Colorado State University GLOBAL CAMPUS

CSC470: SOFTWARE ENGINEERING

Credit Hours: 3

Contact Hours: This is a 3-credit course, offered in accelerated format. This means that 16 weeks of material is covered in 8 weeks. The exact number of hours per week that you can expect to spend on each course will vary based upon the weekly coursework, as well as your study style and preferences. You should plan to spend 14-20 hours per week in each course reading material, interacting on the discussion boards, writing papers, completing projects, and doing research.

Faculty Information: Faculty contact information and office hours can be found on the faculty profile page.

COURSE DESCRIPTION AND OUTCOMES

Course Description:

This course introduces students to software engineering concepts and skills needed to apply software process models to software development. Emphasis will be on the evaluation of system requirements and the analysis of software code components. Students will learn how to develop software testing guidelines to ensure quality and how to effectively communicate with clients.

Course Overview:

This software engineering course focuses on understanding key concepts related to the practices associated with object-oriented software design. It examines the methods required to develop object-based software applications and equip students to analyze application requirements, translate application requirements into standardized software schematics, and develop code using object-oriented design patterns. Students gain an understanding of contemporary software design methodologies.

Course Learning Outcomes:

1. Discuss prescriptive and Agile software process models.
2. Specify software design requirements for a software project.
3. Identify best practices in developing a large-scale software project.
4. Utilize coding techniques appropriate for an identified software project.
5. Demonstrate quality assurance techniques associated with software development.
6. Employ standards for communicating with a client in a software development environment.

PARTICIPATION & ATTENDANCE

Prompt and consistent attendance in your online courses is essential for your success at CSU-Global Campus. Failure to verify your attendance within the first 7 days of this course may result in your withdrawal. If for some reason you would like to drop a course, please contact your advisor.

Online classes have deadlines, assignments, and participation requirements just like on-campus classes. Budget your time carefully and keep an open line of communication with your instructor. If you are having technical problems, problems with your assignments, or other problems that are impeding your progress, let your instructor know as soon as possible.

Course Materials

Required:

- Unhelkar, B. (2018) *Software engineering with UML.* New York: CRC Press. ISBN 978-1138297432
- PlantUML. (n.d.). PlantUML language reference guide. Retrieved from http://plantuml.com/download

Additional Required Course Materials or Web-based Tools/Applications:

One of the following UML Tools:

- UMLet - http://www.umlet.com/
- Gliffy - https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693
- Microsoft Visio - https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600
- PlantUML - http://plantuml.com/class-diagram

In addition, Modules 1 – 6 require the following C++ development tool:

- Microsoft Visual Studio 2017, Community Edition - https://www.visualstudio.com

NOTE: *All non-textbook required readings and materials necessary to complete assignments, discussions, and/or supplemental or required exercises are provided within the course itself. Please read through each course module carefully.*

Course Schedule

Due Dates

The Academic Week at CSU-Global begins on Monday and ends the following Sunday.

- **Discussion Boards:** The original post must be completed by Thursday at 11:59 p.m. MT and peer responses posted by Sunday at 11:59 p.m. MT. Late posts may not be awarded points.
- **Opening Exercises:** Take the Opening Exercise before reading each week's content to see which areas you will need to focus on. You may take these exercises as many times as you need. The Opening Exercises will not affect your final grade.
- **Mastery Exercises:** Students may access and retake Mastery Exercises through the last day of class until they achieve the scores they desire.
- **Critical Thinking:** Assignments are due Sunday at 11:59 p.m. MT.

Weekly Reading and Assignment Details

Module 1

Readings

· Chapter 1 in *Software Engineering with UML*

Opening Exercise (0 points)

Discussion (25 points)

Critical Thinking (60 points)

Option 1

Analyzing Object-Oriented Inheritance Hierarchy: Designing A Fast Food Combo Meal

For this course you will be working with Visual Studio 2017 (link below). Download the application for use in all assignments. If you have questions on how to use Visual Studio, contact your instructor directly.

Consider the following composition diagram:

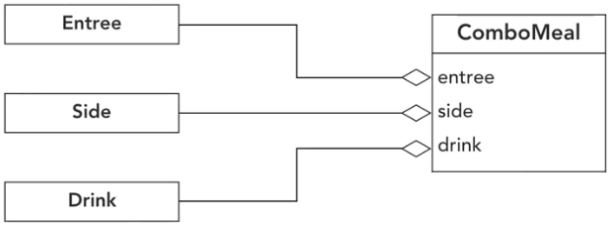


Image Caption:

```
@startuml
Class ComboMeal {
                + Entree entree
                + Side side
                + Drink drink
}
ComboMeal o-- Entree
ComboMeal o-- Side
ComboMeal o-- Drink
@enduml
```

Note for UML Captioning

An informal standard exists for captioning UML diagrams. A special syntax exists for describing and generating UML diagrams (link below). Details are provided in the *Drawing UML with Plant UML* reference guide on how to caption UML diagrams.

If you are unfamiliar with UML composition diagrams, refer to its definition found at UML Composition (link below). Using the attached Visual Studio project, implement a "ComboMeal" class, as shown in the UML diagram, which includes a console print statement for the contents of a combo meal.

Compile and submit all of your work into a single zip file.

Links:

- Visual Studio (https://www.visualstudio.com)
- UML Composition (https://www.uml-diagrams.org/composition.html)

- Drawing UML with Plant UML (http://plantuml.com/download)
- CT_VisualStudioProject_Mod01 (Download zip file in Module 1)

Option 2

Not applicable. There is one CT assignment for each module throughout this course.

Mastery Exercise (10 points)

Portfolio Milestone (0 points)

Throughout this course you will learn software engineering design concepts and skills used to develop robust, turn-key software solutions. Review the Portfolio Project prompt in the Module 8 Materials folder and begin thinking about how you would model a software solution for the system specified.

Module 2

Readings

- Chapters 2 and 3 in *Software Engineering with UML*

Opening Exercise (0 points)

Discussion (25 points)

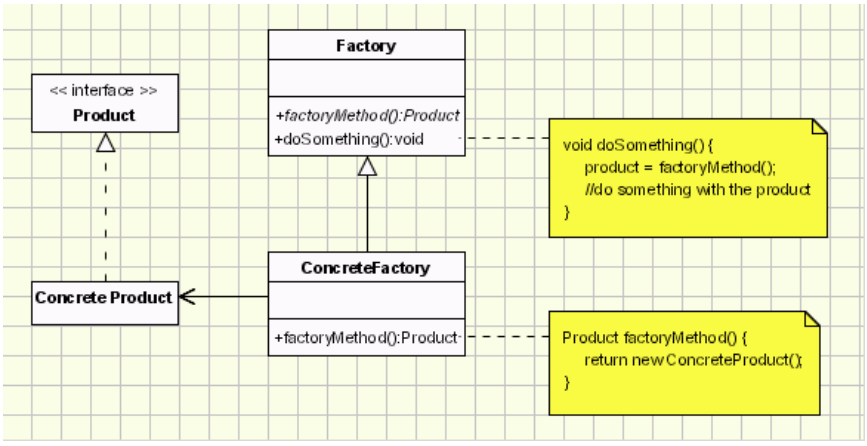
Critical Thinking (60 points)

Option 1

Factory Pattern

For this course you will be working with Visual Studio 2017 (link below). If you have not done so already, download the application for use in all assignments. If you have questions on how to use Visual Studio, contact your instructor directly.

Consider the following UML diagram that represents a software design pattern known as a Factory Pattern.



(Source: https://www.oodesign.com/factory-method-pattern.html)

Image Caption:

@startuml Product

```
Class Factory {
        + factoryMethod() : Product
        + doSomething() : void
}
Class ConcreteFactory {
        + factoryMethod() : Product
}
note right of Factory #yellow
void doSomething()
{
        product = factoryMethod();
        //do something with the product
}
end note
note right of ConcreteFactory #yellow
Product factoryMethod()
{
        return new ConcreteProduct();
}
end note
Factory <|-- ConcreteFactory
ConcreteFactory -left-> ConcreteProduct
Product <|-- ConcreteProduct
@enduml
```

Part 1: Using the UML diagram template for a factory pattern, draft a factory pattern diagram for a CoffeeMakerFactory that prompts a user to select a type of coffee she likes and returns the object of what she selected. You may use the C++ source file in the attached Visual Studio project as a guide for drafting your UML diagram.

Use one of the following UML tools (links below) to create your diagrams for this assignment:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)

Part 2: Using the Module 2 Visual Studio project, implement a CoffeeMakerFactory class that prompts the user to select a type of coffee she likes and returns the object of what she selected.

Compile and submit all of your work into a single zip file.

Image Credit: Based off of illustrations explained in the *Drawing UML with Plant-UML* reference guide. Retrieved from http://plantuml.com/PlantUML_Language_Reference_Guide.pdf ©Open Source.

Links:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)
- CT_VisualStudioProject_Mod02 (Download zip file in Module 2)

Option 2

Not applicable. There is one CT assignment for each module throughout this course.

Mastery Exercise (10 points)

Module 3

Readings

Chapters 5 and 6 in *Software Engineering with UML*

Opening Exercise (0 points)

Discussion (25 points)

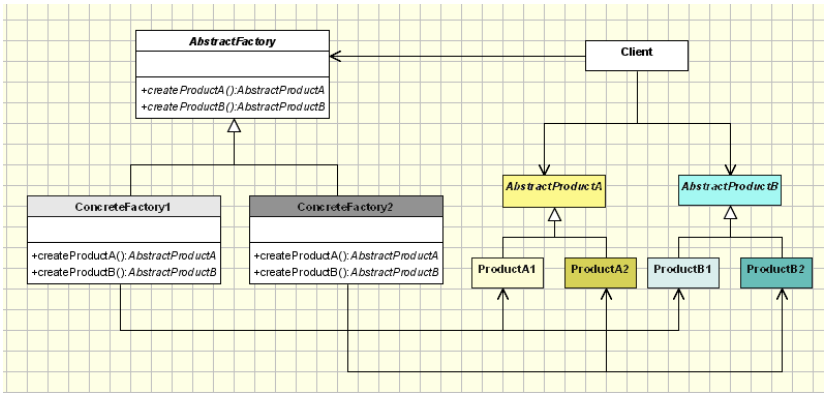
Critical Thinking (60 points)

Option 1

Abstract Factory Pattern

For this course you will be working with Visual Studio 2017 (link below). If you have not done so already, download the application for use in all assignments. If you have questions on how to use Visual Studio, contact your instructor directly.

Consider the following UML diagram that represents a software design pattern known as an Abstract Factory Pattern.



(Source: https://www.oodesign.com/abstract-factory-pattern.html)

Image Caption:

@startuml
abstract class AbstractProductA
abstract class AbstractProductB

```
abstract class AbstractFactory {
        + createProductA() : AbstractProductA
        + createProductB() : AbstractProductB
}
Class ConcreteFactory1 {
        + createProductA() : AbstractProductA
        + createProductB() : AbstractProductB
}
Class ConcreteFactory2 {
        + createProductA() : AbstractProductA
        + createProductB() : AbstractProductB
}
Client -right-> AbstractFactory
Client -down-> AbstractProductA
Client -down-> AbstractProductB
AbstractFactory <|-- ConcreteFactory1
AbstractFactory <|-- ConcreteFactory2
AbstractProductA <|-- ProductA1
AbstractProductA <|-- ProductA2
AbstractProductB <|-- ProductB1
AbstractProductB <|-- ProductB2
ConcreteFactory1 -down-> ProductA1
ConcreteFactory1 -down-> ProductA2
ConcreteFactory2 -down-> ProductB1
ConcreteFactory2 -down-> ProductB2
@enduml
```

The abstract factory design goal is to have different concrete factories creating similar objects, but the exact construction of the objects is defined differently by each factory which allows objects that belong the same suite or family to be kept together.

Using the UML diagram template for an abstract factory pattern, draft an abstract factory pattern diagram for two car factories, on for a gas-powered car, and one for an electric car. The gas-powered car factory will have its own implementation of a car door and its own gas-powered engine and the electric car factory will have its own implementation of a car door and its own electric engine. You may use the C++ source file in the attached Visual Studio project as a guide for drafting your UML diagram.

Use one of the following UML tools (links below) to create your diagrams for this assignment:

- UMLet
  (http://www.umlet.com/)
- Gliffy
  (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio
  (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)

Compile and submit all of your work into a single zip file.

Links:

- UMLet
  (http://www.umlet.com/)

- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)
- CT_VisualStudioProject_Mod03 (Download zip file in Module 3)

Option 2

Not applicable. There is one CT assignment for each module throughout this course.

Mastery Exercise (10 points)

Module 4

Readings

- Chapter 7 in *Software Engineering with UML*

Opening Exercise (0 points)

Discussion (25 points)

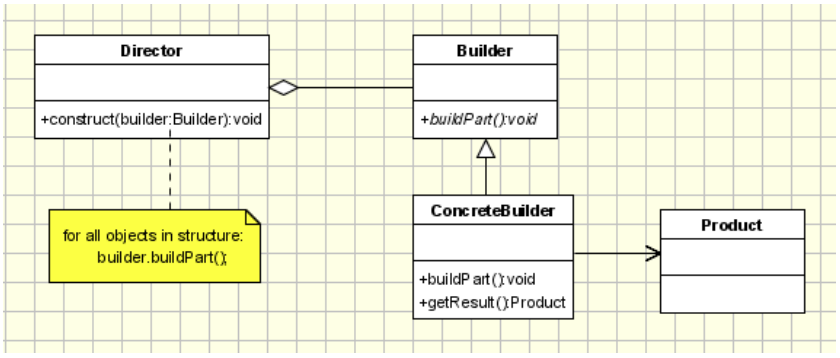
Critical Thinking (60 points)

Option 1

Builder Design Pattern

For this course you will be working with Visual Studio 2017 (link below). If you have not done so already, download the application for use in all assignments. If you have questions on how to use Visual Studio, contact your instructor directly.

Consider the following UML diagram that represents a software design pattern known as a Builder Design Pattern.



(Source: https://www.oodesign.com/builder-pattern.html)

Image Caption:

```
@startuml
Class Director {
        + construct(builder : Builder) : void
}
Class Builder {
        + buidPart() : void
```

```
}
Class ConcreteBuilder {
        + buildPart() : void
        + getResult() : Product
}
Builder <|-- ConcreteBuilder
ConcreteBuilder -right-> Product
Director <|-- Builder
note bottom of Director #yellow
for all objects in structure:
        builder.buildPart();
end note
@enduml
```

The Builder Design Pattern allows us to build complex objects by separating the construction of the object from its representation. Unlike the factory or abstract factory, which builds your objects in one shot, this allows us to build a complex object one step at a time. It also allows us to use the same step or process to create different representations of the complex object. This is a particularly useful design when creating an object that is made up of other objects.

Using the UML diagram template for a builder pattern, draft a UML diagram for a combo meal served at a fast food restaurant. A combo meal has one entrée consist of either a burger or hotdog, one side consisting or either fries or a salad, and on drink. You may use the C++ source file in the attached Visual Studio project as a guide for drafting your UML diagram.

Use one of the following UML tools (links below) to create your diagrams for this assignment:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)

Compile and submit all of your work into a single zip file.

Links:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)
- CT_VisualStudioProject_Mod04 (Download zip file in Module 4)

<u>Option 2</u>

Not applicable. There is one CT assignment for each module throughout this course.

Mastery Exercise (10 points)

Portfolio Milestone (0 points)

Make appropriate corrections to the code you submitted in Modules 1 – 3 (Critical Thinking Assignments). Resubmit your programs from Modules 1 – 3 with all outlined corrections.

Module 5

Readings

- Chapter 11 in *Software Engineering with UML*

Opening Exercise (0 points)

Discussion (25 points)

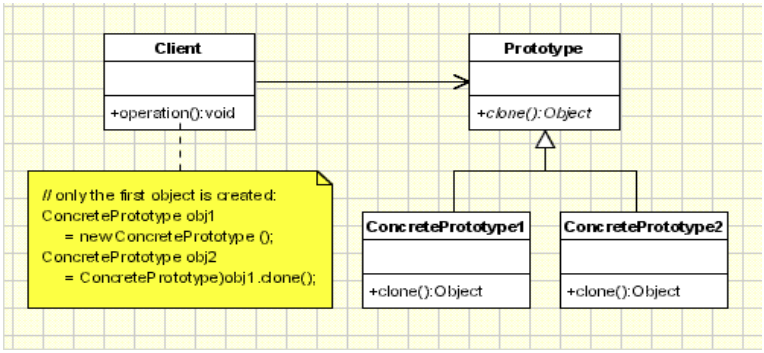
Critical Thinking (80 points)

Option 1

Prototype Design Pattern

For this course you will be working with Visual Studio 2017 (link below). If you have not done so already, download the application for use in all assignments. If you have questions on how to use Visual Studio, contact your instructor directly.

Consider the following UML diagram that represents a software design pattern known as a Prototype Design Pattern.



(Source: https://www.oodesign.com/builder-pattern.html)

Image Caption:

```
@startuml
Class Client {
        + operation() : void
}
Class Prototype {
        + clone() : Object
}
Class ConcretePrototype1 {
        + clone() : Object
}
Class ConcretePrototype2 {
        + clone() : Object
}
```

```
Prototype <|-- ConcretePrototype1
Prototype <|-- ConcretePrototype2
Client -right-> Prototype
note bottom of Client #yellow
        // only the first object is created
        ConcretePrototype obj1
                = new ConcretePrototype();
        ConcretePrototype obj2
                = (ConcretePrototype)obj1.clone;

end note
@enduml
```

The Prototype design pattern allows an object to create customized objects without knowing their class or any details of how to create them. Up to this point it sounds a lot like the Factory Method pattern, the difference being the fact that for the Factory the palette of prototypical objects never contains more than one object.

Using the UML diagram template for a prototype pattern, draft a UML diagram for cloning an animal. The abstract *Animal class* is specified in the C++ file in the attached Visual Studio project. You may use the C++ source file in the attached Visual Studio project as a guide for drafting your UML diagram.

Use one of the following UML tools (links below) to create your diagrams for this assignment:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)

Links:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)
- CT_VisualStudioProject_Mod05 (Download zip file in Module 5)

Option 2

Not applicable. There is one CT assignment for each module throughout this course.

Mastery Exercise (10 points)

Portfolio Milestone #1 (0 points)

Pre-Planning Milestone #1
For your final project, you are going to create a UML State Machine Diagram (SMD) for an automated teller machine (ATM). Your diagram must include the following:

- The customer must pass authentication before withdrawing money
- Authentication is performed by checking a PIN
- The PIN can be correct or not
- Unsuccessful attempts are counted
- If the counter exceeds a limit, then the customer is rejected

If the account balance is zero, then the account is closed.

For this pre-planning milestone, you will want to create appropriate classes that will represent the ATM and identify the attributes that will be needed. Submit this assignment for feedback and also include it with your Portfolio Project submission in Module 8.

Module 6

Readings

· Chapter 12 in *Software Engineering with UML*

Opening Exercise (0 points)

Discussion (25 points)

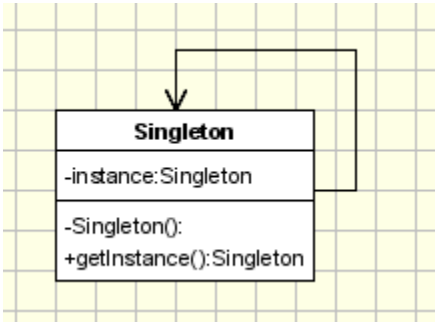
Critical Thinking (50 points)

Option 1

Singleton Design Pattern

For this course you will be working with Visual Studio 2017 (link below). If you have not done so already, download the application for use in all assignments. If you have questions on how to use Visual Studio, contact your instructor directly.

Consider the following UML diagram that represents a software design pattern known as a Singleton Design Pattern.



(Source: https://www.oodesign.com/builder-pattern.html)

Image Caption:

```
@startuml
Class Singleton {
        - instance : Singleton
        - Singleton() :
        + getInstance() : Singleton
}
Singleton -right-> Singleton
@enduml
```

The singleton pattern is one of the simplest design patterns. It involves only one class which is responsible to instantiate itself, to make sure it creates not more than one instance. At the same time it provides a global point of access to that instance. In this case the same instance can be used from everywhere, being impossible to invoke directly the constructor each time.

Part 1: Using the UML diagram template for a singleton pattern, draft a UML diagram for a Leader object. The abstract Leader class is specified in the C++ file in the attached Visual Studio project. You may use the C++ source file in the attached Visual Studio project as a guide for drafting your UML diagram.

Use one of the following UML tools (links below) to create your diagrams for this assignment:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)

Part 2: Using the attached Visual Studio project, explore the main method in the C++ source file and note how the instance of a *Leader* class is instantiated.

a. In the main method, create another *Leader* named *elected* and have *elected* give a speech.
b. Enter a comment in your C++ program explaining why your program is not thread safe and then make your program thread safe.

Compile all of your work into a single document and submit as a Word or PDF file.

Links:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)
- CT_VisualStudioProject_Mod06 (Download zip file in Module 6)

Option 2

Not applicable. There is one CT assignment for each module throughout this course.

Mastery Exercise (10 points)

Module 7

Readings

- Chapter 13 in *Software Engineering with UML*

Opening Exercise (0 points)

Discussion (25 points)

Mastery Exercise (10 points)

Portfolio Milestone (0 points)

Make appropriate corrections to the code you submitted in Modules 4 – 6 (Critical Thinking Assignments). Resubmit your programs from Modules 4 – 6 with all outlined corrections.

Module 8

Readings

- Chapter 14 in *Software Engineering with UML*

Opening Exercise (0 points)

Discussion (25 points)

Mastery Exercise (10 points)

Portfolio Project (350 points)

Part 1. Final Project

For your final project, you are going to create a UML State Machine Diagram (SMD) for an automated teller machine (ATM). Your diagram must include the following:

- The customer must pass authentication before withdrawing money
- Authentication is performed by checking a PIN
- The PIN can be correct or not
- Unsuccessful attempts are counted
- If the counter exceeds a limit, then the customer is rejected
- If the account balance is zero, then the account is closed

Annotate your state box in one of two ways. For internal state transitions, use the following state box:

If a state transitions to itself, use the following notation in the state box:

- entry/Action
- exit/*Action*

A transition from one state to another is a link arrow connecting the two states and is labeled with: event, guard, action. The "event" is the action that causes a state to transition to another; e.g. "CheckPin." The "guard" is a possible outcome of an event; e.g. "correct PIN" or "incorrect PIN". The "action" is the result of the outcome of an event; e.g. if the guard is "incorrect PIN" then the "action" may be something like "increment error counter." Format your transition labels as follows:

- *Event [guard] / action*

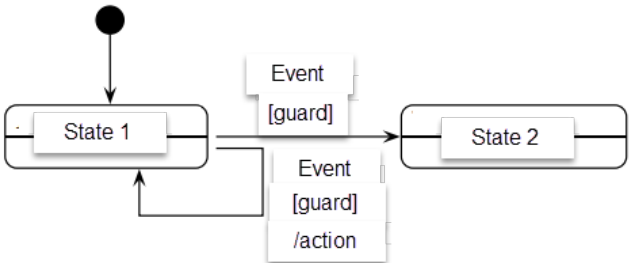


Image Caption:

```
@startuml
[*] --> State1
State1 : event/action
State1 -> State1 : Event [guard]/action
State1 --> State2 : Event [guard]
State2 --> [*]
@enduml
```

Note: Be sure to utilize one of the following UML tools for creating your diagrams for this assignment (links below):

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)

Compile all of your work into a single document and submit as a Word or PDF file.

Part 2. Lessons Learned Reflection

Write a two- to three-page summary that outlines the lessons you have learned in this programming course. Reflect on how these lessons can be applied toward more effective coding.

Your paper must be formatted according to the CSU-Global Guide to Writing and APA and be supported by a minimum of five professional and academic sources. The CSU-Global library is a good place to locate these sources. Review the rubric in the Module 8 Materials folder for specific grading criteria.

Links:

- UMLet (http://www.umlet.com/)
- Gliffy (https://www.gliffy.com/go/html5/launch?app=1b5094b0-6042-11e2-bcfd-0800200c9a66&templateId=4218693_)
- Microsoft Visio (https://www.microsoftstore.com/store/msusa/en_US/cat/Visio/categoryID.69407600)

COURSE POLICIES

Grading Scale	
A	95.0 – 100
A-	90.0 – 94.9
B+	86.7 – 89.9
B	83.3 – 86.6
B-	80.0 – 83.2
C+	75.0 – 79.9
C	70.0 – 74.9
D	60.0 – 69.9
F	59.9 or below

Course Grading

20% Discussion Participation
0% Opening Exercises
8% Mastery Exercises
37% Critical Thinking Assignments
35% Final Portfolio Project

IN-CLASSROOM POLICIES

For information on late work and incomplete grade policies, please refer to our **In-Classroom Student Policies and Guidelines** or the Academic Catalog for comprehensive documentation of CSU-Global institutional policies.

Academic Integrity
Students must assume responsibility for maintaining honesty in all work submitted for credit and in any other work designated by the instructor of the course. Academic dishonesty includes cheating, fabrication, facilitating academic dishonesty, plagiarism, reusing /repurposing your own work (see CSU-Global Guide to Writing & APA for percentage of repurposed work that can be used in an assignment), unauthorized possession of academic materials, and unauthorized collaboration. The CSU-Global Library provides information on how students can avoid plagiarism by understanding what it is and how to use the Library and internet resources.

Citing Sources with APA Style
All students are expected to follow the CSU-Global Guide to Writing & APA when citing in APA (based on the most recent APA style manual) for all assignments. A link to this guide should also be provided within most assignment descriptions in your course.

Disability Services Statement
CSU-Global is committed to providing reasonable accommodations for all persons with disabilities. Any student with a documented disability requesting academic accommodations should contact the Disability Resource Coordinator at 720-279-0650 and/or email ada@CSUGlobal.edu for additional information to coordinate reasonable accommodations for students with documented disabilities.

Netiquette
Respect the diversity of opinions among the instructor and classmates and engage with them in a courteous, respectful, and professional manner. All posts and classroom communication must be conducted in accordance with the student code of conduct. Think before you push the Send button. Did you say just what you meant? How will the person on the other end read the words?

Maintain an environment free of harassment, stalking, threats, abuse, insults, or humiliation toward the instructor and classmates. This includes, but is not limited to, demeaning written or oral comments of an ethnic, religious, age, disability, sexist (or sexual orientation), or racist nature; and the unwanted sexual advances or intimidations by email, or on discussion boards and other postings within or connected to the online classroom.
If you have concerns about something that has been said, please let your instructor know.