ICT2622 Object-Oriented Analysis Notes Phase Ch 8 – Summary Notes
SATZINGER JACKSON BURD (6TH EDITION)

CHAPTER 8: Approaches to System Development

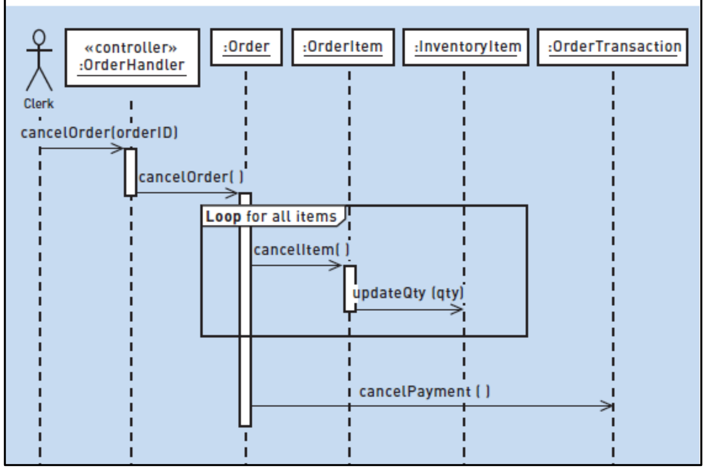
THE OBJECT-ORIENTED APPROACH

- The object-oriented approach—views an information system as a collection of interacting objects that work together to accomplish tasks
- There are no processes or programs; there are no data entities or files
- The system consists of objects.
- An object is a thing in the computer system that is capable of responding to messages.

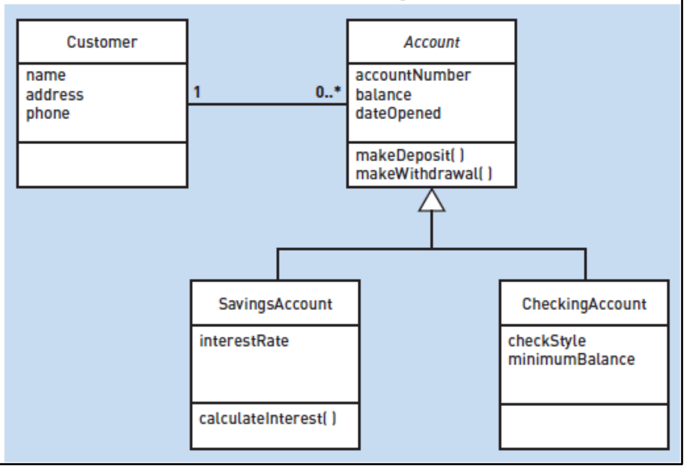
- Object-oriented analysis (OOA) - the process of identifying and defining the use cases and the sets of objects (classes) in the new system
- object-oriented analysis (OOA) defines objects that do the work and determines what user interactions (called use cases) are required to complete the tasks.
- Object-oriented design (OOD) defines all the additional types of objects that are necessary to with people and devices in the system, it shows how the objects interact complete tasks, and it refines the definition of each type of object so it can be implemented with a specific language or environment.
- Object-oriented programming (OOP) is the writing of statements in a programming language to define what each type of object does. object-oriented languages support object classes, inheritance, reuse, and encapsulation.

- An object is a type of thing - it could be a customer or an employee or it could be a button or a menu
- Identifying types of objects means classifying things.
- Some things, such as customers, exist outside and inside the system e.g. there is the real customer, who is outside the system, and the computer representation of the customer, which is inside the system
- A classification or "**class**" represents a collection of similar object
- Object-oriented development uses a UML class diagram (introduced in Chapter 4) to show all the classes of objects that are in the system
- For every class, there may be more specialized subclasses (superclass / subclass)
- Subclasses exhibit or "inherit" the characteristics of their superclass.
- A UML sequence diagram shows how objects interact or collaborate while carrying out a task
- The object-oriented approach yields several key benefits, among them naturalness and reusability
- Because the object-oriented approach involves classes of objects and many systems in the organization use the same objects, these classes can be used over and over again whenever they are needed. For example, almost all systems use
- menus, dialog boxes, windows, and buttons, but many systems within the same company also use customer, product, and invoice classes

A UML sequence diagram showing object interactions for a use case



A class diagram created during object-oriented design



ICT2622 Object-Oriented Analysis Notes Phase Ch 8 – Summary Notes
SATZINGER JACKSON BURD (6TH EDITION)

CHAPTER 8: Approaches to System Development

AGILE DEVELOPMENT

- Agile development - is a philosophy and set of guidelines for developing information systems in an unknown, rapidly changing environment, and it can be used with any system development methodology
- Agile Modelling is a philosophy about how to build models, some of which are formal and detailed, others sketchy and minimal

Agile Development Philosophy and Values

- Four basic values:
  - Value responding to change over following a plan
  - Value individuals and interactions over processes and tools
  - Value working software over comprehensive documentation
  - Value customer collaboration over contract negotiation

- Agile development provides more flexibility in project schedules and letting the project teams plan and execute their work as the project progresses
- Customers must continually be involved with the project team
- Because working software is being developed throughout the project, customers are continually involved in defining requirements and testing components.
- Models and modelling are critical to Agile development

Agile Modelling Principles

1. Develop Software as Your Primary Goal
2. Enable the Next Effort as Your Secondary Goal - although high-quality software is the primary goal, long-term use of that code is also important
3. Minimize your modelling activity (few and simple) - Create only the models that are necessary. Do just enough to get by.
4. Embrace change, and change incrementally - be flexible and respond quickly to change, change is the norm.
5. Model with a purpose - develop the model in sufficient detail to satisfy the reason and the audience.
6. Build multiple models - enough models to make sure you have addressed all the issues.
7. Build high-quality models and get feedback rapidly - to avoid error in models, get feedback rapidly while the work is still fresh
8. Focus on content rather than representation
9. Learn from each other with open communication - adaptive approaches emphasize working in teams
10. Know your models and how to use them
11. Adapt to specific project needs - adapt to fit the needs of the business and the project