Case requirements

- The requirements for the first iteration of the NextGen POS:
  - Implement a basic, key scenario of the Process Sale use case.
  - Implement a Start Up use case as necessary to support the initialization needs of the iteration.
  - Nothing fancy or complex is handled.
  - No collaboration with external services.
  - No complex pricing rules are applied.

Incremental Development

- Incremental development for the same use case across iteration.
  - A use case or feature is often too complex to complete in one short iteration. Therefore, different parts or functions must be allocated to different iterations.
What is a domain model

- A domain model is a visual representation of conceptual classes or real-situation objects in a domain.
- Various names: conceptual models, domain object models, analysis object models
- A domain model is illustrated with a set of CLASS DIAGRAMS in which no operations (method signatures) are defined
- Input
  - Problem Description, Use Cases, ...
- Output
  - A set of class diagrams

Example

Identifying a rich set of conceptual classes is at the heart of OO analysis

Domain models is a visual dictionary

- Domain model provides a conceptual perspective
  - Domain objects or conceptual classes
  - Associations between conceptual classes
  - Attributes of conceptual classes
- The information it illustrates could alternatively have been expressed in plain text
Object and Class concepts

- Object is a thing with **unique identity** in a problem domain
  - **Joe Smith, the University of Sydney, Australia** are all objects
  - All objects have identity and are distinguishable
    - Two apples with the same color, shape, and texture are still individual apples
    - Objects are distinguished by their inherent existence and not by descriptive properties that they may have

All objects have identity and are distinguishable.

Domain objects are the focus of modelling.

**Classes**

- A class describes a group of objects with the same properties, behaviour and possible relationships.
  - An object is an instance of a class
  - **Person**, **University** and **Country** are classes
- Why bother with conceptual class?
  - The power of abstraction
  - The level of abstraction is a matter of judgement and is related with the application.
- Client's description of a future system may have a combination of classes and objects

Object and class diagram

```
Class: Person
  - name : string
  - dob : Date

Person
  - name : string = Joe Smith
  - dob : Date = Oct 21 1983

Person
  - name : string = Mary Sharp
  - dob : Date = March 16 1980
```

Objects with Values

Class with Attributes
Class and conceptual class

- Domain model is a visualization of things in a real-world domain of interest.
- Domain model should not show software classes.

Domain model vs. Data model

- Data model – persistent data to be stored somewhere.
- Domain model also include:
  - Temporary object
  - Object with no attribute

Motivation

- Help to understand the key concepts in a business or problem domain.
- Lower representational gap with OO Modeling.
How to create a Domain Model

Steps
- Find the conceptual classes
- Draw them as classes in a UML class diagram
- Add associations and attributes

Find conceptual classes

Three strategies to find conceptual classes
- Reuse or modify existing models
  - There are published, well-crafted domain models and data models for common domains: inventory, finance, health...
  - Books: *Analysis patterns* by Martin Fowler, *Data Model Patterns* by David Hay, *Data Model Resource Book* by Len Silverston
- Use a category list
- Identify noun phrases

Use a category list

Use a conceptual class category list:
- **Physical or tangible objects**
  - Register, Airplane
- **Specifications, descriptions of things**
  - ProductSpecification, FlightDescription
- **Places**
  - Store, Airport
- **Transactions**
  - Sale, Payment, Reservation
- **etc.** (see Table 9.1 for a fuller list)
Noun Phrase Identification

- Noun Phrase Identification [Abbot 83]
  - Analyze textual description of the domain
  - Identify nouns and noun phrases (indicate candidate classes, objects or attributes)
- Caveats:
  - Automatic mapping isn't possible
  - Textual descriptions are ambiguous!
    (different words may refer to the same class, people may refer to things at different abstract levels: class, object, or somewhere in between)

Example: a typical description

An institution may issue many credit card accounts, each identified by an account number. Each account serves one or more customers who reside at the mailing address. The institution periodically issues a statement for each account. The statement lists a payment due date, finance charge, and minimum payment. The statement itemizes various transactions that have occurred through the billing interval: cash advances, interest charges, purchases, fees and adjustments to the account. The name of the merchant is associated for each purchase.

Mapping parts of speech to object model components

<table>
<thead>
<tr>
<th>MailingAddress</th>
<th>Customer</th>
<th>AccountNumber</th>
<th>CreditCardAccount</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Institute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Statement</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CashAdvance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Interest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Purchase</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fee</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Adjustment</td>
</tr>
</tbody>
</table>
Example: find and draw conceptual classes

- Case study: POS Domain
- Input:
  - fully dressed use case model (excellent textual description of the domain)
- Iteration 1: success (cash-only) scenario of Process Sale use case.
- Strategy: category list & noun phrase identification
- There is no such thing as a "correct" list. It is a somewhat ARBITRARY collection of domain vocabulary that the modelers consider NOTEWORTHY.

POS Domain model: Input

Initial POS domain model
Agile modeling style

- It is normal to miss significant conceptual classes during early domain modeling and to discover them later during DESIGN sketching or PROGRAMMING.
- We can maintain the model by drawing it with a software engineering tool
- OFTEN, a long-life OO analysis domain model does not add value

Include report objects?

- Receipt is a noteworthy term in POS domain
- It has duplicate information contained in sale and payment.
- Guidelines
  - In general, showing a report of other information in a domain model is not useful
  - However, it might have special role in terms of business rules. We might have a reason to show it in the model.

Associations

- Definition
  - An association is a relationship between objects that indicates some meaningful and interesting connection
  - Objects are usually "linked" by a (or a number of) physical or conceptual connections.
  - Cashier "Joe Smith" works on Register "12".
  - Flight "QF435" flies from "Sydney" to "Melbourne"
When to show association

- Associations for which knowledge of the relationship needs to be preserved for some duration
  - In the POS domain
    - We need to remember what SalesLineItem instances are associated with a Sale.
    - Cashier may look up ProductionDescription during the process of sale, but we do not need to remember who looked up what after the sales has been made.
  - We should avoid showing too many associations

Associations and implementation

- There is no direct relationship between associations and implementation.
- Associations do not indicate a certain implementation construct.
- Many associations may be implemented in software differently

Association Notation

- "reading direction arrow" has no meaning except to indicate direction of reading the association label when excluded.
- Multiplicity

- Records currently
  - Sale
  - Register
  - association name

- "associating class name"
Naming association

- Name an association based on a ClassName-VerbPhrase-ClassName format
- Eg.
  - Sales paid-by CashPayment
    - Bad alternative: Sales uses CashPayment
  - Piece is-on Square
    - Bad alternative: Piece has Square

Applying UML: multiplicity

- Multiplicity defines how many instances of a class A can be associated with one instance of class B

![Diagram of class relationships]

Multiplicity values

- Multiplicity focus on the relationship at a particular moment, rather than over a span of time.
- Eg. "In countries with monogamy laws, a person can be married to only one other person at any particular moment, even though over a span of time, that same person may be married to many persons."

<table>
<thead>
<tr>
<th>Multiplicity values</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>zero or more, “many”</td>
</tr>
<tr>
<td>1</td>
<td>one or more</td>
</tr>
<tr>
<td>1..40</td>
<td>one to 40</td>
</tr>
<tr>
<td>5</td>
<td>exactly 5</td>
</tr>
<tr>
<td>3, 5, 8</td>
<td>exactly 3, 5, or 8</td>
</tr>
</tbody>
</table>
Object Diagram to Class Diagram

-owns stock of

Mary : Person
GE : Company
Sue : Person
IBM : Company

Jeff : Person
-owns stock of

Multiple association between two classes

Description: Flight “QF435” flies from “Sydney” to “Melbourne”
Reflexive association

- A conceptual class may have association with itself
- Eg. A Directory can contain other directories
- Eg. A node in a liked list is linked to another node

Discussion Question

- Prepare a class diagram from the object diagram below

Find association

- Use the common associations list
  - A is a physical part of B
  - Square - Board
  - A is a logical part of B
  - SalesLineItem - Sale
  - A uses or submanages B
  - Cashier - Register
  - etc. (see table 9.2 on p. 155 for a fuller list)
- Omit associations that can be defined in terms of other associations.
  - Multiple paths between classes may indicate redundancy
  - However
    - keep the extra association if multiplicity is important.
    - Keep the extra association if they represent an important relationship in the real world.
### The Credit Card Problem

An institution may issue many credit card accounts, each identified by an account number. Each account serves one or more customers who reside at the mailing address. The institution periodically issues a statement for each account. The statement lists a payment due date, finance charge, and minimum payment. The statement itemizes various transactions that have occurred through the billing interval: cash advances, interest charges, purchases, fees, and adjustments to the account. The name of the merchant is associated for each purchase.

### POS Partial Domain Model

![POS Partial Domain Model Diagram]

### Attributes

- **An attribute** is a logical data value of an object.
- **When to show attribute**
  - Include attributes that the requirements suggest or imply a need to remember information.
  - **Eg.**
    - Sale needs a `dateTime` attribute.
    - Store needs a `name` and `address`. 

UML notation

- Attributes are shown in the second compartment of the class box.
- Type and other information may optionally be shown
- Derived attribute
  - The total attribute in the Sale can be calculated from other information
  - It is worth noting down.

```
Sale
<table>
<thead>
<tr>
<th>dateTime</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Money</td>
</tr>
</tbody>
</table>
```

Suitable attribute types

- Most attribute types should be what are often thought of as "primitive" data types
- Relate conceptual classes with an association, not with an attribute

```
Value
<table>
<thead>
<tr>
<th>Date</th>
<th>Local Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>User</td>
</tr>
</tbody>
</table>
```

Class or Attributes

- Class or attribute?
  - If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attributes

```
Date
| 2023-04-01 |
| or...    |

Flight
| 01:00 AM |
| or...    |
```

COMP5028 Object-Oriented Analysis and Design
© Dr. Ying Zhou, School of IT, The University of Sydney
Data types as attributes

- We sometimes represent what may initially be considered a number or string as a new data type class in the domain model.
- We can show the data type only in attribute compartment or as a separate class associated with another class.

Attributes are NOT foreign keys

- During implementation, there are many ways to relate objects, including foreign key, but the decision should be deferred till DESIGN stage.

Description Class

- What’s the rationale for representing the description or specification separately from the item?
- In the following domain model, can you tell the difference between (description, price, itemID) and serial number?
Description classes

- When are description classes useful?
  - There needs to be a description about an item or service, independent of the current existence of any examples of those items or services.
  - Deleting instances of things they describe results in a loss of information that needs to be maintained, but was incorrectly associated with the deleted things.
  - It reduces redundant or duplicated information.

Iterative and evolutionary domain model

- Is the domain model correct?
  - There is no such thing as a single correct domain model!
  - Domain model is incrementally evolve over several iterations.
  - In each iteration, the domain model is limited to the prior and current scenarios under consideration.

Discussion Question II

- Prepare a domain model to describe undirected graphs. An undirected graph consists of a set of vertices and a set of edges. Edges connect pairs of vertices.
- How would you modify the model to describe directed graphs?

Example of undirected graph

Example of directed graph
Generalization

- We can identify commonality among concepts and define super class (general concept) and sub class (specialized concept) relationships in a domain model.

![Diagram showing generalization relationships with classes and subclasses]

Conceptual super class and sub class

- **100% rule**
  - 100% of the conceptual superclass’s definition should be applicable to the subclass. The subclass must conform to 100% of the superclass’s attributes and associations.
- **Is-a rule**
  - All the instances of a subclass must be instances of their superclasses.

Conceptual subclass

- When to define a conceptual subclass?
  - The subclass has additional attributes of interest.
  - The subclass has additional associations of interest.
  - The subclass concept is operated on, handled, reacted to, or manipulated differently than the superclass or other subclasses, in a way that is of interest.
  - The subclass concept represents an animated thing (for example, animal, robot) that behaves differently than the superclass or other subclasses in a ways that are of interest.

![Diagram showing subclass relationships with classes and subclasses]
Example subclass partitions

<table>
<thead>
<tr>
<th>Conceptual Subclass Motivation</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass has additional associations of interests</td>
<td>CreditPayment, subclass of Payment, is associated with a CreditCard class.</td>
</tr>
<tr>
<td>The subclass is operated upon, handled, reacted to, or manipulated differently</td>
<td>CreditPayment, subclass of Payment, is handled differently than other kinds of payments in how it is authorized</td>
</tr>
<tr>
<td>The subclass concept represents an animated thing that behaves differently than the superclass or other subclasses</td>
<td>Car, Trucks are subclasses of PoweredVehicle, Car behaves differently from Truck</td>
</tr>
</tbody>
</table>

When to define a conceptual superclass

- Create a conceptual superclass in a generalization relationship to subclass when:
  - The potential conceptual subclasses represent variations of a similar concept.
  - The subclasses will conform to the 100% and Is-a rule.
  - All subclasses have the same attribute which can be factored out and expressed in the superclass.
  - All subclasses have the same association which can be factored out and related to the superclass.

Example
Thanks