Q1. Having quality design goals is of most help in reducing the complexity of:

1. **designing the system.**
2. building the system.
3. maintaining the system.
4. cost and time estimates for developing the system.
5. understanding the system.

Q2. An interface *abstracts* a module. Abstraction helps most in reducing the complexity of:

1. designing the system.
2. building the system.
3. maintaining the system.
4. cost and time estimates for developing the system.
5. **understanding the system.**

Q3. Which of the following is not an issue when considering "programming-in-thelarge"?

1. Understanding user requirements
2. Applying appropriate software development processes
3. Building models of a system
4. Validating user input
5. Making design trade-offs

Q4. Which statement is not true about software engineering?

1. It requires a team effort.
2. It should build a quality system.
3. It should solve a real user problem.
4. It is an ad-hoc development effort.
5. It deals with multiple versions of the software.

Q5. Which statement about the UML (Unified Modeling Language) is true?

1. The UML makes us think about the world in a certain way.
2. The UML can be used to model only software systems.
3. The UML can be used for only object-oriented software systems.
4. The UML is a software development process.
5. The UML provides a fixed set of modeling elements.

Q6. Which of the following UML concepts is not a classifier?

1. class
2. operation
3. association
4. method
5. attribute

Q7. In software engineering, we build models of a software system to:

1. reduce the workload of the project team.
2. help us deal with the complexity of a problem.

3. reduce the amount of communication with users.
4. know which people to hire into the project team.
5. know which language to use for implementation.

Q8. Which of the following is not a property of an attribute in the UML?

1. data type
2. multiplicity
3. visibility
4. signature
5. changeability

Q9. Is it possible for there to be more than one association between any two classes?

1. Yes, always.
2. Yes, but only for binary and higher order associations. 3. Yes, if the multiplicity is many to many (N:M)
4. No, no way.
5. Gee, I don't know!

Q10. In the UML the multiplicity of an association specifies

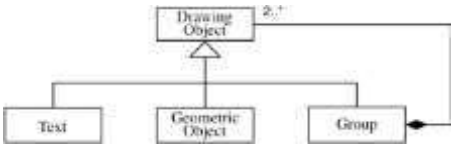
1. which classes can be related to each other.
2. how many classes participate in the association.
3. the navigability of the association.
4. the number of objects that must/can be related.
5. whether role names are required.

Q11. Considering what is true **in the real world**, what is the most likely multiplicity of the IssuedTo association?

1. CreditCard (1..*) -----IssuedTo----- (1..1) Person
2. CreditCard (0..*) -----IssuedTo----- (1..1) Person
3. CreditCard (1..1)-----IssuedTo----- (0..*) Person
4. CreditCard (1..1)-----IssuedTo----- (1..*) Person
5. CreditCard (0..*) -----IssuedTo----- (1..*) Person

Q12. A drawing object can be either text, a geometric object or a group of text and geometric objects. Which kind or kind(s) of relationship(s) are needed to correctly model this situation?

1. association only
2. generalization only
3. composition and generalization only
4. association and generalization only
5. all of association, composition and generalization



Q13. When is an association class needed?

1. When the multiplicity of the relationship is many-to-many (N:M).
2. When the relationship is unidirectional.
3. When the relationship has properties.
4. When the relationship is mandatory.
5. When the relationship is optional.

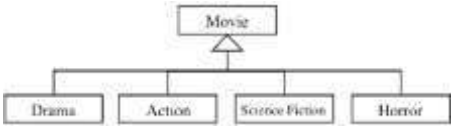
Q14. In the UML, the concept of generalization

1. relates two different classes by an association relationship.
2. allows a class to remove attributes and operations from its subclasses.
3. supports the concept of stereotype.
4. allows a class to specialize its attributes and operations.
5. links instances of different classes together.

Q15. Which of the following is a generalization coverage constraint?

1. complex
2. disunion
3. overloaded
4. common
5. disjoint

Q16. Considering what is true **in the real world**, the generalization shown in the figure is:



1. overlapping and complete.
2. disjoint and complete.
3. overlapping and incomplete.
4. disjoint and incomplete.
5. none of the above.

Q17. Which of the following statements is true about ***custom*** software?

1. The number of copies in use is high.
2. The requirements come from market research.
3. The development effort is high.
4. The requirements come from hardware needs.
5. The development effort is low.

Q18. A milestone in a software development project is

1. a management decision point.
2. a problem that delays the project.
3. the time at which a project starts.
4. the time at which a project completes.
5. a meeting with the client.

Q19. When developing a software development plan, the first task to do is to:

1. decide on the implementation environment.
2. define the scope of the project.
3. identify the deliverables.
4. develop a schedule for the project.
5. identify the project risks.

Q20. Which of the following is not a way to deal with risks in software development?

1. avoid (replan or change requirements)
2. ignore (act as if it won't happen)
3. mitigate (devise tests to see if it occurs)
4. confine (restrict the scope of its effect)
5. monitor (constantly be on the lookout for it)

Q21. Which of the following is not an emphasis of an Agile development process?

1. individuals and interactions
2. comprehensive test plan
3. client involvement/collaboration
4. working software
5. responsiveness to change

Q22. In domain modeling we capture the system's most important

1. user interfaces.
2. functional requirements.
3. use cases and scenarios.
4. classes and associations.
5. acceptance tests.

Q23. What do we capture in use-case modeling?

1. user interface requirements
2. hardware requirements
3. acceptance tests
4. system behaviour
5. data requirements

Q24. "The system should register a student in less than a second" is an example of what type of requirement?

1. functional
2. pseudo
3. data
4. nonfunctional
5. user interface

Q1. Defensive programming means that you

1. do not let anyone else read your code.

2. do not let input data crash your program.
3. do not let anyone else change your code.
4. do not use object-oriented programming languages.
5. do not let anyone else test your code.

Q2. The primary purpose of refactoring code is to

1. reduce the number of input parameters.
2. isolate bugs by rewriting the code.
3. add assertions to the code.
4. improve the internal structure of the code.
5. improve the interaction among components.

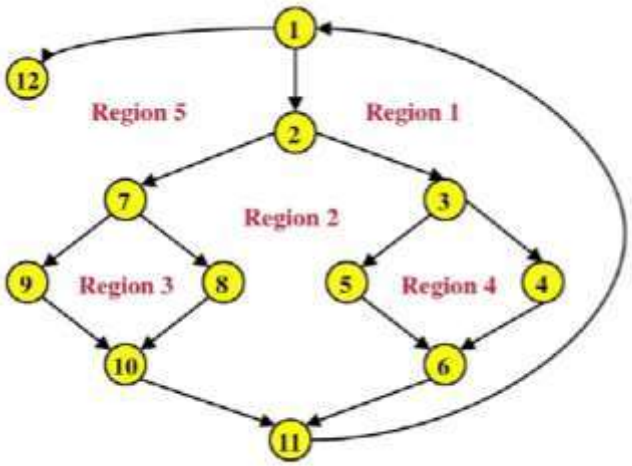
Q3. Basis path testing ensures that we have tested at least

1. all interactions between programs.
2. all the ways of executing the loops in a program.
3. all the statements in a program.
4. all the data structures in a program.
5. all the inputs to a program.

Q4. What is the cyclomatic complexity of the procedure in the figure?

```
    Procedure: process records
1.  Do while records remain
2.    read record
3.    If record field 1 = 0
4.      If record field 2 = 0
5.        store in buffer
6.        increment counter
7.      Else
8.        print record
9.      Endif
10.   Else
11.     If record field 3 = 0

12.       store in file
13.       increment counter
14.     Else
15.       delete record
16.     Endif
17.   Endif
18. Enddo
    End
```



Flow graph node to procedure statement mapping:

Node	Statement
1.	1
2.	2, 3
3.	4
4.	5, 6
5.	7, 8
6.	9
7.	10, 11
8.	12, 13
9.	14, 15
10.	16
11.	17
12.	18

1. 7
2. 6
3. 5
4. 4
5. 3

Q5. When testing a *nested* loop, we initially test the inner loop while holding the outer loop at

6. its minimum value.
7. its middle value.
8. its maximum value.
9. both its minimum and maximum value.
10. any value.

Q6. Black box testing uses test values at the boundaries of a subdomain because

1. these values are easier for us to figure out.
2. this will make integrating components easier.
3. errors are more likely to occur here.
4. black box testing only works for such values.
5. these values can be given to us by the users.

Q7. One of the required inputs for a program that does room scheduling for a public venue is the day of the week, which is input as Sunday, Monday, etc. When testing this program for the input day of the week, what are the **minimum number of test values** that you would use?

1. 5
2. 7
3. 8
4. 9
5. 10

Q8. Black box testing techniques are used in what type of testing?

1. unit
2. condition
3. loop
4. integration
5. data flow

Q9. Test cases for state-based testing can be derived from the

1. use case model.
2. state machine diagrams.
3. basis paths.
4. domain model.
5. nonfunctional requirements.

Q10. The purpose of an acceptance test plan is to

1. verify the acceptability of the code structure.
2. specify the criteria for determining whether the system is finished.
3. list all the requirements for the system.
4. define the scope of the system development.
5. define the goals of the system.

Q11. One purpose of system analysis and design is to

1. determine the cost of developing the system.
2. determine the time required to implement the system.
3. capture the system's nonfunctional requirements.
4. adapt the requirements to the implementation environment.
5. obtain the client's approval for developing the system.

Q12. In the Model-View-Controller (MVC) architectural pattern, the Model represents

1. the objects used to render data in the user interface.
2. the process control mechanism used by the system.
3. the controls with which the user interacts.
4. the data viewed and manipulated by the user.
5. the business logic of the application.

Q13. System analysis and design distributes the functionality of a use case into boundary, entity and control classes because

1. these are the only kinds of classes available to us.
2. these classes are easier to test.
3. we want to isolate specific types of changes to specific types of classes.
4. state machine diagrams use these classes.
5. object interaction is more easily described using these classes.

Q14. According to the best practices of analysis object interaction, which one of the following interactions **should not** be allowed?

1. interactions between a boundary object and a control object
2. interactions between two control objects
3. interactions between a boundary object and an entity object
4. interactions between two boundary objects
5. interactions between a control object and an entity object

Q15. A design class is most cohesive when it

1. is designed by only one person.
2. does not require database access.
3. has few dependencies to other classes.
4. is used in only one scenario of a use case.
5. has responsibilities that are closely related.

Q16. A state machine diagram describes the behaviour of

1. an object.
2. a use case.
3. an operation.
4. an actor.
5. a class.

Q17. A state machine diagram responds to every event that

1. occurs.
2. occurs provided it is not processing an activity.
3. changes a value of the object's attributes.
4. triggers a transition.
5. sends a message.

Q18. Which of the following **is not true** about design patterns?

1. They help novices behave like experts.
2. They are described using program code.
3. They represent a solution to a problem in a context.
4. They can be used to handle nonfunctional requirements.
5. They make extensive use of inheritance and delegation.

Q19. The **Strategy** design pattern is used to

1. defer instantiation of a class to its subclasses.
2. provide a placeholder for another object.
3. decouple an abstraction from its implementation.
4. restrict a class to have only one instance.
5. encapsulate an interchangeable family of algorithms.

Q20. The **Mediator** design pattern is used to

1. provide a unified interface to a subsystem.
2. provide a placeholder for another object.
3. decouple an abstraction from its implementation.
4. restrict a class to have only one instance.
5. encapsulate how a set of objects interact.

Q21. If an organization has a standards handbook for software development, then which of the following probably **is not true**?

1. A project can decide to not follow any standards.
2. A project can decide to ignore a standard.
3. A project can decide to use a standard as is.
4. A project can decide to modify a standard.
5. A project can decide to create a new standard.

Q22. The cyclomatic complexity of a program can tell us something about which of the following external attributes (design goals) of software quality?

1. usability - how easy it is to use
2. learnability - how easy it is to learn how to use
3. safety - how likely it is to be error free
4. integrity - how well it prevents unauthorized access
5. installability - how easy it is to install

Q23. In terms of software quality assurance, the purpose of both the SEI Process Capability Maturity Model (SEI-CMM) and the People Capability Maturity Model (PCMM) is to:

1. educate and train managers.
2. provide feedback on current practices.
3. assess and improve current practices.
4. develop best practices.

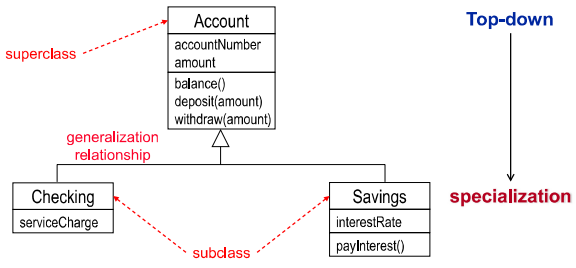
5. improve the standards handbook.

Q24. Which of the following **cannot be shown** on a Gantt chart?

1. the task dependencies
2. the time estimates for each task
3. the resource assignment for each task
4. the tasks that lie on the critical path
5. the progress of each task

GENERALIZATION (cont'd)

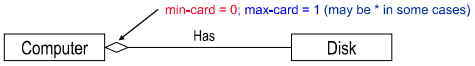
discriminator: An attribute of enumeration type that indicates which property of a class is used to create a generalization relationship.



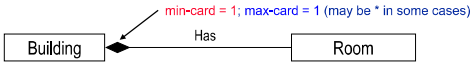
AGGREGATION/COMPOSITION ASSOCIATION

- A special type of association in which there is a "part-of" relationship between one class and another class.

☞ A component may exist independent of the aggregate object of which it is a part → aggregation. [<> adornment]



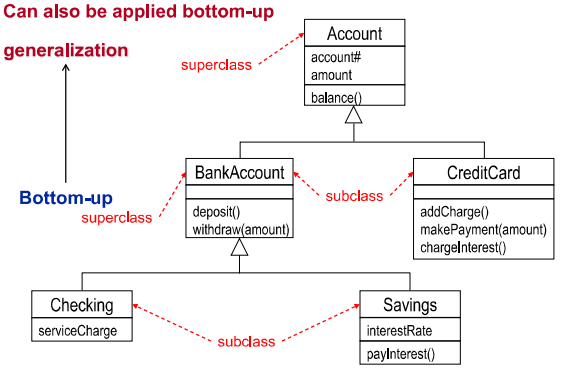
☞ A component may not exist independent of the aggregate object of which it is a part → composition. [◆ adornment]



GENERALIZATION (cont'd)

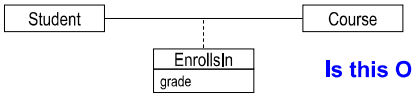
Can also be applied bottom-up

generalization

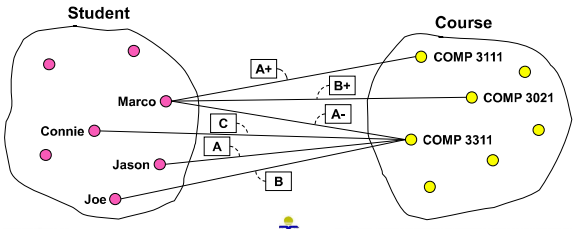


ASSOCIATION: ASSOCIATION CLASS (cont'd)

Option 3: Use an association class.

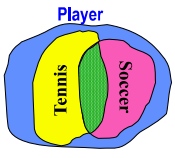


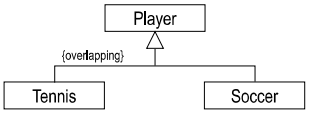
Is this OK?



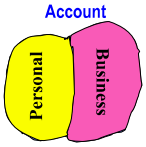
GENERALIZATION: DISJOINTNESS COVERAGE

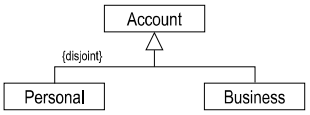
overlapping - A superclass object can be a member of more than one subclass.





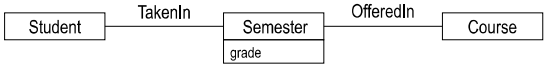
disjoint - A superclass object is a member of at most one subclass.

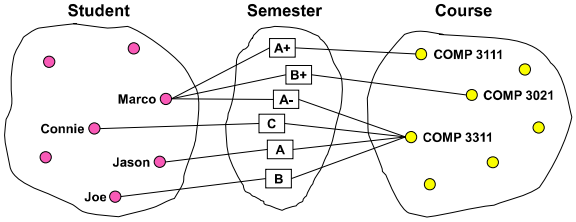




ASSOCIATION: ASSOCIATION CLASS (cont'd)

Option 4: Use a separate class.

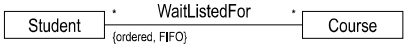




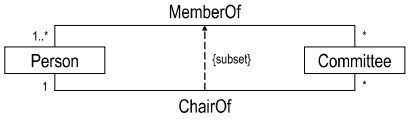
Note that each Semester object *must be related* to a Student and Course.

COMMON ASSOCIATION CONSTRAINTS

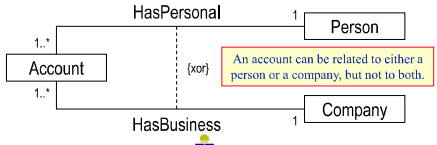
ordering



subset

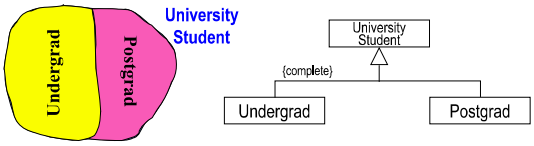


xor

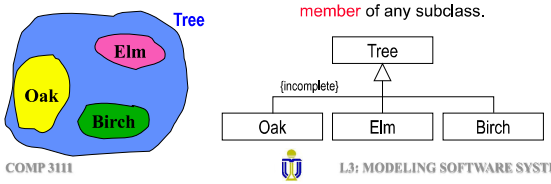


GENERALIZATION: COMPLETENESS COVERAGE

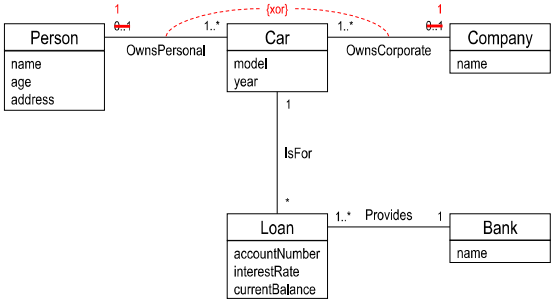
complete - All superclass objects must be members of some subclass.



incomplete - Some superclass object is not a member of any subclass.



EXERCISE: CAR OWNERSHIP AND LOANS — SOLUTION REVISITED

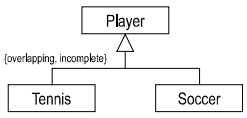


Problem 3
It is still possible for a car to be owned by both a person and a company or to not be owned at all!

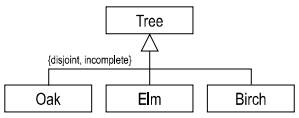
Requires an exclusive or constraint (see later).

GENERALIZATION: COVERAGE TYPES

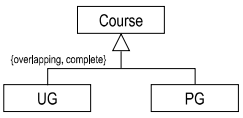
overlapping, incomplete



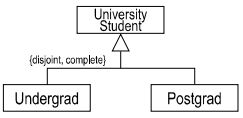
disjoint, incomplete



overlapping, complete

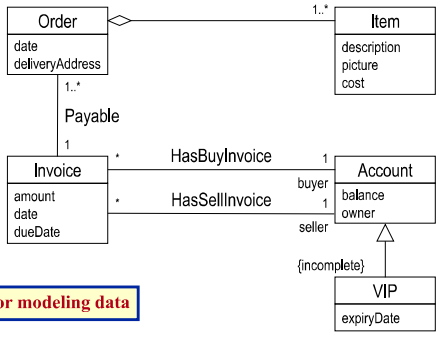


disjoint, complete



MODELING SOFTWARE SYSTEMS USING UML: SUMMARY

Static Modeling Example

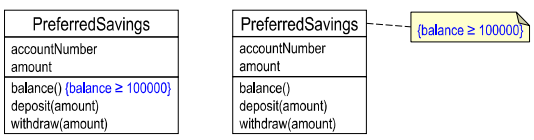


For modeling data

CONSTRAINTS

A *constraint* is an assertion about properties of model elements that must be satisfied by the system.

Example A preferential savings account whose balance always must be greater than $100,000.



A constraint is a statement that can be tested (true/false) and should be enforced by the system implementation.

Name: Casper Kristiansson **Student#:** 20938643 **Date:** 14/9/2022 **Lec. sec.:** L2

COMP 3111: Software Engineering

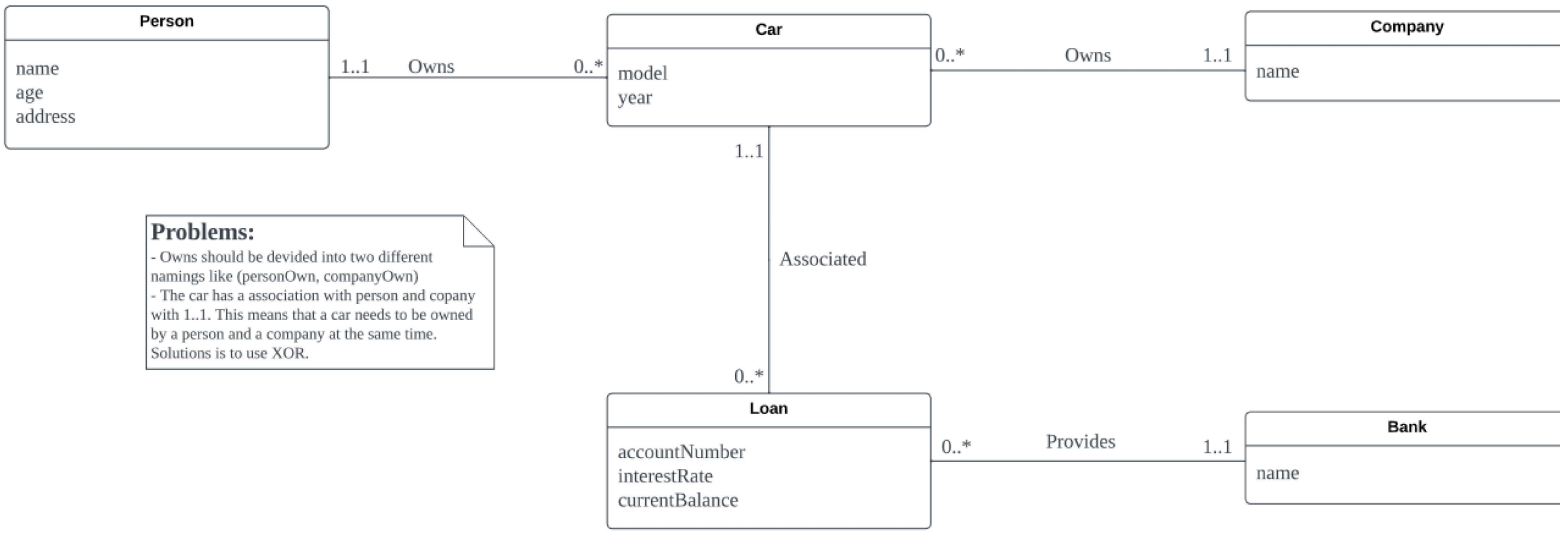
Lecture 2 Exercise: Modeling Software Systems using UML

The classes shown below are required to represent information about car ownership and car loans. However, they have some attributes that are *internal object identifiers (OIDs)*, some of which are being used to represent relationships, and that *should not appear* in a class diagram. All such attributes conveniently have names ending in ID. 🙂

Persons or companies may own cars. The car ownerID is the ID either the person or company that owns the car. A car may have only one owner (person or company). A car may have no loan or multiple loans. A bank provides a loan to a person or a company for the purchase of a car. Only the car owner may obtain a loan on the car. The car owner type and the loan customer type indicate whether the car owner/loan holder is a person or company.

On the class diagram below:

1. **Replace all OIDs with associations.**
2. **Indicate the most likely multiplicities for all associations.**
3. **Cross out any unnecessary attributes.**

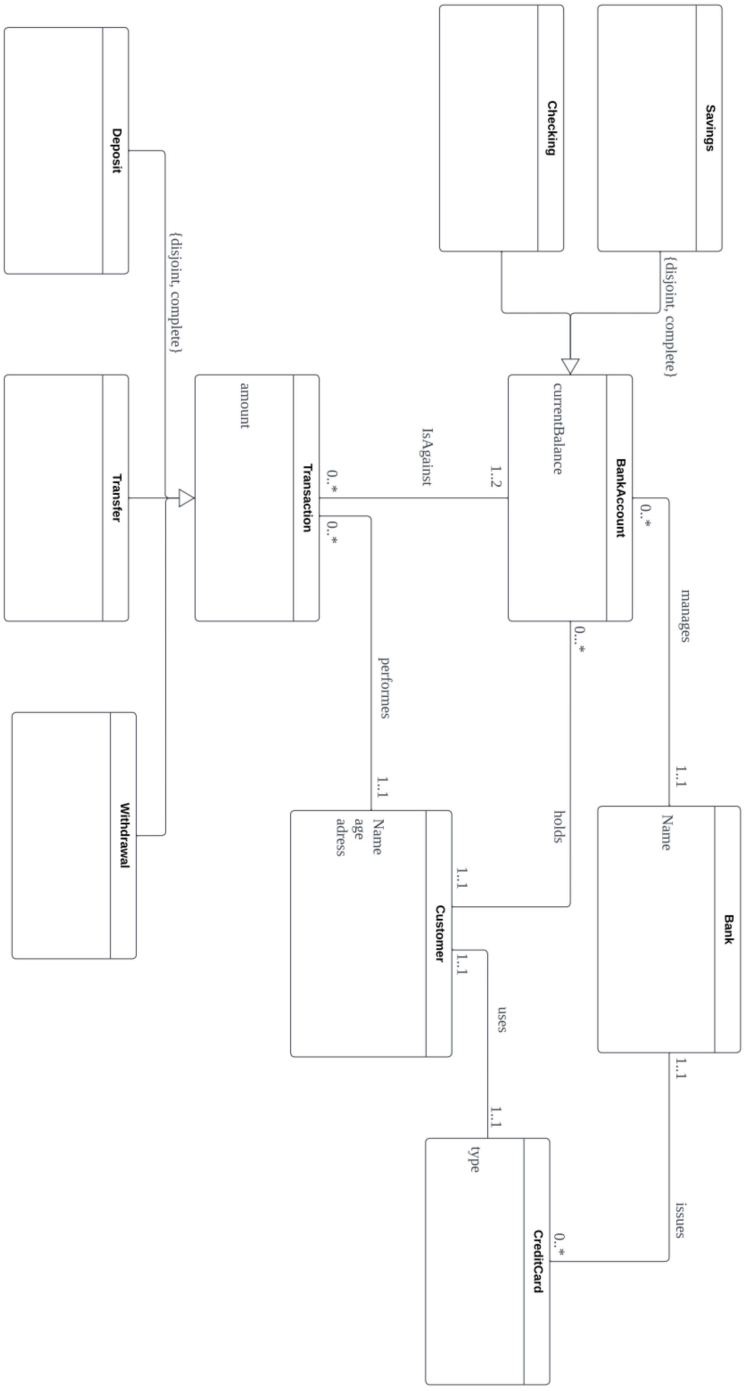


COMP 3111: Software Engineering

Lecture 3 Exercise: Modeling Software Systems using UML

Bank	BankAccount	Checking	CreditCard	Customer
Deposit	Savings	Transaction	Transfer	Withdrawal

Banks issue credit cards (e.g., Visa, MasterCard, etc.) to customers. Each credit card is issued by only one bank to only one customer. Customers hold bank accounts with banks. Each bank account is with only one bank and held by only one customer. A bank account may be a savings account or a checking account. Each savings account has a passbook, which shows the transactions made against the account. Checking accounts do not have passbooks. Customers make transactions against their bank accounts. A transaction can be a deposit, withdrawal or transfer.

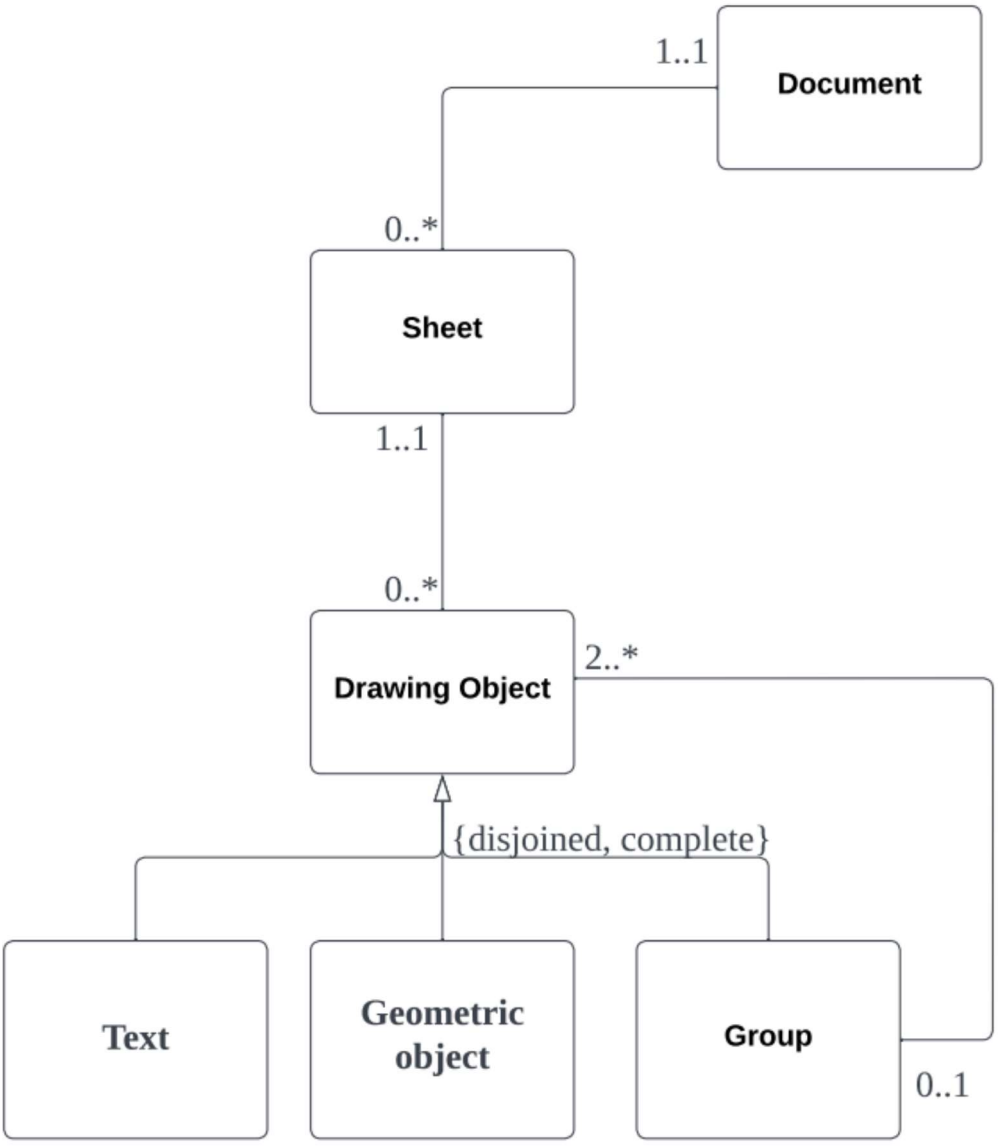


COMP 3111: Software Engineering

Lecture 4 Exercise: Modeling Software Systems using UML

We want to represent information about a graphical editor that supports grouping. A document is composed of several sheets. Each sheet can contain several drawing objects, which can be text, geometric objects or groups. A group is simply a collection of drawing objects, possibly including other groups. A group contains two or more drawing objects. A drawing object can be a member of at most one group. Each sheet belongs to at most one document and each drawing object belongs to exactly one sheet.

In the space below, construct a class diagram for the graphical editor, that shows, as necessary, associations, aggregations, compositions, generalizations among the classes as well as any necessary association classes and constraints. Associations should be named, if necessary. Using the problem statement and real-world knowledge, give the most likely multiplicities for each association, aggregation and composition. If a multiplicity cannot be inferred from the problem statement or real-world domain knowledge, then indicate this with a "?".



At the beginning of each term, students may request a course catalogue containing a list of course offerings needed for the term.
Classes: Student, CourseOffering
associations:
Attributes: term (of Course Offering)
Generalizations:

Information about each course, such as instructor, department, and prerequisites are included to help students make informed decisions.
Classes: Course, Instructor, Department
Associations:
Attributes: prerequisite (of Course)
Generalizations:

The new system will allow students to select four course offerings for the coming term.
Classes: Student
Associations:
Attributes:
Generalizations:

In addition, each student will indicate two alternative choices in case a course offering becomes filled or is canceled.
Classes:
Associations:
Attributes:
Generalizations:

No course offering will have more than forty students or fewer than ten students.
Classes: Course
associations:
Attributes: maxStudents, minStudents
Generalizations:

A course offering with fewer than ten students will be canceled.
Classes: Course
associations:
Attributes: isCanceled
generalizations:

Once the registration process is completed for a student, the registration system sends to the billing system so the student can be billed for the term.
Classes:
associations:
Attributes:
generalizations:

Instructors must be able to access the online system to indicate which courses they will and to see which students signed up for their course offerings.
Classes: Instructor, Student, CourseOffering
Associations:
Attributes:
Generalizations:

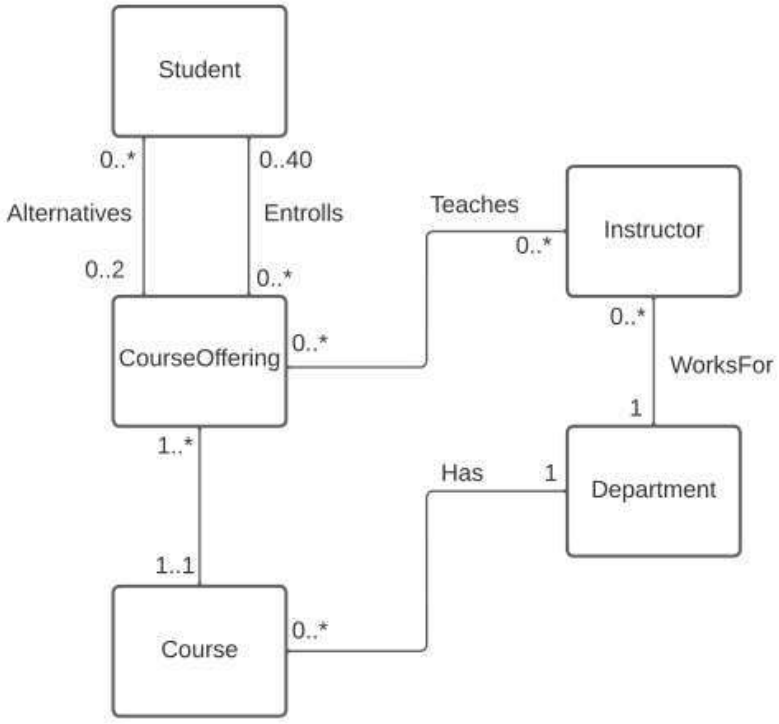
For each term, there is a period of time that students can change their schedule. Studen able to access the system during this time to add or drop course.
Classes: Student, CourseOffering
associations:
Attributes: deadLine (of CourseOffering)
generalizations:

Lecture 6 Exercise: ASU Course Registration—Domain Model

1. On the accompanying worksheet containing the problem statement, identify all the classes, attributes, association classes, associations, generalizations and multiplicity constraints that are relevant to include in the domain model for the new system. *(Only those that are explicitly given in or implied by the requirements statement should be included.)*

2. In the space below construct a class diagram showing how the classes identified in (1) are related by associations, aggregations/ compositions and generalizations. Show the *most likely multiplicities for all associations*, making reasonable assumptions where necessary. If a multiplicity cannot be inferred from the requirements statement or common real-world domain knowledge, then indicate this with a "?".

Do not show the attributes of the classes in the class diagram.



Name: (1) Michell Dib **Student#:** (1) 20938667 **Lec. sec.:** _____

Name: (2) Casper Kristiansson **Student#:** (2) 20938643 **Date:** 02-10-2022

COMP 3111: Software Engineering

Lecture 7 Exercise: Movie Shop—Domain Model

- The system must be able to handle both physical and digital videos.

classes: Movie
associations:
attributes: Movie: Physical, digital
generalizations:

- It must be able to record which videos are sold and rented and by whom.

classes: Customer
associations: Customer purchases video, Customer rents video
attributes: Customer: rentedVideo
generalizations:

- For sold videos, the quantity sold should be recorded; for physical video rental, which copy is rented and when it is due back should be recorded.

classes: RentalCopy
associations: Movie has RentalCopy, Customer rents RentalCopy
attributes:
generalizations:

- The system should keep track of overdue rentals of physical videos and send email notices to customers who have videos overdue.

classes:
associations:
attributes: Customer: email
generalizations: Member is a kind of customer -> Customer generalizes Member

- There will be a customer membership option for an annual fee, which will entitle a member to discounts (10%) on the sale and rental of videos.

classes: Member
associations:
attributes:
generalizations:

- Members should be able to make reservations for physical video rentals either in person at the shop, by telephone or via the Web.

classes: RentalCopy
associations: Movie CanBeAPhysical RentalCopy, Member reserves RentalCopy
attributes:
generalizations:

- A member can reserve at most five physical videos at any one time, but there is no limit on how many physical videos a member or nonmember can rent at any one time.

classes:
associations:
attributes:
generalizations:

- As an added feature, the shop would like to allow customers (either members or nonmembers) to input, via the Web, mini-reviews (up to 100 words) and a rating (from 1, lowest, to 10, highest) of videos they have purchased or rented.

classes: Review

associations: Customer Provides Review, Review isFor Movie

attributes: Review: reviewTest, rating

generalizations:

- These reviews should be anonymous if the customer so wishes (i.e., customers can specify whether they want their name to be made known when other customers browse the reviews).

classes:

associations:

attributes: Review:anonymous

generalizations:

- A sales clerk should be able to enter and update the following information about all customers (members or nonmembers): name, address, phone number, age, sex, and email address.

classes:

associations:

attributes: Customer: name, address, phoneNumber, age, sex, email

generalizations:

- Members are assigned a membership number by the shop when they become members and a password, which allows them to change their personal information and to buy and rent digital videos via the Web.

classes:

associations:

attributes: Member: memberNumber, password

generalizations:

- The shop manager should be able to generate various reports on the sale and rental of videos.

classes:

associations:

attributes:

generalizations:

- A sales clerk should be able to sell and rent physical videos and process the return of rented physical videos.

classes:

associations:

attributes:

generalizations:

- When selling or renting physical videos, a sales clerk must be able to look up customer information and determine whether the customer is a member.

classes:

associations:

attributes:

generalizations:

- A sales clerk must be able to enter basic information about a video (i.e., video id, title, leading actor(s), director, producer, genre, synopsis, release year, running time, selling price, and rental price).

classes:

associations:

attributes:

generalizations:

Name: (1) Michell Dib **Student#:** (1) 20938667 **Lec. sec.:** _____

Name: (2) Casper Kristiansson **Student#:** (2) 20938643 **Date:** 02-10-2022

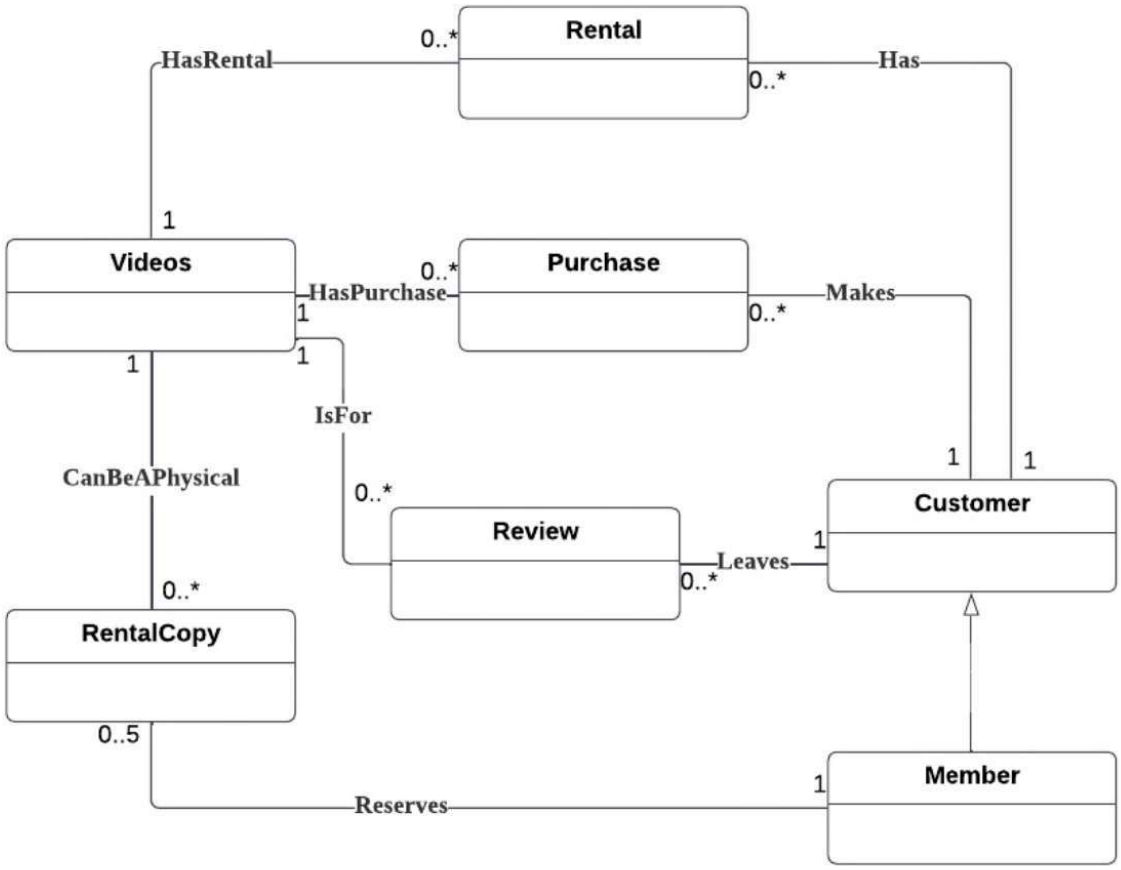
Name: (3) Nicole Wijkman **Student#:** (3) 20938875

Name: (4) Karolina Sjökvist **Student#:** (4) 20938693

COMP 3111: Software Engineering

Lecture 7 Exercise: Movie Shop—Domain Model

In the space below construct the class diagram resulting from your discussion with another group.



ASU USE-CASE MODEL: USE CASES

In addition, each **student** will indicate two alternative choices in case a course offering becomes filled or is canceled.

functionality: Student: ~~*select alternate choices*~~ → *select course offering*
Someone: *cancel course offering*

No course offering will have more than forty **student**s or fewer than ten **student**s.

functionality: None (constraint on functionality)

A course offering with fewer than ten **student**s will be canceled

functionality: No new functionality in this statement

Once the registration process is completed for a **student**, the registration system sends information to the **billing system** so the student can be billed for the term.

functionality: Billing System: *receive billing information*

USE-CASE MODELING: IDENTIFYING ACTORS

- Who or what uses the system?
- What roles do they play in the interaction?
- Who gets and provides information to the system?
- What other systems interact with this system?
- Who installs, starts and shuts down, or maintains the system?

☞ **It is possible to have both a domain model class and an actor that represent the same thing.**

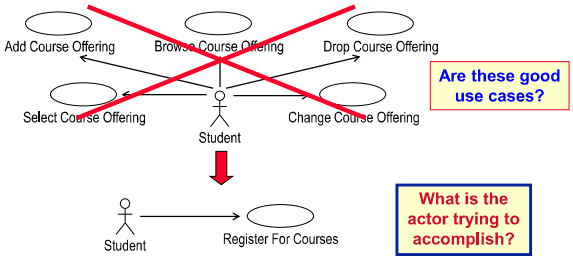
☞ **Input/output devices are NEVER actors!**

For each actor, briefly describe the role it plays when interacting with the system.

WHAT IS A GOOD USE CASE?

A use case typically represents a major piece of functionality that is complete from beginning to end.

A use case must deliver something of value to an actor.



USE-CASE MODELING: IDENTIFYING USE CASES AND SCENARIOS

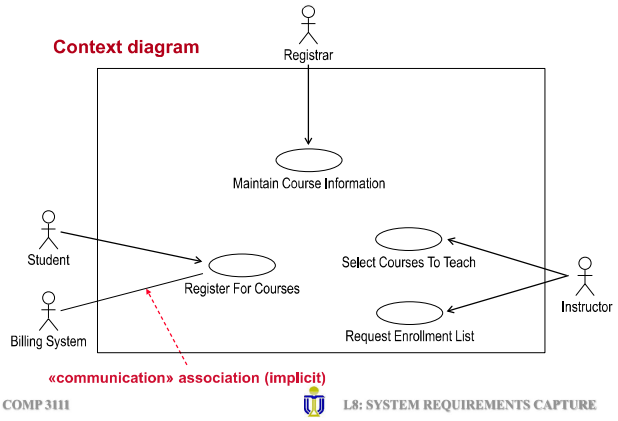
- What are the tasks that an actor wants the system to perform?
- What information does an actor access (create, store, change, remove or read) in the system?
- Which external changes does an actor need to inform the system about?
- Which events does an actor need to be informed about by the system?
- How will the system be supported and maintained?

☞ **State a use case name from the perspective of the actor as a present-tense, verb phrase in active voice.**

☞ **Provide a description of the purpose of the use case and an outline of its functionality.**

☞ **Use application domain terms in descriptions (i.e., from the glossary/data dictionary).**

ASU USE-CASE MODEL: USE-CASE DIAGRAM



ASU USE-CASE MODEL: USE CASES

At the beginning of each term, **student**s may request a course catalogue containing a list of course offerings needed for the term.

functionality: Student: ~~*request course catalogue*~~ → *browse course offering*
Someone: ~~*prepare course catalogue*~~ → *prepare course offering*

Information about each course, such as **instructor**, department, and prerequisites are included to help **student**s make informed decisions.

functionality: Part of *prepare course offering* functionality.

The new system will allow **student**s to select four course offerings for the coming term.

functionality: Student: *select course offering*

• The system must be able to handle both physical and digital videos.
Actors:
Functionality:

• It must be able to record which videos are sold and rented and by whom.
Actors: Customer
Functionality: Customer: buys video, rents video

• For sold videos, the quantity sold should be recorded; for physical video rental, which copy is rented and when it is due back should be recorded.
Actor:
Functionality:

• The system should keep track of overdue rentals of physical videos and send email notices to customers who have videos overdue.
Actor:
Functionality: Customer receives an overdue notice

• There will be a customer membership option for an annual fee, which will entitle a member to discounts (10%) on the sale and rental of videos.
Actor: Member
Functionality: Member get discount

• Members should be able to make reservations for physical video rentals either in person at the shop, by telephone or via the Web.
Actor:
Functionality: Member reserves videos, Sales clerk reserve video

• A member can reserve at most five physical videos at any one time, but there is no limit on how many physical videos a member or nonmember can rent at any one time.
Actor:
Functionality:

• As an added feature, the shop would like to allow customers (either members or nonmembers) to input, via the Web, mini-reviews (up to 100 words) and a rating (from 1, lowest, to 10, highest) of videos they have purchased or rented.
Actor:
Functionality: Customer writes a review, Member writes a review

• These reviews should be anonymous if the customer so wishes (i.e., customers can specify whether they want their name to be made known when other customers browse the reviews).
Actor:
Functionality: Customer browse reviews, Member browse reviews

• A sales clerk should be able to enter and update the following information about all customers (members or nonmembers): name, address, phone number, age, sex, and email address.

Actor: Sales Clerk
Functionality: Sales clerk: enters information about customers, Update customer information

• Members are assigned a membership number by the shop when they become members and a password, which allows them to change their personal information and to buy and rent digital videos via the Web. actors & functionality:
Actor:
Functionality: Member update personal information, Member: Buy video, Member: Rent video

• The shop manager should be able to generate various reports on the sale and rental of videos. actors & functionality:
Actor: Shop manager
Functionality: Shop manager: generates reports on sales and rentals

• A sales clerk should be able to sell and rent physical videos and process the return of rented physical videos. actors & functionality:
Actor:
Functionality: Sales clerk: sells, and rents physical videos, Sales clerk: processes return of rented physical videos

• When selling or renting physical videos, a sales clerk must be able to look up customer information and determine whether the customer is a member. actors & functionality:
Actor:
Functionality: Sales clerk: looks up customer information

• A sales clerk must be able to enter basic information about a video (i.e., video id, title, leading actor(s), director, producer, genre, synopsis, release year, running time, selling price, and rental price).
Actor:
Functionality: Sales clerk: enters information about video

Name: (1) Karolina Sjoekvist **Student#:** (1) 20938693 **Lec. sec.:** _____

Name: (2) Casper Kristiansson **Student#:** (2) 20938643 **Date:** 07/10/2022

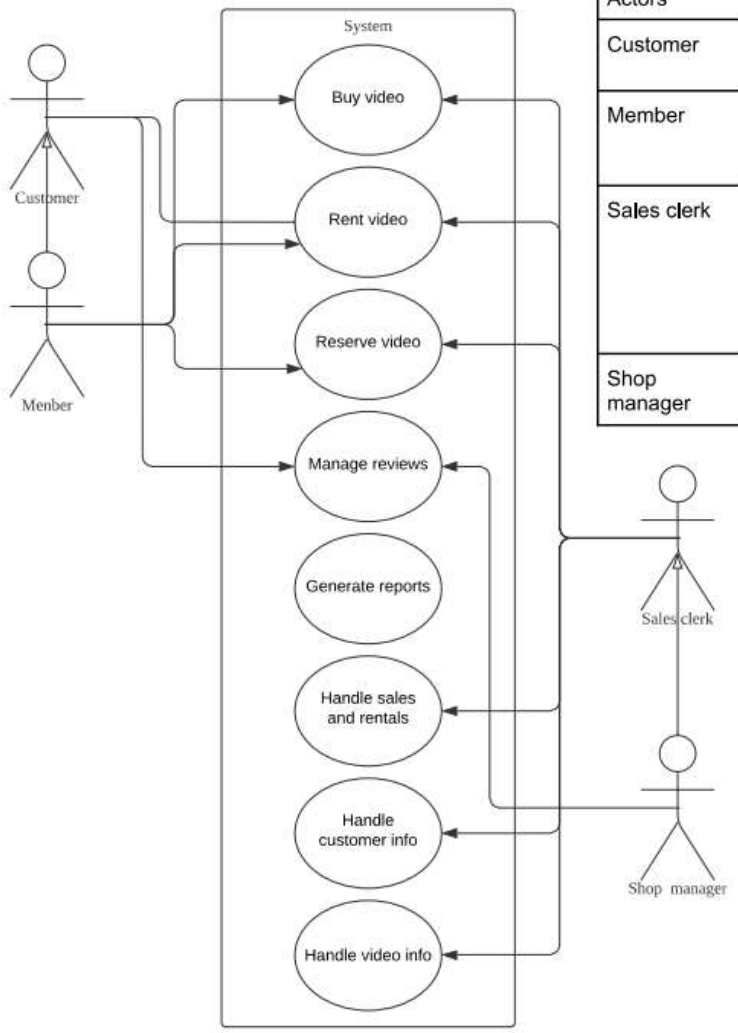
Name: (3) Michell Dib **Student#:** (3) 20938667

Name: (4) Nicole Wijkman **Student#:** (4) 20938875

COMP 3111: Software Engineering

Lecture 8 Exercise: Movie Shop—Use-Case Model Context Diagram

In the space below construct the use-case context diagram resulting from your discussion with another group.



Actors	Required Functionality	Use cases
Customer	Buys video, Rents video, Writes reviews, Browse reviews	Buys video, Rents video, Write reviews
Member	Buys video, Rents video, Reserves video, Get discount, Manages their personal information	Buys video, Rents video, Write reviews, Reserve video
Sales clerk	Enters customer information, Updates customer information, Sells and rents physical videos, Process returns, Look up customer information, Enters information about video	Handle sales and rentals, Handle customer information, Handle video information
Shop manager	Generates reports about sales and rentals	Generate reports

Name: (1) Karolina Sjoekvist **Student#:** (1) 20938693 **Lec. sec.:** _____

Name: (2) Casper Kristiansson **Student#:** (2) 20938643 **Date:** 07/10/2022

COMP 3111: Software Engineering

Lecture 9 Exercise: Citygas—Use-Case Model

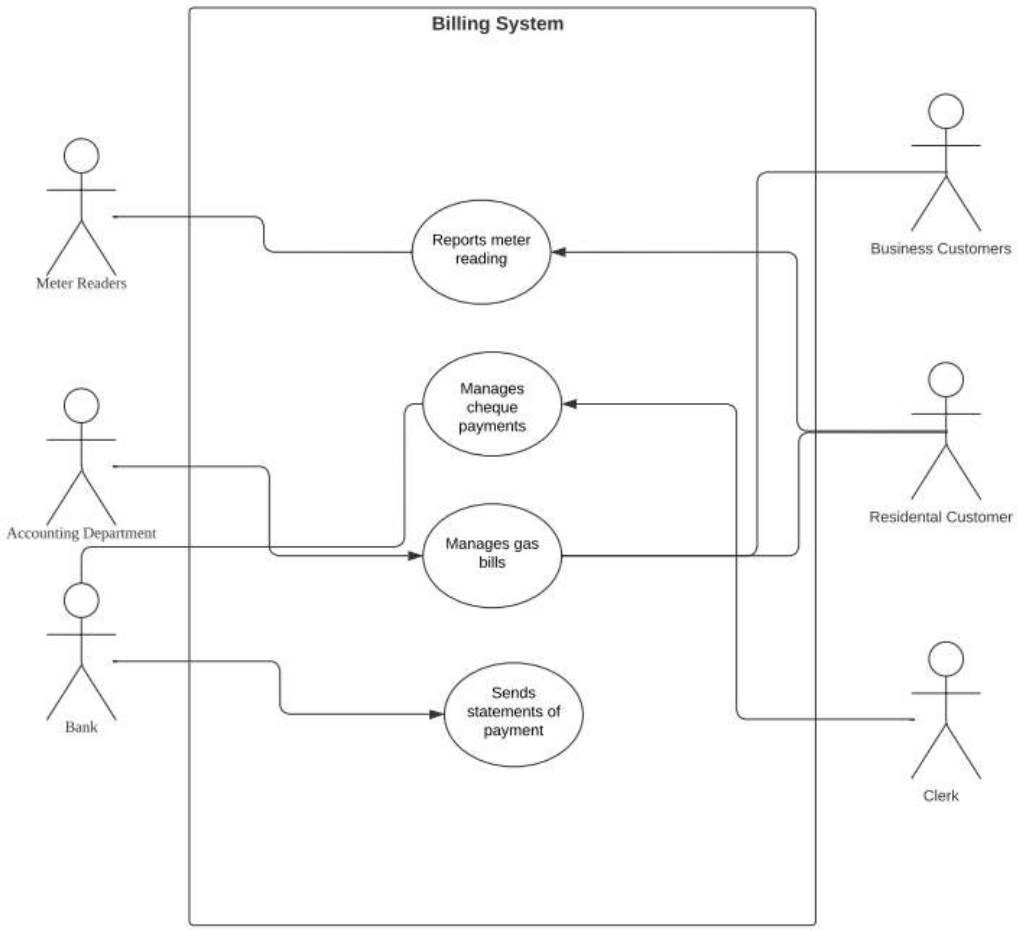
Citygas is designing a new billing system. For business customers, meter readers read the meter monthly. For residential customers, some require a meter reader to read the meter monthly, while others enter their meter reading directly into the system over the telephone. The meter readers enter the meter readings into handheld devices that upload the meter reading wirelessly to the billing system.

The meter readings are used to prepare a bill that is sent to the customer. Payments can be made by cheque or by autopay. For a cheque payment, a clerk records the payments in the system and then the payment is deposited in the bank. For autopay payments, the banks send a daily electronic statement of payments received to the billing system.

Special meter readings are required when a customer sells a building or a flat and turns over possession to the new owner. These have to be calculated and billed separately to reflect a bill for part of a month. A special reading is also provided when a meter is repaired or replaced. A separate bill is not prepared in these cases.

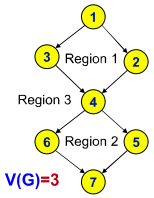
The accounting department is provided with a daily summary of payments received and a weekly list of bills unpaid for ninety days. The accounting department tries to collect unpaid bills. If they are not successful, they cut off service and send the billing department a notice to terminate the account.

From the above requirements statement, construct a context use-case diagram that shows all required actors and their associated uses cases.



BASIS PATH TESTING: INDEPENDENT PATHS

```
Procedure: example()
1.  If c1
2.     f1()
3.  Else
4.     f2()
5.  Endif
6.  If c2
7.     f3()
8.  Else
9.     f4()
10. Endif
11. End
```

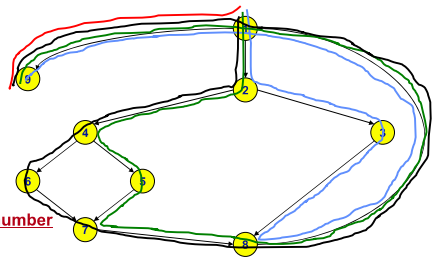


How many independent paths are there in the basis set?

Recall: An independent path introduces at least one new set of statements or a new condition (i.e., it traverses at least one new edge).

☞ V(G) is just an **upper bound** on the number of independent paths.

EXAMPLE BASIS PATH TESTING: INDEPENDENT PATHS



Statement number

1-14

1-2-3-4-5-12-13-14

1-2-3-6-7-8-11-12-13-14

1-2-3-6-7-9-10-11-12-13-14

WHITE BOX TESTING: BASIS PATH TESTING (CONT'D)

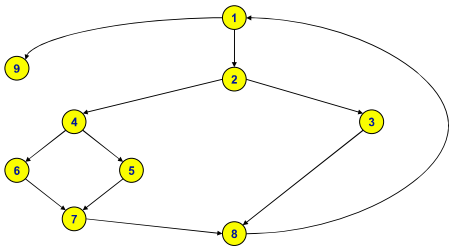
4. Prepare test cases that force the execution of each path in the basis set.

Notes:

- The basis set refers to the statement numbers in the program.
- Count each logical test—compound tests count as the number of Boolean operators + 1 (i.e., count each simple predicate).
- Basis path testing should be applied to all components, if possible, and to critical components **always.**

Basis path testing *does not* test all possible combinations of all paths through the code; *it just tests every path at least once.*

EXAMPLE BASIS PATH TESTING: FLOW GRAPH



Flow graph node to program statement number mapping:

Node	Statement	Node	Statement	Node	Statement
1.	1	4.	6, 7	7.	11
2.	2, 3	5.	8	8.	12, 13
3.	4, 5	6.	9, 10	9.	14

WHITE BOX TESTING: BASIS PATH TESTING (CONT'D)

2. Determine the cyclomatic complexity of the flow graph.

cyclomatic complexity V(G): A quantitative measure of the logical complexity of the code.

Cyclomatic complexity provides an upper bound on the number of paths that need to be tested in the code.

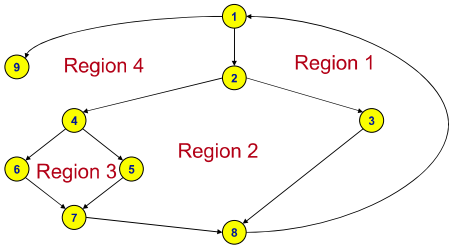
Ways to compute cyclomatic complexity V(G):

☞ V(G) = the number of regions (areas bounded by nodes and edges—area outside the graph is also a region)

☞ V(G) = the number of edges - the number of nodes + 2

☞ V(G) = the number of (simple) predicate nodes + 1

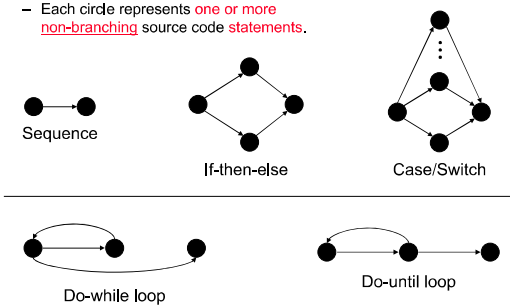
EXAMPLE BASIS PATH TESTING: CYCLOMATIC COMPLEXITY



WHITE BOX TESTING: BASIS PATH TESTING (CONT'D)

1. From the code, draw a corresponding flow graph.

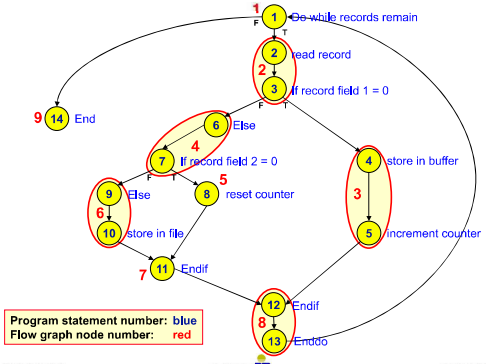
- Each circle represents one or more non-branching source code statements.



EXAMPLE BASIS PATH TESTING: FLOW GRAPH

```
Procedure: process records
1.     Do while records remain
2.        read record
3.        If record field 1 = 0
4.           store in buffer
5.           increment counter
6.        Else
7.           If record field 2 = 0
8.              reset counter
9.           Else
10.             store in file
11.          Endif
12.       Endif
13.    Enddo
14. End
```


EXAMPLE BASIS PATH TESTING: FLOW GRAPH



Name: Casper Kristiansson **Student#:** 20938643 **Lec. sec.:** L12

Date: 2022-10-19

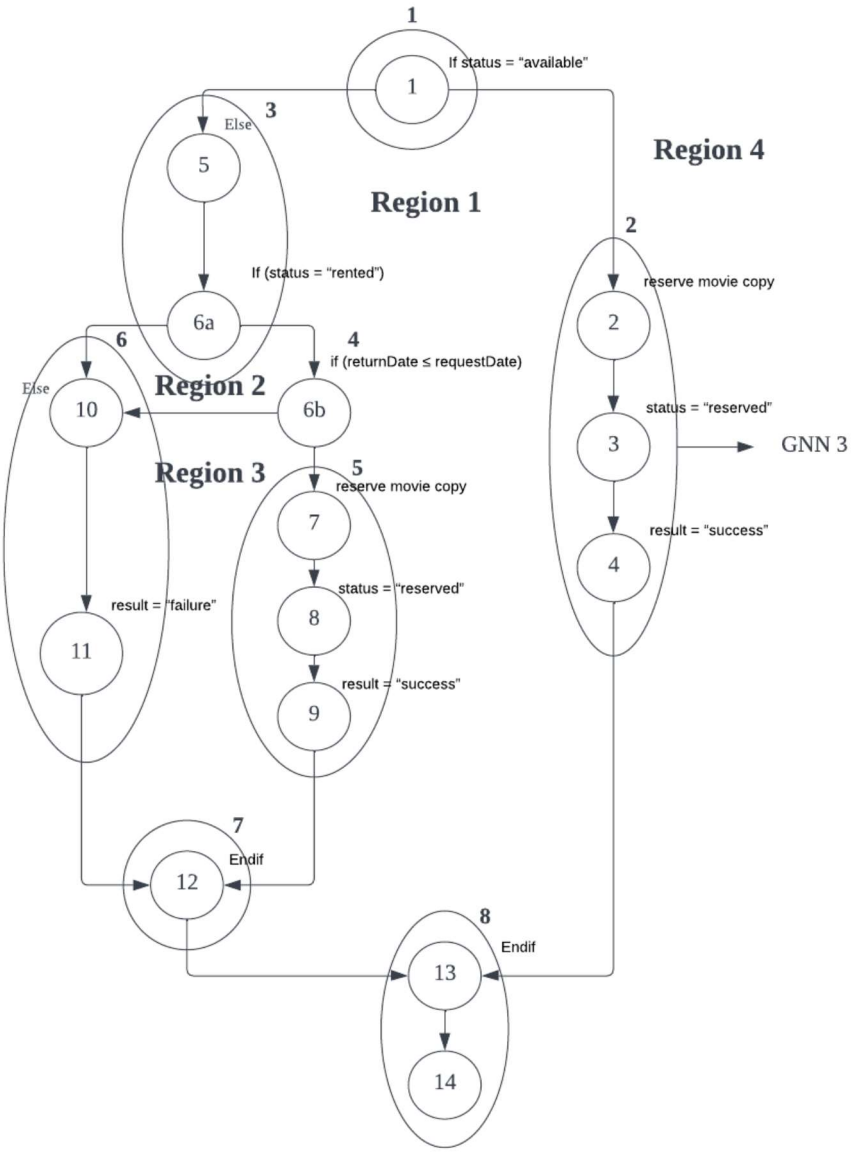
COMP 3111: Software Engineering

Lecture 12 Exercise: Movie Shop Basis Path Testing—Flow Graph

The reserveMovieCopy procedure is used for reserving a physical copy of a movie in the shop.

```
     Procedure: reserveMovieCopy return(result)
1.       If status = “available”
2.           reserve movie copy
3.           status = “reserved”
4.           result = “success”
5.       Else
6.           If (status = “rented”) AND (returnDate ≤ requestDate)
7.               reserve movie copy
8.               status = “reserved”
9.               result = “success”
10.          Else
11.              result = “failure”
12.          Endif
13.      Endif
14.  End
```

Draw the flow graph for the reserveMovieCopy procedure. Clearly label the regions of the flow graph. Show the flow graph node number to program statement number mapping.



STATE MACHINE DIAGRAM: SPECIAL STATES

● initial state (start state)

☞ Each diagram must have at most one initial state.

◉ final state (end state)

☞ Each diagram can have multiple final states.

☞ No initial or final state indicates looping behaviour.

- States may be named. InCredit
- States may be unnamed (called anonymous states) ☐

STATE MACHINE DIAGRAM

A *state machine diagram* describes the behavior inside an object.
(What an object does when it receives a message.)

- It is a directed graph that shows:
  - the states of a single object (nodes).
  - the events that cause state changes (arcs).
- It shows all the messages that an object can send and receive.
- It describes all the possible states an object can get into during its life time.
- It is drawn for a single class to show the lifetime behavior of a single object.

STATE MACHINE DIAGRAM: EVENT

An *event* is something that happens at an instantaneous point in time.

Event types

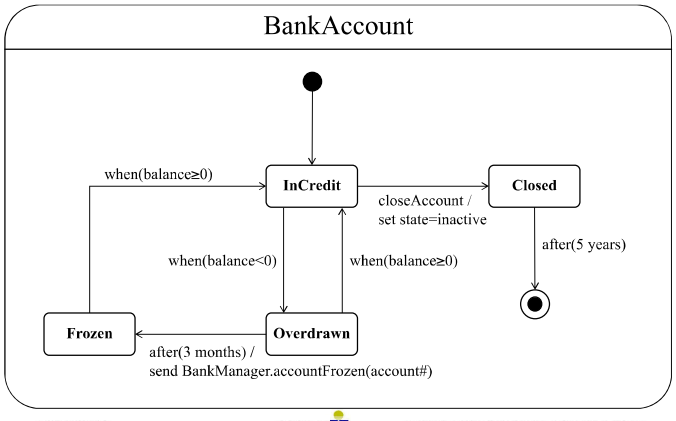
call event – the receipt of a *synchronous call* from an object (a request that an operation be performed)

change event – a specified Boolean condition becoming true
→ when(balance < 0)

time event – absolute time → when(date=07/03/2009)
– elapsed time → after(10 seconds)

signal event – the receipt of an *asynchronous communication* from an object

EXAMPLE STATE MACHINE DIAGRAM

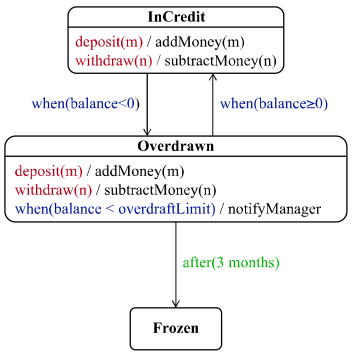


STATE MACHINE DIAGRAM: EVENT TYPE EXAMPLES

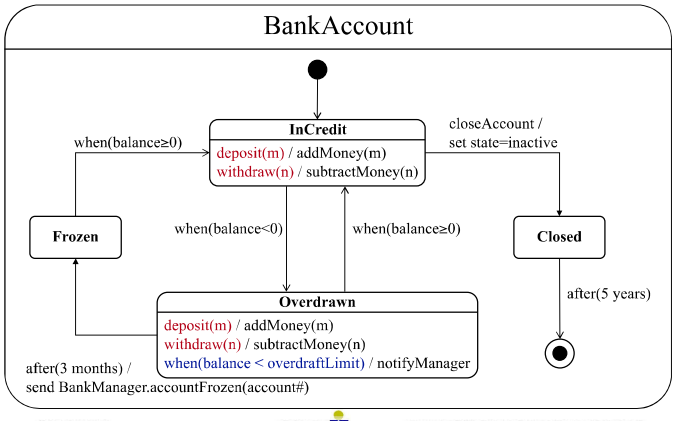
Call event

Change event

Time event



EXAMPLE STATE MACHINE DIAGRAM



Name: Casper Kristiansson **Student#:** 20938643 **Lec. sec.:** L16

Date: 2022-11-16

COMP 3111: Software Engineering

Lecture 16 Exercise: Movie Shop—State Machine Diagram

Construct a state machine diagram showing the states that an instance of the RentalCopy class can be in *with respect to its rental status*. Show only the states, transitions and the events and/or conditions, if any, that cause a transition to be taken. *Do not show the activities that can occur within a state.*

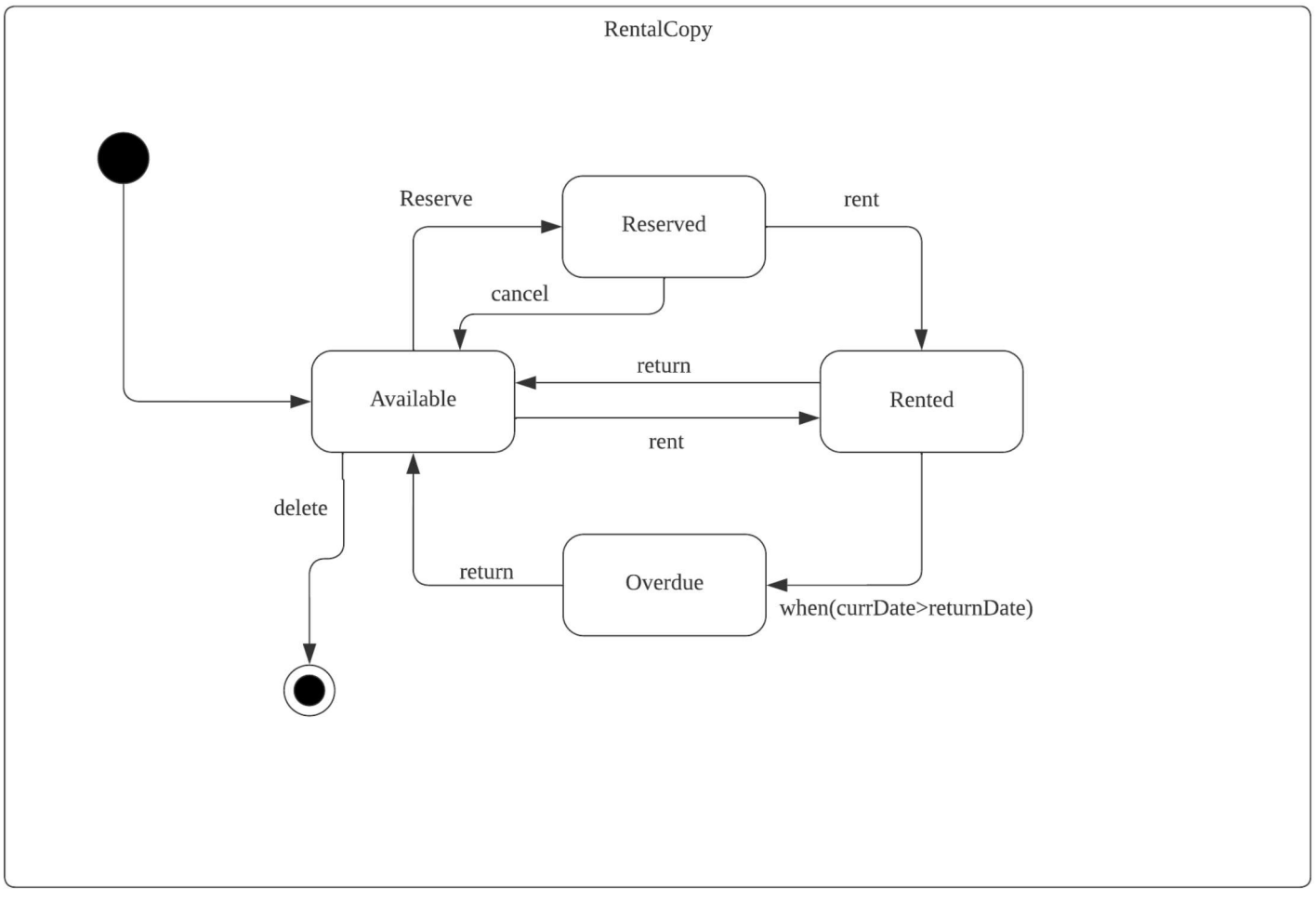
Problem statement requirements that could be relevant to determining the states of a RentalCopy object:

- It must be able to record which movies are sold and rented and by whom.
- For sold movies, the quantity sold should be recorded; for physical movie rental, which copy is rented and when it is due back should be recorded.
- The system should keep track of overdue rentals of physical movies and send email notices to customers who have movies overdue.
- Members should be able to make reservations for physical movie rentals either in person at the shop, by telephone or via the Web.
- A member can reserve at most five physical movies at any one time, but there is no limit on how many physical movies a member or nonmember can rent at any one time.

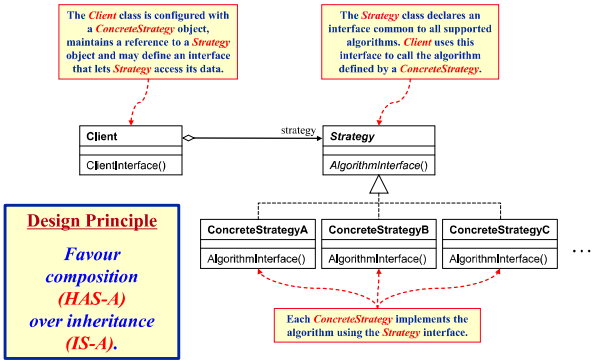
physical movies.

L16 Exericse - Casper Kristiansson

casperkr | November 16, 2022



STRATEGY PATTERN: CLASS DIAGRAM



Design Principle

Favour composition (HAS-A) over inheritance (IS-A).

THE OPEN-CLOSED PRINCIPLE

Design Principle

Classes should be open for extension, but closed for modification.

We should allow a design to incorporate new behaviour without modifying existing code.

How best to achieve it?

Design Principle

Identify the aspects of your application that vary and separate them from what stays the same.

DESIGN PRINCIPLE

Strive for loosely coupled designs between objects that interact.

Loosely coupled designs minimize the interdependency between objects.

Design Principle

Identify the aspects of your application that vary and separate them from what stays the same.

Take what varies and "encapsulate" it so it won't affect the rest of your code.

The result?

Fewer unintended consequences from code changes and more flexibility in your systems.

Design Principle

Program to an interface, not an implementation.

That way, the Duck classes won't need to know any of the implementation details of their behaviours.

Name: Casper Kristiansson **Student#:** 20938643 **Lec. sec.:** L17

Date: 2022-11-16

COMP 3111: Software Engineering

Lecture 17 Exercises: Design Patterns

Exercise 1: Which of the following are disadvantages of using inheritance to provide Duck behaviour (Choose all the apply.)?

- ☐ A. Code is duplicated across subclasses.
- ☒ B. Runtime behaviour changes are difficult.
- ☐ C. We can't make ducks dance.
- ☒ D. Hard to gain knowledge of all duck behaviours.
- ☐ E. Ducks can't fly and quack at the same time.
- ☒ F. Changes can unintentionally affect other ducks.

Exercise 2: How does the observer pattern use the following principles?

Design Principle: Identify the aspects of your application that vary and separate them from what stays the same.

Encapsulate different classes so that the actual instances of a specific class need to change all of the instances won't change.

Design Principle: Program to an interface, not an implementation.

That way the duck classes won't need to know any of the implementations of others. Meaning that you shouldn't hardcode which makes it more difficult to change during runtime. Meaning that you want to make it more flexible. For example, not hardcoding a class for a dog but rather have it inherited from an animal class.

Design Principle: Favour composition over inheritance.

By using composition over inheritance because the design becomes a lot more flexible which in practice will make the code more reusable.

Name: Casper Kristiansson Student#: 20938643 Lec. sec.: L18

Date: 2022-11-16

COMP 3111: Software Engineering

Lecture 18 Exercises: Design Patterns

Exercise 1: An object is loosely coupled if it is dependent on only a few other objects and/or the dependencies are abstract. In the Observer design pattern, the subject may have hundreds of observers (dependencies). In light of this, explain why the Observer design pattern is said to be loosely coupled.

Exercise 2: Consider the code for the Singleton design pattern given below. Modify this code so that at most 5 instances are created and each instance can be individually referenced. Note that the keyword static indicates a class attribute/operation.

```
public class Singleton {
    private static Singleton instance = null;

    private Singleton() { }

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```

Question 1

The observer pattern is loosely coupled because it allows less coupling between different subjects.

Question 2

```
public class Singleton {
    private static Singleton instance = null;
    private []Singleton instances = [];

    private Singleton() { }

    public static Singleton getInstance(index) {
        if (instances[index] == null) {
            instances[index] = new Singleton();
        } else if (index > 5) {
            return null;
        }

      return instances[index];
    }
}
```