

1. Introduction





System

- A system is an arrangement where all its component work according to the specific defined rules. It is a method of organizing, working, or performing one or more tasks according to a fixed plan.



Embedded System

- **Embedded System:** which carries out a defined function and is embedded in a physical environment, is optionally surrounded by other subsystems and has an optional user interface.



Embedded System

- It is mostly designed for a specific function or functions within a larger system. For example, a fire alarm is a common example of an embedded system which can sense only smoke.



Embedded Software

- **Embedded Software:** is a piece of software that is embedded in hardware or non-PC devices. It is written specifically for the particular hardware that it runs on and usually has processing and memory constraints because of the device's limited computing capabilities.



Embedded Software

- Examples of embedded software include those found in dedicated GPS devices, factory robots, some calculators and even modern smartwatches



Design (activity)

- The activity that defines how a system is built from several components (architecture elements).



System Design

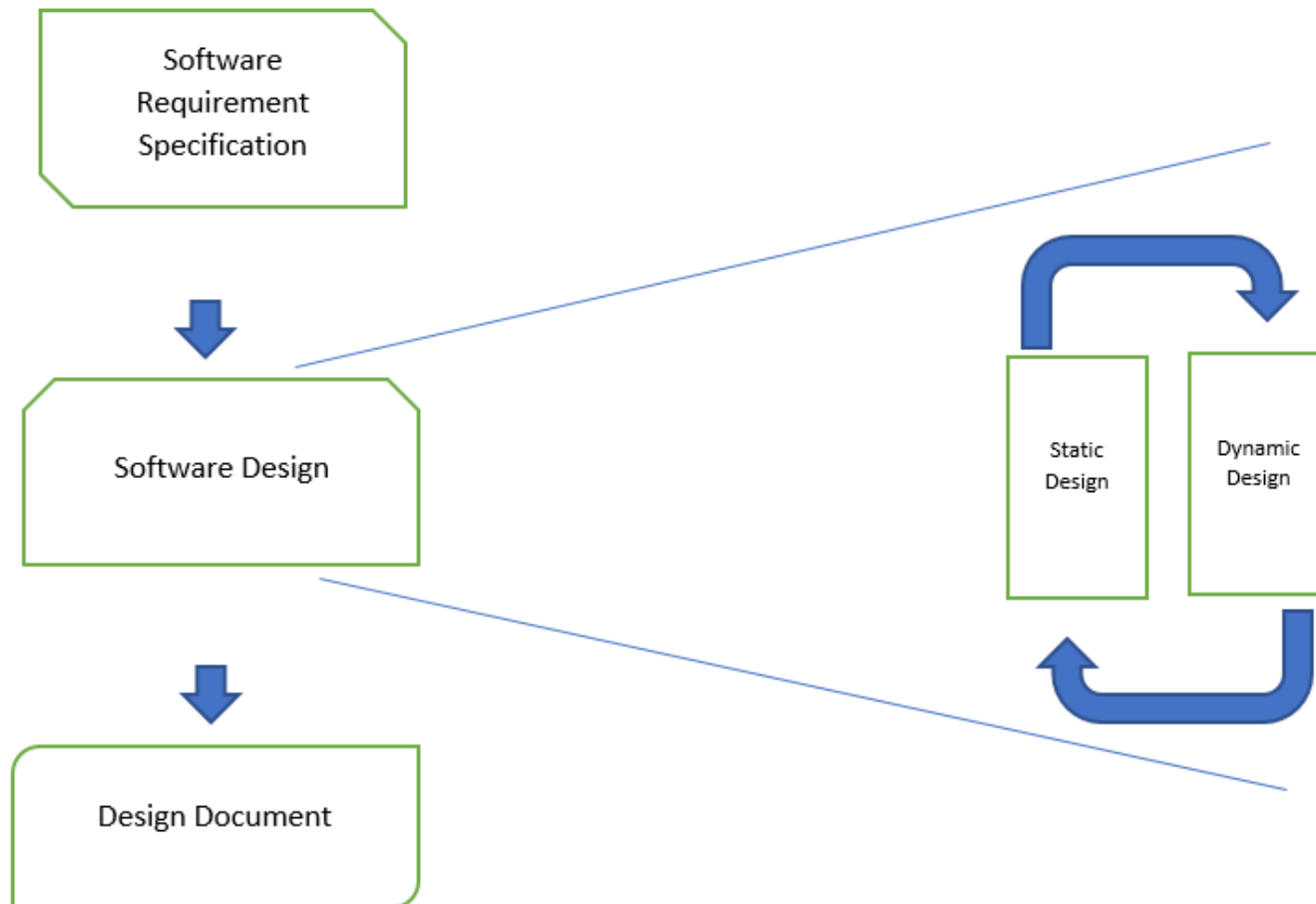
- with the specification that it is the design of a system, i.e. various elements such as hardware, software, mechanics.



Software Design

- with the specification that it is the design of a software (software elements).

Design scope





Design scope

- In the static design, you define the structure, the structure includes modules and how they connected. On the other hand, the dynamic design defines how the modules interact together.



Design scope

- Static design is responsible for executing the correct function, on the other hand, the dynamic design enables the function to be executed at the correct time.



Objective of the course

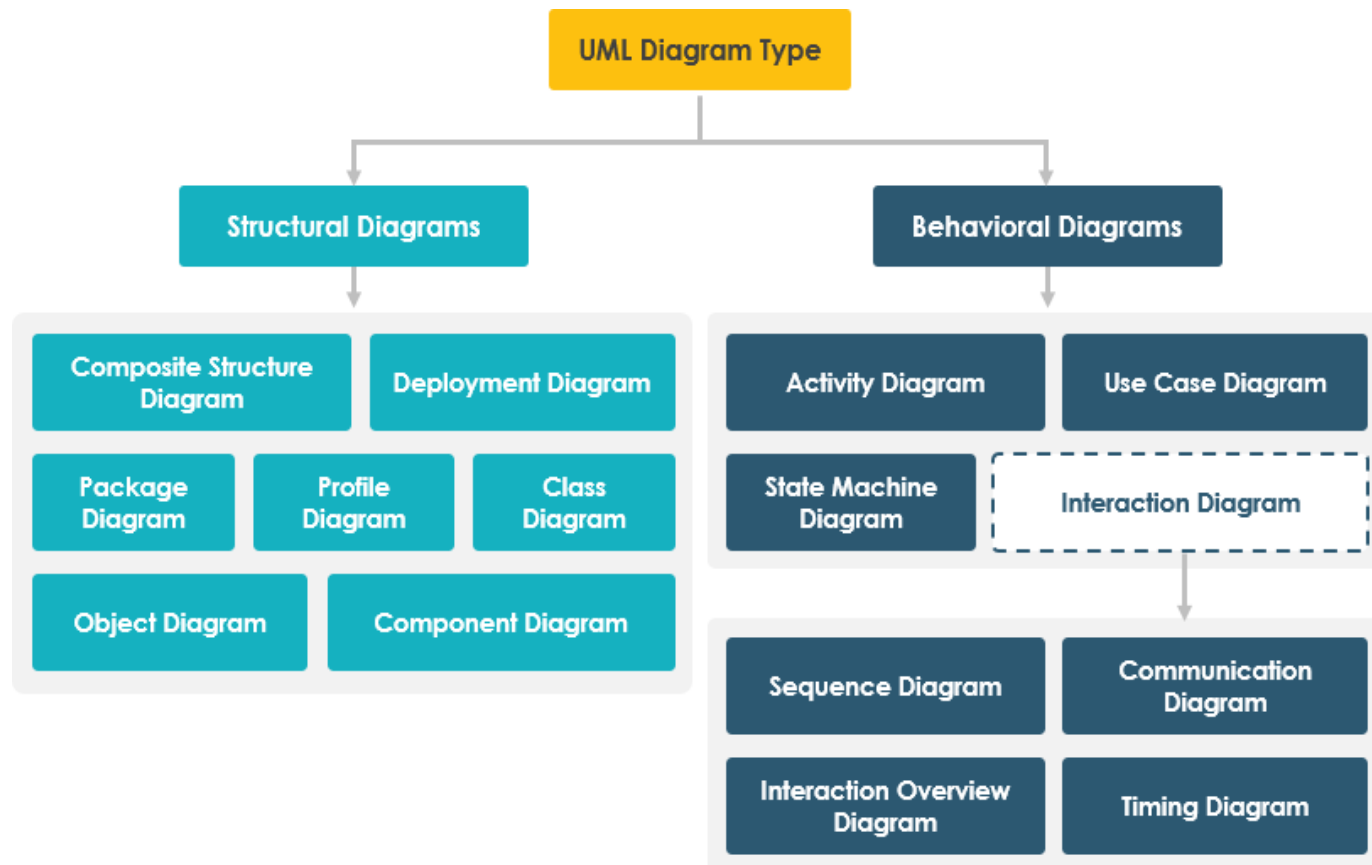
1. To build and analyze models for embedded application using the concept of UML.
2. To work with UML tools and represent the model using suitable diagrams.
3. To write applications using the OOP concepts
4. To write applications using JAVA constructs for general purpose and embedded systems



UML

- It is a generic developmental modelling language used for analysis, design and implementation of software systems. The purpose of UML is to provide a simple and common method to visualize a software system's inherent architectural properties.

UML





UML

- **Structural (Static) view:**

emphasizes the static structure of the system using objects, attributes, operations and relationships. It includes class diagrams and composite structure diagrams. Its specify the structure of the object.



UML

- **Behavioral(Dynamic) view:**

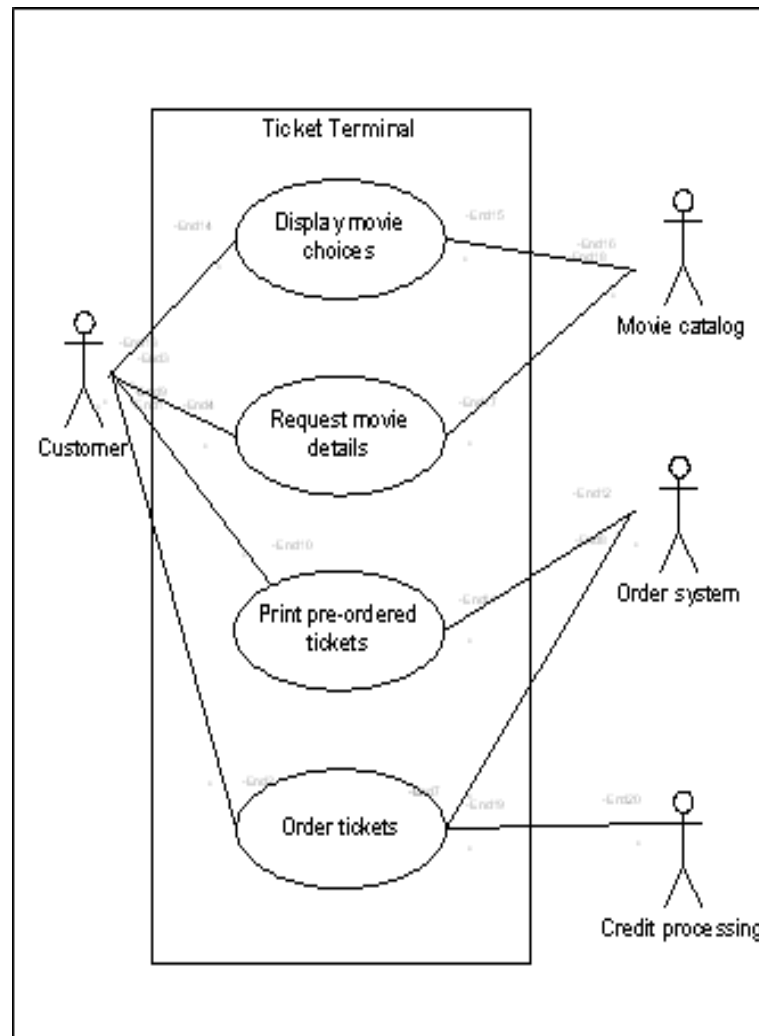
emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams, and state machine diagrams. Its represent the object interaction during runtime.



UML

- If you are describing what the program is able to *do*, you might write user stories, then draw out **use case diagrams** to elaborate on them.

UML

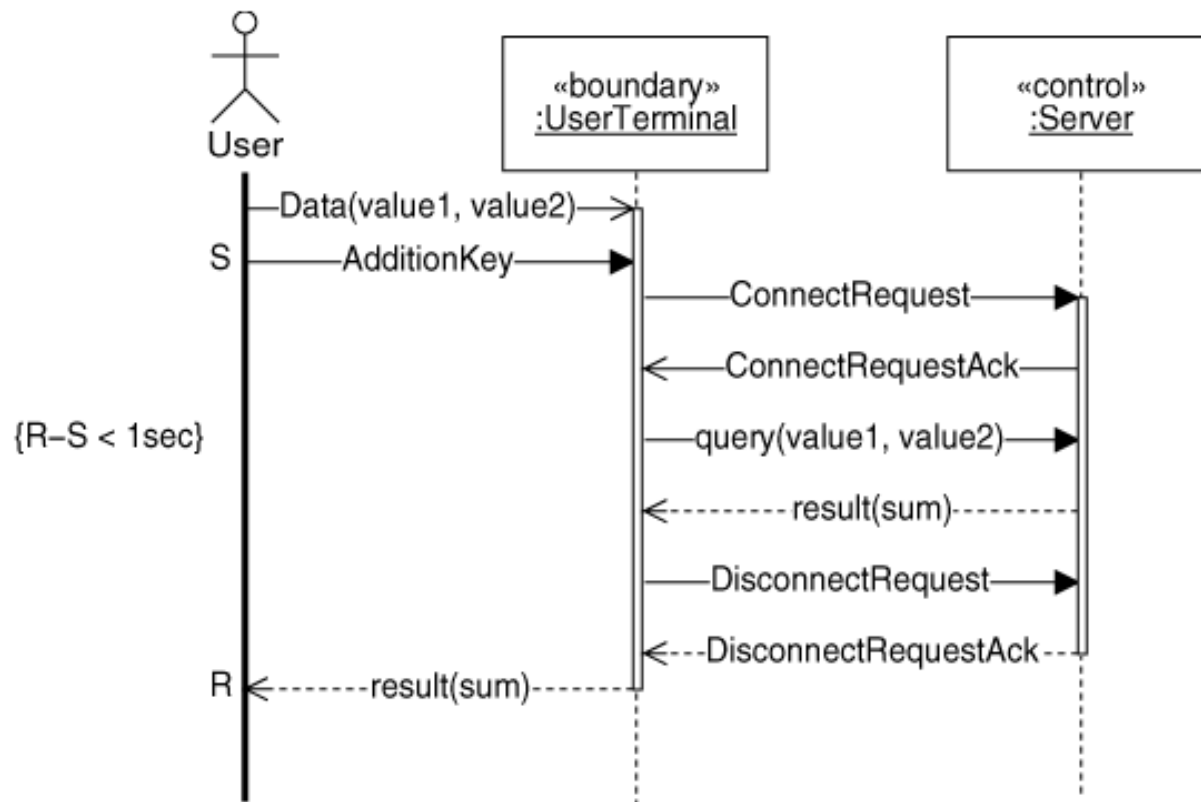




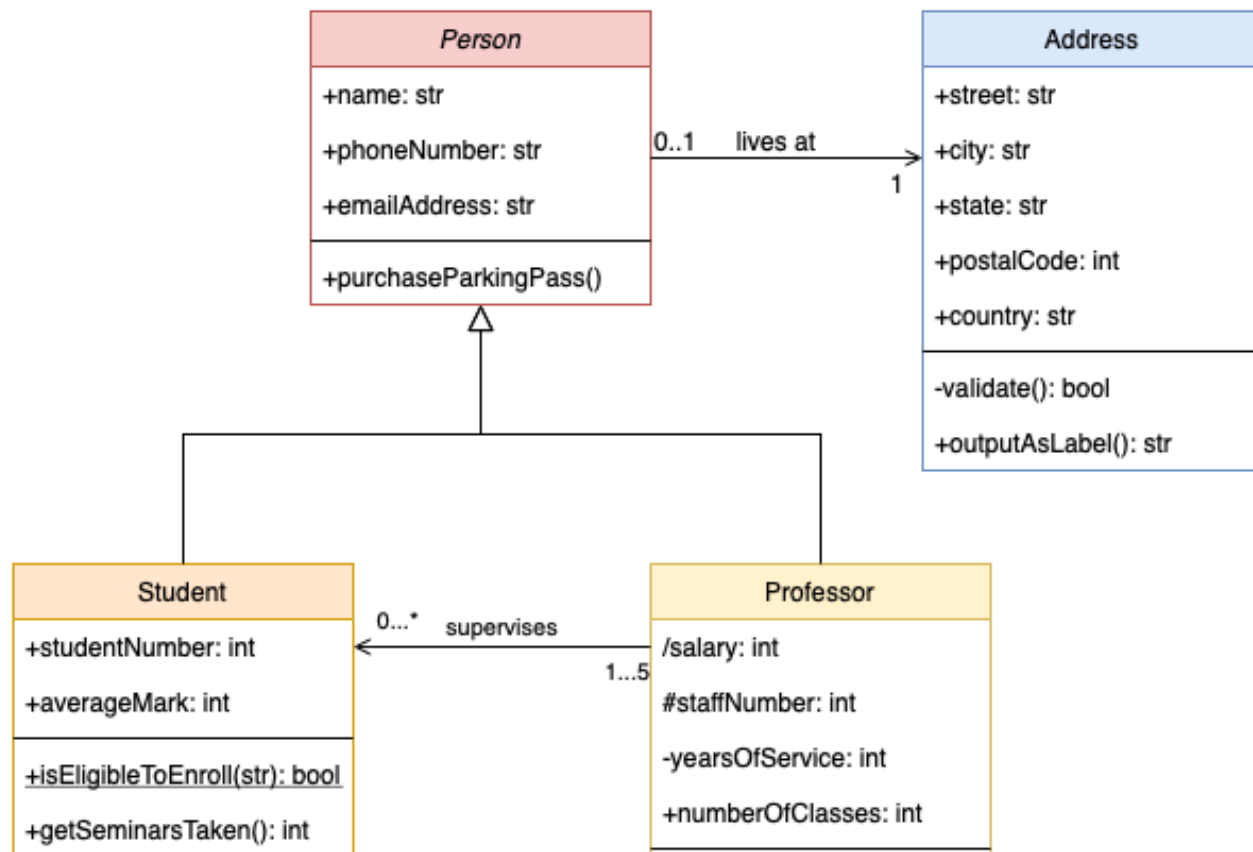
UML

- Once you have an idea what the program is able to do, you might design the structure of the program. Things like **sequence diagrams** and **class diagrams** help structure your program.

UML



UML





UML

- Describing the state of a program's execution might help explain a program's implementation of a feature, so **object diagrams** help in this by giving a snapshot of a program's state.

UML

