UML Use Case Diagrams

"It's a diagram that shows relationships between actors and uses cases in a system."

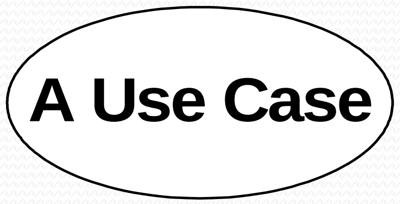
What is a use case diagram?

- "It's a diagram that shows relationships between actors and uses cases in a system."
- *"A use case diagram is a visual representation of the relationships between actors and use cases together that documents the system's intended behavior"*
- Use case diagrams are used during ***requirements elicitation and analysis as a graphical*** means of representing the functional requirements of the system.
- Use cases are developed during requirements elicitation and are further refined and corrected as they are reviewed (by stakeholders) during analysis.

Use Case Diagrams

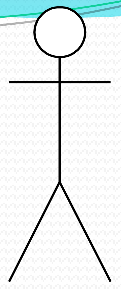
- Description of a system’s behavior as it responds to a request that originates from outside of that system. Describes a set of sequences.
- Each sequence represents the interactions of things outside the system (*actors*) with the system itself (and key abstractions)
- Use cases represent the *functional requirements* of the system (non-functional requirements must be given elsewhere)

Use case



- Each use case has a descriptive name
- Describes what a system does but not how it does it.
- Use case names must be unique within a given package
- Examples: withdraw money, process loan
- The use case names starts with a strong verb

Actor


An Actor

- Actors have a name
- An actor is a set of roles that users of use cases play when interacting with the system
- An actor is a person, organization, or external system that plays a role in one or more interactions with your system
- They are external entities
- They may be external an system or DB
- Actors are typically drawn as stick figures on UML Use Case diagrams
- Examples: Customer, Loan officer

Actor (Cont..)

- Draw Actors To The Outside Of A Use Case Diagram
- Name Actors With Singular, Business-Relevant Nouns
- Associate Each Actor With One Or More Use Cases
- Actors Model Roles, Not Positions
- Use **<<system>>** to Indicate System Actors
- Actors Don’t Interact With One Another

What is a Use Case?

- Use case captures some user-visible functionality
- Granularity of functionality depends on the level of detail in your model
- Each use case achieves a discrete goal for the user
- Use Cases are generated through requirements elicitation
- A use case describes a sequence of actions that provide a measurable value to an actor.
- A use case is drawn as a horizontal ellipse on a UML use case diagram

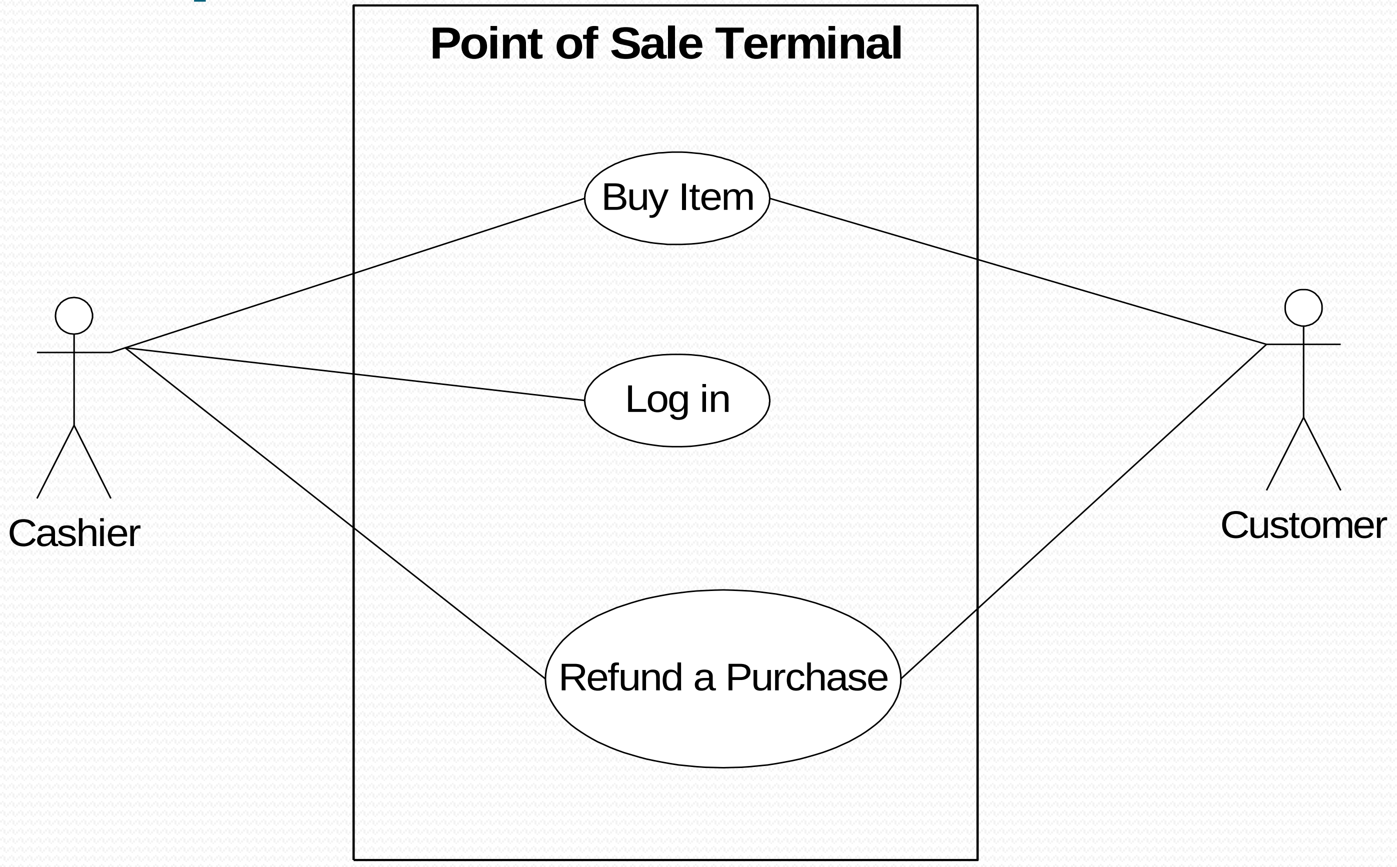
Goals vs. Interaction

- Goals – something the user wants to achieve
  - Format a document
  - Ensure consistent formatting of two documents
- Interaction – things the user does to achieve the goal
  - Define a style
  - Change a style
  - Copy a style from one document to the next

Developing Use Cases

- Understand what the system must do – capture the goals
- Understand how the user must interact to achieve the goals – capture user interactions
- Identify sequences of user interactions
- Start with goals and refine into interactions

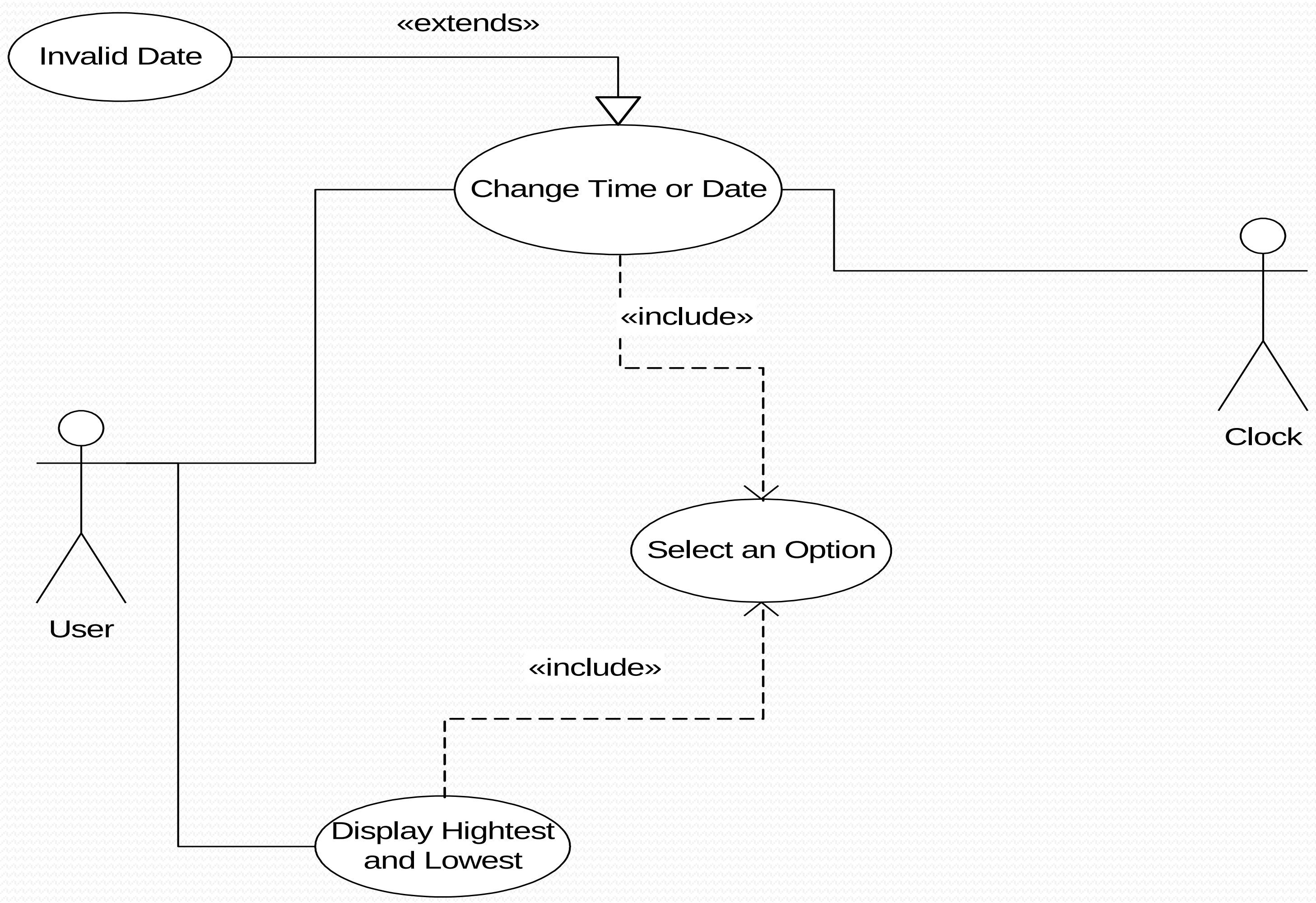
Example


Point of Sale Terminal
Buy Item
Log in
Refund a Purchase
Cashier
Customer

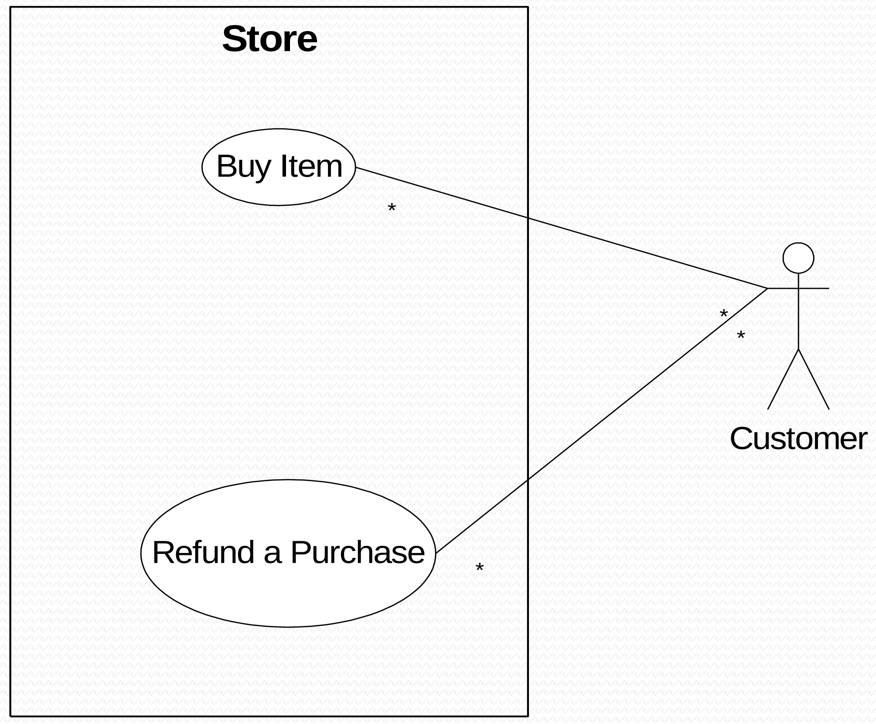
Refining Use Cases

- Separate internal and external issues
- Describe flow of events in text, clearly enough for customer to understand
  - Main flow of events
  - Exceptional flow of events
- Show common behaviors with *includes*
- Describe extensions and exceptions with *extends*

Extend and Include



System Boundary Box



System Boundary Box (Cont…)

- The rectangle around the use cases is called the system boundary box
- As the name suggests it indicates the scope of your system
    - The use cases inside the rectangle represent the functionality that you intend to implement.

Use Case – Buy Item

- Actors: Customer (initiator), Cashier
- Type: Primary
- Description: The costumer arrives at the checkout with items to purchase. Cashier records purchases and collects payment. Customer leaves with items

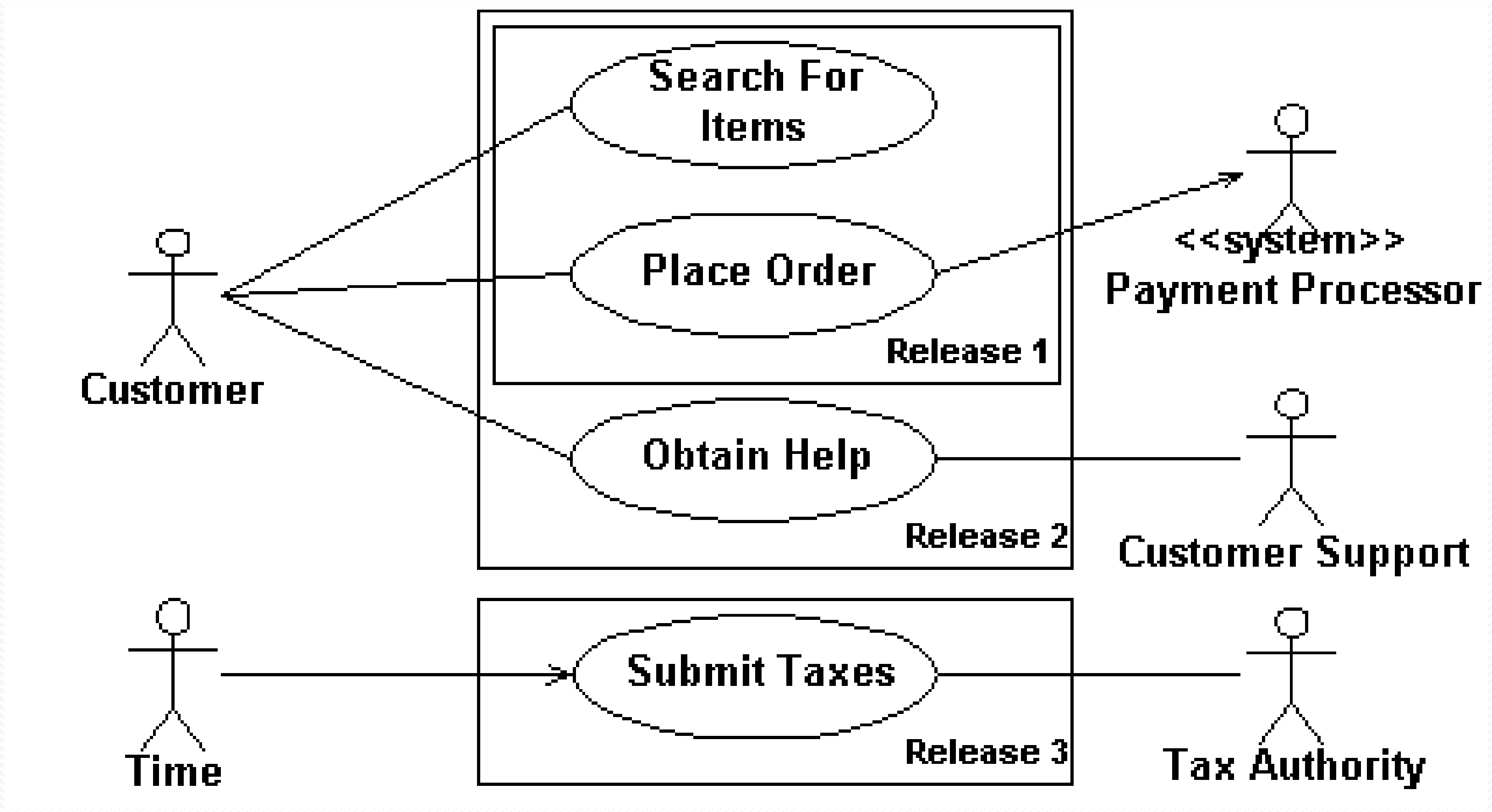
Relationships

- Types of relationships that may appear on a use case diagram:
  - An association between an actor and a use case
  - An association between two use cases
  - A generalization between two actors
  - A generalization between two use cases

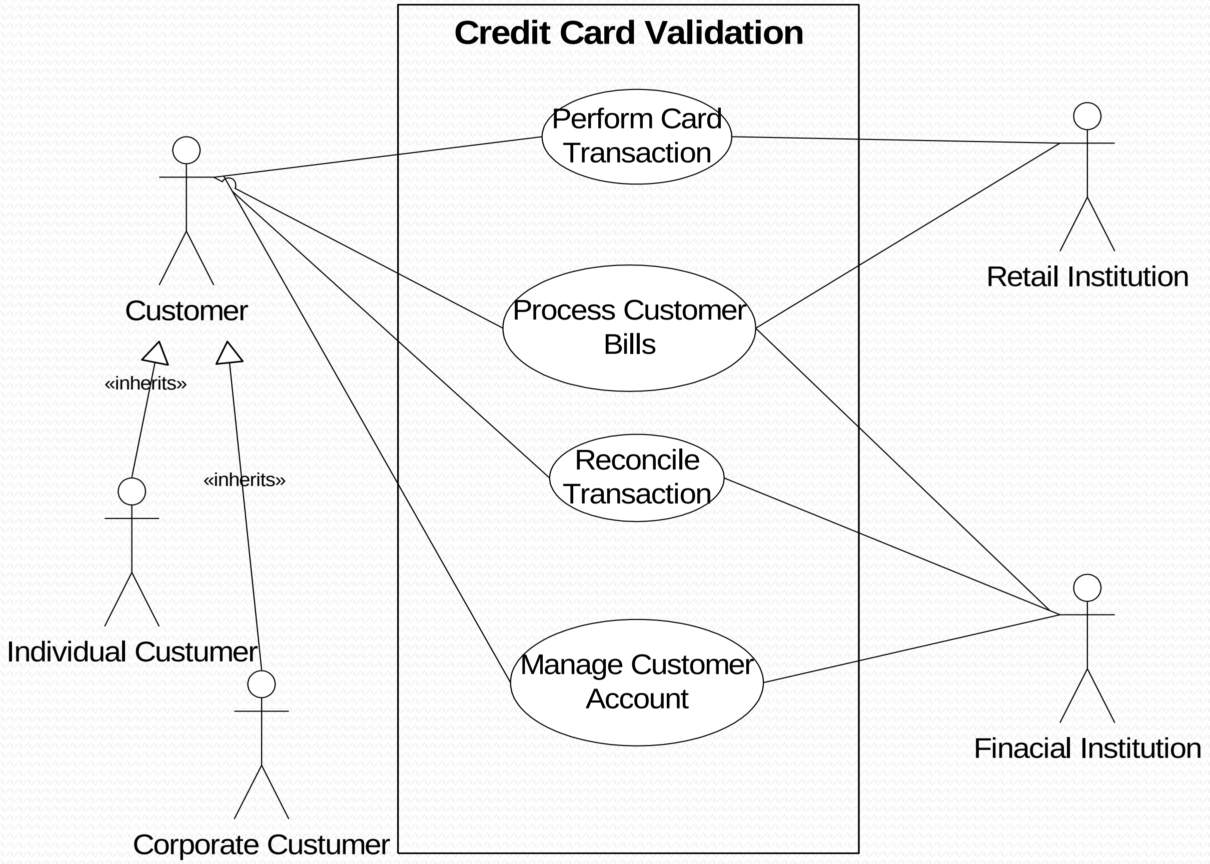
Relationships (Cont…)

- Associations are depicted as lines connecting two modeling elements with an optional open-headed arrowhead on one end of the line indicating the direction of the initial invocation of the relationship.
- Generalizations are depicted as a close-headed arrow with the arrow pointing towards the more general modeling element.

Example (Associations)



Example (Generalization)



Example: Weather Monitoring Station

- This system shall provide automatic monitoring of various weather conditions. Specifically, it must measure:
  - wind speed and direction
  - temperature
  - barometric pressure
  - humidity
- The system shall also proved the following derived measurements:
  - wind chill
  - dew point temperature
  - temperature trend
  - barometric pressure trend

Weather Monitoring System Requirements

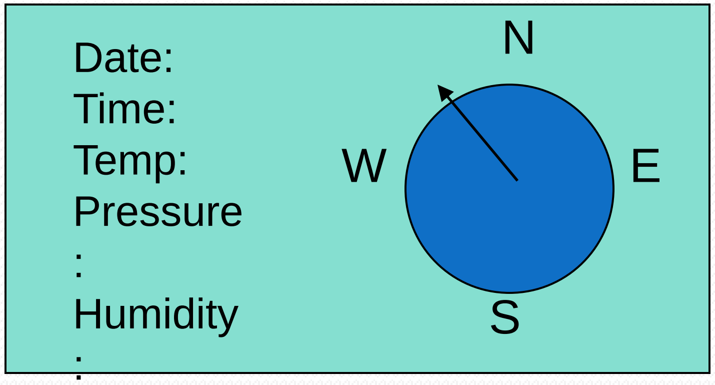
- The system shall have the means of determining the current time and date so that it can report the highest and lowest values for any of the four primary measurements during the previous 24 hour period.
- The system shall have a display that continuously indicates all eight primary and derived measurements, as well as current time and date.
- Through he use of a keypad the user may direct the system to display the 24 hour low or high of any one primary measurement, with the time of the reported value.
- The system shall allow the user to calibrate its sensors against known values, and set the current time and date.

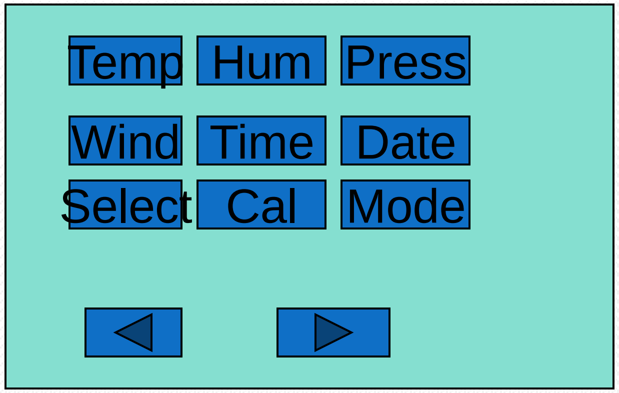
Hardware Requirements

- Use a single board computer (486?)
- Time and date are supplied by an on-board clock accessible via memory mapped I/O
- Temperature, barometric pressure, and humidity are measured by on board circuits with remote sensors.
- Wind direction and speed are measure from a boom encompassing a wind vane (16 directions) and cups (which advance a counter every revolution)
- User input is provided through an off the shelf keypad, managed by onboard circuit supplying audible feed back for each key press.
- Display is off the self LCD with a simple set of graphics primitives.
- An onboard timer interrupts the computer every 1/60 second.

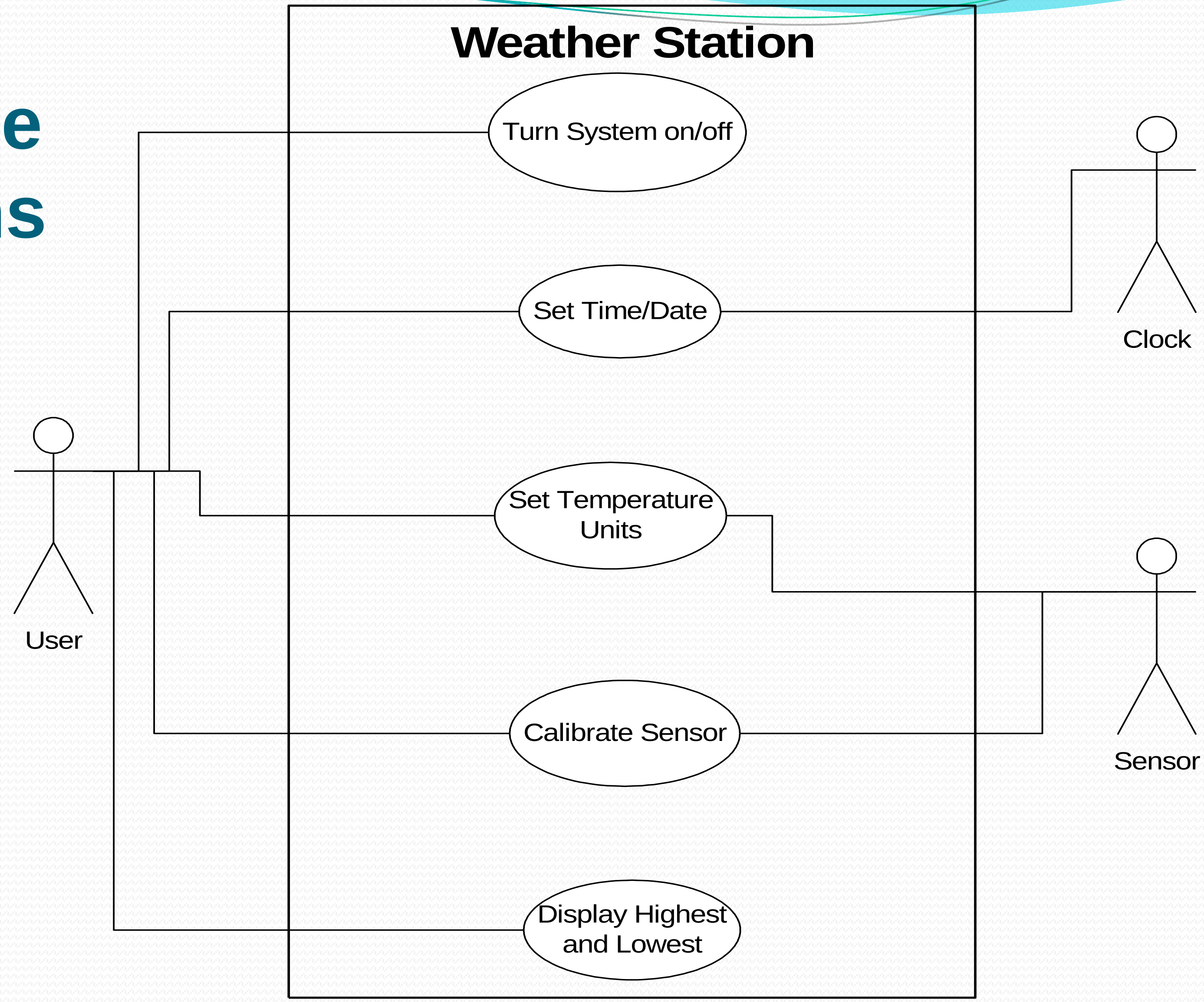
Display and Keypad

- LCDDisplay – Values and current system state (Running, Calibrating, Selecting, Mode)
  - Operations: drawtext, drawline, drawcircle, settextsize, settextstyle, setpensize
- Keypad allows user input and interaction
  - Operations: last key pressed
  - Attributes: key





Use Case Diagrams



Scenario: Powering Up

1. Power is turned on
2. Each sensor is constructed
3. User input buffer is initialized
4. Static elements of display are drawn
5. Sampling of sensors is initialized

The past high/low values of each primary measurement is set to the value and time of their first sample.

The temperature and Pressure trends are flat.

The input manager is in the Running state

Scenario: Setting Time and Date

1. User presses Select key
2. System displays selecting
3. User presses any one of the keys Time or Date. Any other key is ignored except Run
4. System flashes the corresponding label
5. Users presses Up or Down to change date or time.
6. Control passes back to step 3 or 5

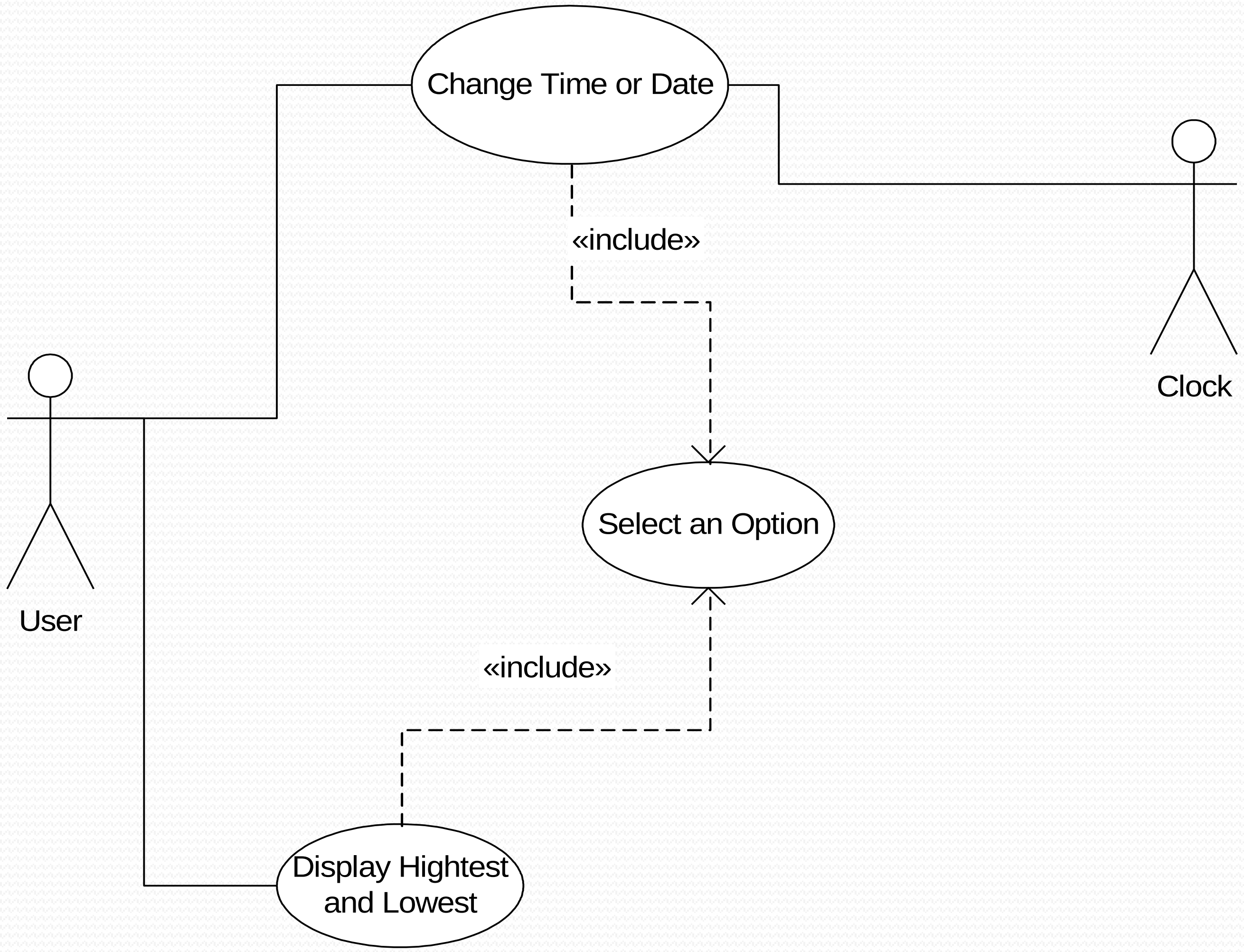
User may press Run to abandon the operation.

Scenario: Display highest and lowest

1. User presses Select key
2. System displays selecting
3. User presses any one of the keys (Wind, Temp, Humidity, Pressure). Any other key is ignored except Run
4. System flashes the corresponding label
5. Users presses Up or Down to select display of highest or lowest in 24 hour period. Any other key press is ignored except for Run
6. System displays value with time of occurrence
7. Control passes back to step 3 or 5

User may press Run to abandon the operation.

Use Case Diagrams



Well-Structured Use Cases

- Names a single identifiable and reasonably atomic behavior of the system
- Factors common behavior by pulling such behavior from other use cases that include it
- Factors variants by pushing such behavior into other uses cases that extend it
- Describes flow of events clearly
- Described in a minimal set of scenarios

END

- NEXT: We will look at Class Diagrams