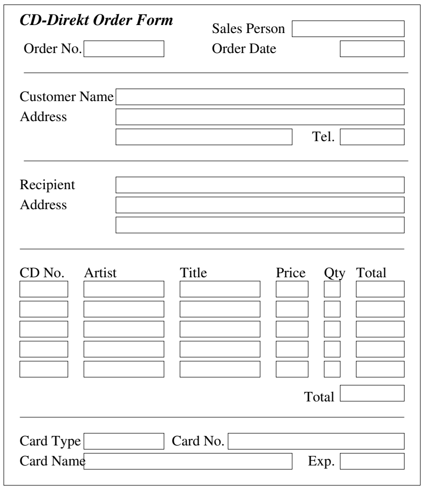
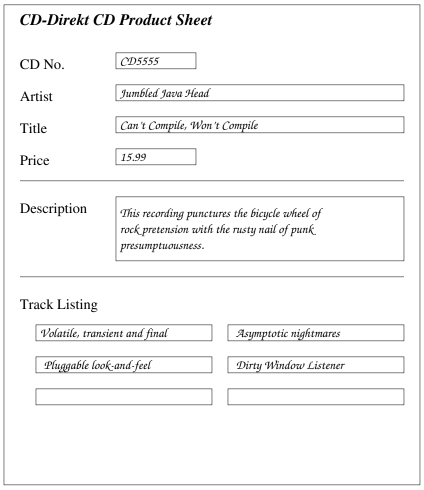
Lecture 1

CD-Direkt is a CD tele-sales company. It advertises exclusive CDs on television. Customers ring up to place orders for CDs. The sales person uses an order form to record customer details, order details (including the recipient of the order if different from the customer, and credit card payment details) and order lines (to record which CDs the customer wants to purchase). Information about CDs is kept in a ring-binder, containing one A4 sheet per CD. A credit check is made with the credit card company, while the customer is still on the line.

Completed orders are passed to the warehouse staff for despatch. The warehouse staff keep the outstanding order forms in a box near the kettle.

Sometimes customers ring up to find out the status of their orders. These calls presently get forwarded to the warehouse staff who can tell customers whether an order has been despatched or is waiting for stock.





Analysis

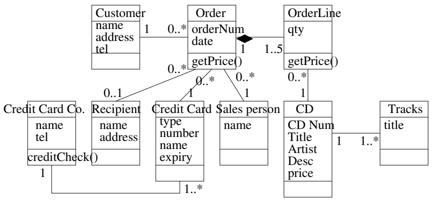
- It takes a long time to record an order (re-recording details of even those customers who have purchased from the company before; finding CD details in the ring-binder; calling the credit card company). This implies lost sales (people who can't get thro' give up.)

- Enquiries about orders are handled inefficiently. This implies lost sales (people don't re-order if they get a bad impression).

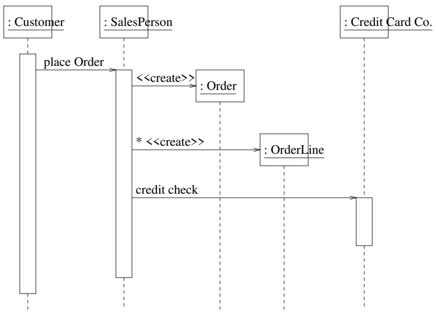
- *The Managing Director wants us to automate order entry and tracking to provide faster call processing.*

We model the problem domain (class diagram & sequence diagram) and determine use cases.

Analysis: Class Diagram



Analysis: One Sequence Diagram (placing an order)

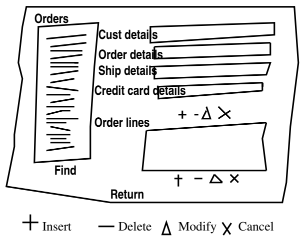


Analysis: Use Cases

Design: Class Diagram

Design: Sequence Diagrams for Use Cases (adding a new customer)

Design: GUI mock-up



Implementation

It should be easy to code up each class and the relationships between classes (IDEs even automate the production of skeletal code from UML Class Diagrams)

- The interface is the list of operations (plus perhaps some getters/setters that were too obvious to include until now)
- The attributes suggest instance variables
- The Sequence Diagrams help you to write the bodies of the methods
- Generalisation is typically implemented by using super/subclasses (`extends`)
- Association is typically implemented using references (navigability arrows determine whether to store references ‘at both ends’)