

Development Cycle #2

- ❖ Add functionality to *Buy Items* use case:
 - cash, credit, and check payments are accepted
 - A different authorization service is used for each credit type (i.e. Visa, MasterCard, etc.)
 - The same authorization service is used for all checks written by customers
 - POST is responsible for communicating with each authorization service (usually a bank)

1

Buy Items Enhancements

- Check and credit payments are for the exact purchase amount only
- The credit card reader is a dumb device that only sends the card information to the terminal
- Communication with an external service is via a phone modem
- A phone number must be dialed each time an authorization is requested

2

Start Up Enhancements

- ❖ Assume the time and date are correct
- ❖ Minimal initialization required to support the enhanced *Buy Items* use case
- ❖ No initializing information is stored in the database

3

Buy Items Omissions

- No inventory maintenance
- Stand alone store support only (no chain stores)
- Manual entry of UPCs (no bar code reader)
- No tax calculations
- No coupons or special pricing policies
- Cashier does not have to log in
- No record is kept of individual buying habits
- No POST control of the cash drawer

4

Buy Items Omissions (cont.)

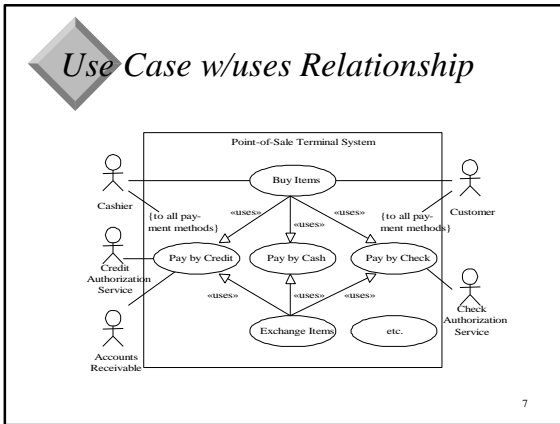
- Cashier ID and POST ID are not shown on the sales receipt (but store name, address, and date & time of the sale are shown)
- All completed sales are recorded in an historical log (but, no inventory control)
- Only one payment type is used per sale
- All payments are “in full”, no partial or installment payments are allowed

5

Relating Multiple Use Cases

- ❖ The different forms of payment may be expressed as separate use cases
- ❖ Update the use case diagram to show the relationships between these use cases
- ❖ The UML supports the *uses* and *extends* associations between use cases
- ❖ Payment options are primary candidates for both the *uses* and *extends* relationships

6



Use Cases with uses Relationship

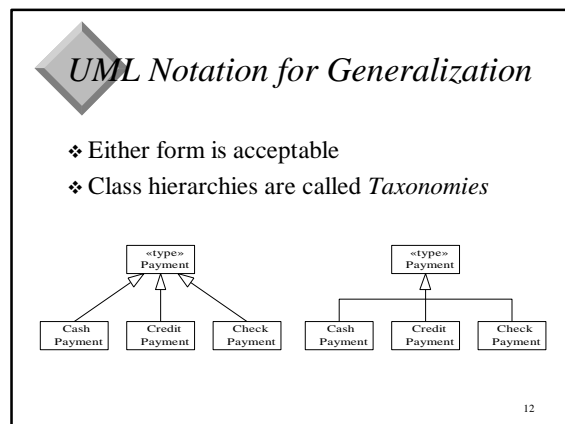
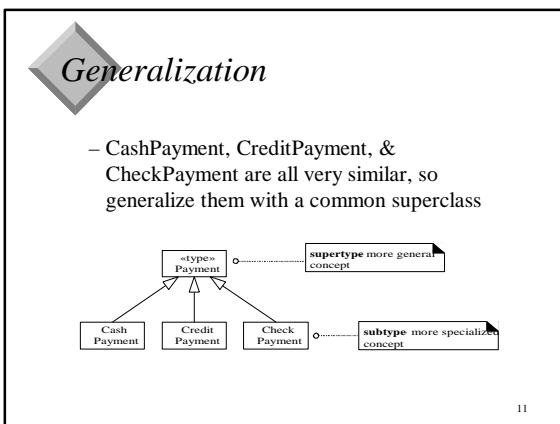
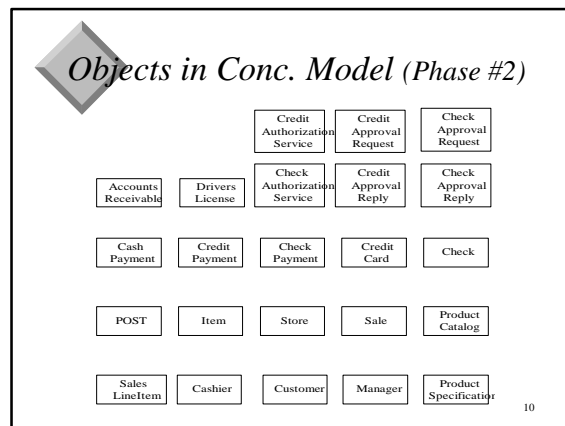
- ❖ When use cases are in a *uses* relationship, the use case documents need to express the association in prose
- ❖ The use case that uses the behavior of another should indicate the connection with the *initiate* keyword in the use case text
 - If X payment, **initiate** Pay by X
- ❖ Expanded Use Cases, pp. 324 - 328

8

Extending the Conceptual Model

- ❖ The new concepts in phase #2 include:
 - CreditCard, Check
 - CashPayment, CreditPayment, CheckPayment (i.e. Transaction types)
 - CreditAuthorizationService, CheckAuthorizationService (Obvious!)
 - AccountsReceivable
- ❖ Noun Phrase identification from Use Cases:

9



Generalization: the "is-a" Relationship

- Any derived object can be used where an object of the base type is required

13

Inheritance & Generalization

- The subclasses inherit attributes, associations, & operations from the parent

14

Avoid "Partitioning" Subtypes

- Partitioning leads to duplication
 - avoid "cut & paste" operations
- Only partition when functionality differs between subtypes

15

Inheritance

- Likewise, don't generalize unless there is common behavior among the subclasses

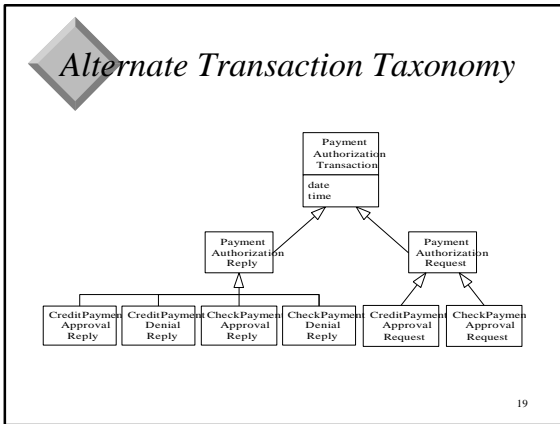
16

AuthorizationService Generalization

17

Avoid being too "fine grained"

18



Abstract Classes

- ❖ Some classes are too abstract to be instantiated as objects
 - Sensor, Car, Animal, etc. (can't find one!)
- ❖ Abstract classes are “placeholders” in a taxonomy, since they serve a useful abstraction purpose
- ❖ But, can not be instantiated!

