchiseAddress = Franchise.getAddress() vendorName = Vendor.getName() totalDollarAmount = Product.productPrice.sum() taxAmount = Virginia.getTaxRate() * totalDollarAmount1
PaymentReport Class Invariants totalDollarAmount = Invoice.totalDollarAmount.sum() totalTaxAmount = Invoice.taxAmount.sum() finalTotal = totalDollarAmount + totalTaxAmount
RequestIssuance Class Invariants vendorName = PaymentReport.getVendorName() finalAmount = Check.getCheckAmount() checkNumber = Check. getCheckNumber()
Check Class Invariants vendorName = RequestIssuance.getVendorName

Method Contracts

Method Name: requestProduct	Class Name: Franchise	ID: 1
Clients (Consumers): Vendor		
Associated Use Cases: Create Invoice		
Description of Responsibilities: Implement the necessary behavior to add a new product to an existing invoice.		
Arguments Received: aProduct : Product		
Type of Value Returned: void		
Pre-Conditions: None		
Post-Conditions: None		

Method Name: addVendor	Class Name: Franchise	ID: 2
Clients (Consumers): UnivCenterClerk and Vendor		
Associated Use Cases: Manage Vendors, Add Vendor		
Description of Responsibilities: Implement the necessary behavior to add a new Vendor		
Arguments Received: name : String, address : string, vendorNumber : int		
Type of Value Returned: void		
Pre-Conditions: None		
Post-Conditions: None		

Method Name: updateVendorName	Class Name: Franchise	ID: 3
Clients (Consumers): UnivCenterClerk and Vendor		
Associated Use Cases: Manage Vendors, Change/Update Vendor		
Description of Responsibilities: Implement the necessary behavior to update the name attribute of an existing Vendor		
Arguments Received: vendor : Vendor, newName : string		
Type of Value Returned: void		
Pre-Conditions: vendor must already exist		
Post-Conditions: None		

Method Name: updateVendorAddress	Class Name: Franchise	ID: 4
Clients (Consumers): UnivCenterClerk and Vendor		
Associated Use Cases: Manage Vendors, Change/Update Vendor		
Description of Responsibilities: Implement the necessary behavior to update the address attribute of an existing Vendor		
Arguments Received: vendor : Vendor, newAddress : string		
Type of Value Returned: void		
Pre-Conditions: vendor must already exist		
Post-Conditions: None		

Method Name: deleteVendor	Class Name: Franchise	ID: 5
Clients (Consumers): UnivCenterClerk and Vendor		
Associated Use Cases: Manage Vendors, Delete Vendor		
Description of Responsibilities: Implement the necessary behavior to delete an existing Vendor		
Arguments Received: vendor : Vendor		
Type of Value Returned: void		
Pre-Conditions: vendor must already exist		
Post-Conditions: None		

Method Name: createInvoice	Class Name: Vendor	ID: 6
Clients (Consumers): Franchise and Invoice		
Associated Use Cases: Create Invoice		
Description of Responsibilities: Implement the necessary behavior to consolidate all of the products a Franchise requests at one time into one invoice		
Arguments Received: productList : Product (see assumptions)		
Type of Value Returned: void		
Pre-Conditions: None		
Post-Conditions: None		

Method Name: fillInvoice	Class Name: Invoice	ID: 7
Clients (Consumers): Invoice		
Associated Use Cases: Create Invoice		
Description of Responsibilities: Implement the necessary behavior to fill the products on the invoice with the appropriate accounting codes		
Arguments Received: aInvoice:Invoice		
Type of Value Returned: void		
Pre-Conditions: An Invoice must already exist with products on it		
Post-Conditions: None		

Method Name: fillIPaymentReport	Class Name: PaymentReport	ID: 8
Clients (Consumers): PaymentReport		
Associated Use Cases: Create payment report		
Description of Responsibilities: Implement the necessary behavior to fill a payment report with invoice objects that correspond to a specified time interval		
Arguments Received: invoiceList : Invoice (see assumptions)		
Type of Value Returned: PaymentReport		
Pre-Conditions: At least on Invoice object must already exist for a given time interval		
Post-Conditions: None		

Method Name: createPaymentReport	Class Name: Invoice	ID: 18
Clients (Consumers): PaymentReport		
Associated Use Cases: Create payment report		
Description of Responsibilities: Implement the necessary behavior to create a payment report with a template ready to be filled with invoice objects		
Arguments Received: VendorName : Vendor		
Type of Value Returned: PaymentReport		
Pre-Conditions: At least one vendor must have sent an invoice		
Post-Conditions: None		

Method Name: sumGrandTotal	Class Name: PaymentReport	ID: 9
Clients (Consumers): PaymentReport		
Associated Use Cases: Create Payment Report		
Description of Responsibilities: Implement the necessary behavior to calculate the grand total of a payment report by summing the total and tax amounts for every invoice on a given payment report.		
Arguments Received: invoiceList : Invoice (see assumptions)		
Type of Value Returned: double		
Pre-Conditions: At least one Invoice object must already exist on a payment report		
Post-Conditions: None		

Method Name: createRequestIssuance	Class Name: PaymentReport	ID: 10
Clients (Consumers): RequestIssuance		
Associated Use Cases: Create request issuance		
Description of Responsibilities: Implement the necessary behavior to create and fill a request issuance using an instance of a payment report		
Arguments Received: None		
Type of Value Returned: RequestIssuance		
Pre-Conditions: A payment report object must already exist with at least one invoice associated with it		
Post-Conditions: None		

Method Name: fillRequestIssuance	Class Name: RequestIssuance	ID: 11
Clients (Consumers): RequestIssuance		
Associated Use Cases: Create request issuance		
Description of Responsibilities: Implement the necessary behavior to fill the remaining fields of an instance of a request issuance		
Arguments Received: aReqIssNum : int, aDescription : string		
Type of Value Returned: void		
Pre-Conditions: None		
Post-Conditions: None		

Method Name: createCheck	Class Name: RequestIssuance	ID: 12
Clients (Consumers): Check		
Associated Use Cases: Write check		
Description of Responsibilities: Implement the necessary behavior to create a check using an existing request issuance		
Arguments Received: None		
Type of Value Returned: Check		
Pre-Conditions: None		
Post-Conditions: None		

Method Name: compareDates	Class Name: RequestIssuance	ID: 13
Clients (Consumers): RequestIssuance		
Associated Use Cases: Write check		
Description of Responsibilities: Implement the necessary behavior to determine whether the check amount should be written with a discount, fee, or neither		
Arguments Received: None		
Type of Value Returned: int		
Pre-Conditions: None		
Post-Conditions: None		

Method Name: fillCheck	Class Name: Check	ID: 14
Clients (Consumers): Check		
Associated Use Cases: Write check		
Description of Responsibilities: Implement the necessary behavior to fill the check with the correct amount based on the return value of the compareDates method of a given request issuance		
Arguments Received: None		
Type of Value Returned: void		
Pre-Conditions: None		
Post-Conditions: None		

Method Name: processRequestIssuance	Class Name: Check	ID: 15
Clients (Consumers): RequestIssuance		
Associated Use Cases: Write check		
Description of Responsibilities: Implement the necessary behavior to process a request issuance by filling the date payment approved, amount, and check number fields.		
Arguments Received: aRequestIssuance : RequestIssuance		
Type of Value Returned: void		
Pre-Conditions: The request issuance used in the parameter must already exist		
Post-Conditions: None		

Method Specifications

Method Name: requestProduct	Class Name: Franchise	ID: 100
Contract ID: 1	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Franchise requests a product		

Arguments Received: Data Type:	Notes:
productList : Product	The list of products the franchise needs

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
None		

Argument Returned: Data Type:	Notes:
void	
Algorithm Specification: Request a Product For all Products in Request Products DO Generate a new Product line in Invoice Add Product's Price to Sub-Total	
Misc.Notes:	

Method Name: addVendor	Class Name: Franchise	ID: 101
Contract ID: 2	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Franchise wishes to add a new approved vendor for products		

Arguments Received: Data Type:	Notes:
name : String address : String vendorNumber : Int	Must have vendor information before it can be added

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
Vendor.getVendorInfo() Vendor.createVendor()	Vendor	Get the new vendor info
	Vendor	Create a new vendor

Argument Returned: Data Type:	Notes:
void	

Algorithm Specification:
Add New Vendor to approved vendors
If Franchise has approved new Vendor info THEN
ADD New Vendor to VendorList
Else
No New Vendor could be approved	

Misc.Notes: Adding a new approved vendor	

Method Name: updateVendorName	Class Name: Franchise	ID: 102
Contract ID: 3	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Franchise wishes to update a preapproved vendor's information		

Arguments Received: Data Type:	Notes:
aVendor: Vendor newName : String	Must have existing vendor already

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
Vendor.getAVendor() Vendor.getNewName()	Product	Select a preapproved vendor
	Vendor	Getting new vendor name for the selected vendor

Argument Returned: Data Type:	Notes:
void	

Algorithm Specification:
Update Vendor Name
If Franchise has approved new Vendor Name for a preapproved vendor THEN
ADD New Vendor Name to aVendor
Else
No New Valid Name approved to aVendor	

Misc.Notes: This would normally happen in the case of a vendor being bought out by another company and getting renamed.	

Method Name: updateVendorAddress	Class Name: Franchise	ID: 103
Contract ID: 4	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Franchise wishes to update a preapproved vendor's address		

Arguments Received: Data Type:	Notes:
aVendor: Vendor vendorAddress: String	Must have existing vendor already

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
Vendor.getAVendor() Vendor.getNewAddress()	Product	Select a preapproved vendor
	Vendor	Getting new vendor info for the selected vendor

Argument Returned: Data Type:	Notes:
void	

Algorithm Specification:
Update Vendor Address
If Franchise has approved new Vendor Address for a preapproved vendor THEN
ADD New Vendor Address to aVendor
Else
No New Valid Address approved to aVendor	

Misc.Notes: This would normally happen in the case of a vendor moving locations	

Method Name: deleteVendor	Class Name: Franchise	ID: 104
Contract ID: 5	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Franchise wishes to delete a preapproved vendor's address		

Arguments Received: Data Type:	Notes:
aVendor:Vendor	Must have existing vendor already

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
Vendor.getAVendor() Vendor.deleteAVendor()	Product	Select a preapproved vendor
	Vendor	Deleting the selected vendor from the list of approved vendors

Argument Returned: Data Type:	Notes:
void	None

Algorithm Specification:
Delete Vendor
If Franchise has requested the removal of a preapproved vendor THEN
DELETE aVendor
Else
Do not delete aVendor	

Misc.Notes: This would normally happen when a vendor goes out of business or when a franchise stops using a vendor.	

Method Name: createPaymentReport	Class Name: Invoice	ID: 105
Contract ID: 8	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: One week has been completed and at least one invoice or more has been received from a singular vendor		

Arguments Received: Data Type:	Notes:
VendorName:Vendor	The name of the vendor whose payment report is being created

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
Invoice.getOneWeek()	None	None
Invoice.consolidateInvoices(vendor Name, aOneWeek)	Invoice	Gets the list of invoices from past week and a specific vendor name

Argument Returned: Data Type:	Notes:
aPaymentReport: PaymentReport	Generates an instance of payment report

Algorithm Specification:
Create a Payment Report
IF it has been one week since last payment report was created THEN
Compile list of invoices with one vendor name from last week
Else
No Payment Report can be created	

Misc.Notes: Get list of invoices compiled from one vendor and create payment report	

Method Name: Fill Invoice	Class Name: Invoice	ID: 106
Contract ID: 7	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Invoice has been created with at least one product		

Arguments Received: Data Type:	Notes:
aInvoice: Invoice	Must already have an invoice created to fill it

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
Invoice.getAListProduct(aInvoice)	Product	None
Invoice.AssignUnivAccCode()	Unsigned Long	Assigns accounting code to products and fills the invoice with the correct codes

Argument Returned: Data Type:	Notes:
void	None

Algorithm Specification:
Fill Invoice
IF aInvoice exists AND has ATLEAST one Product(s) THEN
Assign AccountingCode to matching Product
Else
No Accounting Codes to sign	

Misc.Notes: Invoices are scanned onto database first	

Method Name: sumGrandTotal	Class Name: PaymentReport	ID: 107
Contract ID: 9	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Payment Report has been created for aVendor		

Arguments Received: Data Type:	Notes:
aInvoiceList: Invoice	Must already have aListInvoices with associated totals in the payment report

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
None	None	None

Argument Returned: Data Type:	Notes:
GrandTotal:Double	None
Algorithm Specification: Sum Grand Total For every Invoice in the payment report Sum the total AND Tax Else No Grand Total could be calculated	
Misc.Notes: None	

Method Name: compareDates	Class Name: RequestIssuance	ID: 108
Contract ID: 13	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Check Issuance Clerk checks if check should be written with a discount, fee, or neither.		

Arguments Received: Data Type:	Notes:
None	None

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
aPaymentReport.getInvoiceList() Earliest Date = aPaymentReport.getEarliestDate() aPaymentReport.dateDifference(Earliest Date, Current Date)	Date	There are two dates being used; the date of the earliest received invoice from the past week and the current date

Argument Returned: Data Type:	Notes:
0,1,2 : int	To show if the process is on time, early, or late. 0 corresponds to receiving a discount, 1 to standard payment, and 2 is a late fee on payment.
Algorithm Specification: Get list of invoices from past week Get the date from the earliest received invoice Compare the date of the earliest received invoice to the current date CASE IF Earliest Date - Current Date <= 10: Return 0 IF Earliest Date - Current Date > 10 AND <= 30: Return 1 IF Earliest Date - Current Date > 30: Return 2	

ENDCASE
Misc. Notes: This will happen when a Check Issuance clerk is determining to write a check with a discount, fee, or neither.

Method Name: fillRequestIssuance	Class Name: Request Issuance	ID: 109
Contract ID: 11	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Request Issuance has been created		

Arguments Received: Data Type:	Notes:
aReqIssNum : Int aDescription : String	Must already have a req issuance number to fill

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
RequestIssuance.AssignReqIssNumber()	Int	The requestIssuanceNumber attribute in the RequestIssuance class is set equal to the aReqIssNum
RequestIssuance.AssignADescription()	String	The productsDescription attribute in the RequestIssuance class is set equal to the aDescription argument that is passed in the method

Argument Returned: Data Type:	Notes:
void	None
Algorithm Specification: Fill Request Issuance aRequestIssuance.requestIssuanceNumber = aReqIssNum aRequestIssuance.productsDescription = aDescription	
Misc.Notes: None	

Method Name: fillCheck	Class Name: Check	ID: 110
Contract ID: 14	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Check Issuance Clerk has finished comparing dates and has all the info needed to fill check		

Arguments Received: Data Type:	Notes:
CompareDate: Int	To show amount written on check
CheckNumber: Check	Check number to ID the check

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
Check.getCompareDate()	Date	Comparison date

Argument Returned: Data Type:	Notes:
Void	None
Algorithm Specification: Fill Check IF CheckNumber = aRequestIssuance.getcheckNumber invoiceTotal = aPaymentReport.sumGrandTotal() CASE IF CompareDate = 0: Fill Check with 98% of invoiceTotal IF CompareDate = 1: Fill Check with invoice Total IF CompareDate = 2: Fill Check with 102% of invoiceTotal ELSE Return with an error asking user to try again ENDCASE ELSE Return error	
Misc. Notes: Fills the amount written on the check. Error is returned when the check number recorded	

Method Name: processRequestIssuance	Class Name: Check	ID: 111
Contract ID: 15	Programmer: John Doe	Date Due: 12/10/19
Programming Language: • Visual Basic Smalltalk C++ Java		
Triggers/Events: Check has been written		

Arguments Received: Data Type:	Notes:
aCheck:Check	Must have a check created and its information ready

Messages Sent & Arguments Passed: ClassName.MethodName:	Argument Data Type:	Notes:
Check.getARequestIssuance()	N/A	None
RequestIssuance.fill()	N/A	None

Argument Returned: Data Type:	Notes:
void	None
Algorithm Specification: Process Request Issuance If aCheck is written aCheck.checkAmount = aRequestIssuance.finalAmount aCheck.checkNumber = aRequestIssuance.checkNumber aCheck.datePaid = aRequestIssuance.dateApprovedForPayment Paste fields onto a request issuance to process it Else Processing failed	
Misc.Notes: None	

Assumptions and Meeting Reports

1. We combine the two separate software and create one fully functioning system
2. Franchise and store and interchangeable terms
3. Invoice are scanned
4. All documents and reports when generated are uploaded to database
5. Product Class does not show operations only has getters, setters and destructors
6. Requesting Products from vendor will generate all products requested in one productList to go on the Invoice
7. Vendors always have requested products
8. Create invoice method takes a list of products as a parameter
9. The create payment report takes a list of invoices as a parameter
10. The discount is only applied if every invoice on a payment report falls within the discount period which is within 10 days after receiving an invoice. This is why the comparison of dates utilizes the earliest received invoice on a payment report.
11. The scheduledPaymentDate attribute of aRequestIssuance is set equal to 10 days after the date of the earliest received invoice on the corresponding payment report. This is so all offices can expect invoices to be paid within the discount period which is 10 days after receiving an invoice. As a result, the university can save the most amount of money. If the earliest received invoice is early then all the invoices preceding it are also early. This also applies for invoices that are on time and late. For example, if a payment report is made up of 7 invoices, the first of which is received on 1/1/20 and the rest are received after that date, the scheduledPaymentDate of the request issuance that corresponds to that payment report is equal to 1/11/20.
12. The datePaid attribute on the check class is the same as the dateApprovedforPayment attribute on the RequestIssuance class. When aRequestIssuance is processed the dateApprovedforPayment attribute on the RequestIssuance class is set equal to the datePaid attribute from aCheck.
13. The compare dates method in the request issuance class uses the date of the earliest received invoice on the corresponding payment report for comparison. For example, if a payment report is made up of 7 invoices, the first of which is received on 1/1/20 and the rest are received after that date, the compare dates method will compare the current date with 1/1/20 to determine whether a discount, fee, or neither is applied.