Software Development

Modular Design and Algorithm Analysis
• **Functional Decomposition** in computer science, also known as **factoring**, refers to the process by which a complex problem or system is broken down into parts that are easier to conceive, understand, program, and maintain.

• There are different types of functional decomposition defined in computer sciences:
• In **structured programming**, algorithmic decomposition breaks a process down into well-defined steps.

• **Structured analysis** breaks down a software system from the system context level to system functions and data entities.

• **Object-oriented** decomposition, on the other hand, breaks a large system down into progressively smaller classes or objects that are responsible for some part of the problem domain.
• Algorithmic decomposition is a necessary part of object-oriented analysis and design, but object-oriented systems start with and emphasize decomposition into classes.
• More generally, functional decomposition in computer science is a technique for mastering the complexity of the function of a model.
• A functional model of a system is thereby replaced by a series of functional models of subsystems.
Class-Responsibility-Collaboration

- **Class-responsibility-collaboration (CRC) cards** are a brainstorming tool used in the design of object-oriented software.
- They were originally proposed by Ward Cunningham and Kent Beck as a teaching tool, but are also popular among expert designers and recommended by extreme programming supporters.
- Martin Fowler has described CRC cards as a viable alternative to UML sequence diagram to design the dynamics of object interaction and collaboration.
• CRC cards are usually created from index cards.

• Members of a brainstorming session will write up one CRC card for each relevant class/object of their design.

• The card is partitioned into three areas:
  1. On top of the card, the **class** name
  2. On the left, the **responsibilities** of the class
  3. On the right, **collaborators** (other classes) with which this class interacts to fulfill its responsibilities
• Using a small card keeps the complexity of the design at a minimum.

• It focuses the designer on the essentials of the class and prevents her/him from getting into its details and inner workings at a time when such detail is probably counter-productive.

• It also forces the designer to refrain from giving the class too many responsibilities.

• Because the cards are portable, they can easily be laid out on a table and re-arranged while discussing a design with other people.
Unified Modeling Language

- **The Unified Modeling Language (UML)** is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system.

- It was created and developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software during 1994–95 with further development led by them through 1996.
• In 1997 it was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since.

• In 2000 the Unified Modeling Language was also accepted by the International Organization for Standardization (ISO) as an approved ISO standard.

• Since then it has been periodically revised to cover the latest revision of UML.
• The Unified Modeling Language (UML) offers a way to visualize a system's architectural blueprints in a diagram, including elements such as:
  – Any activities (jobs)
  – Individual components of the system and how they can interact with other software components.
  – How the system will run
  – How entities interact with others (components and interfaces)
  – External user interface
• Although originally intended solely for object-oriented design documentation, it has been extended to cover a larger set of design documentation.
• **UML** has many types of diagrams which are divided into two categories.

• **Some types represent structural information,** and the rest represent general types of **behaviour,** including a few that represent different aspects of **interactions.**
Structural UML diagrams

- Class diagram
- Component diagram
- Composite structure diagram
- Deployment diagram
- Object diagram
- Package diagram
- Profile diagram
Behavioral UML diagrams

- Activity diagram
- Communication diagram
- Interaction overview diagram
- Sequence diagram
- State diagram
- Timing diagram
- Use case diagram
Activity 3.4

- Use the CRC and the UML methodology to decompose two functions for your culminating activity. (one for each method)

- Save your work as john_s_3_4_OODM under the appropriate folder for your class in the DropOff folder on the X: drive