UML sequence diagrams

- *sequence diagram*: an "interaction diagram" that models a single scenario executing in the system
  - perhaps 2nd most used UML diagram (behind class diagram)

- relation of UML diagrams to other exercises:
  - CRC cards \(\rightarrow\) class diagram
  - use cases \(\rightarrow\) sequence diagrams

### Sequence diag. key parts

- **participant**: object or entity that acts in the diagram
  - diagram starts with an unattached "found message" arrow

- **message**: communication between participant objects

- the axes in a sequence diagram:
  - horizontal: which object/participant is acting
  - vertical: time (down \(\rightarrow\) forward in time)
Representing objects

- Squares with object type, optionally preceded by "name:"
  - write object's name if it clarifies the diagram

Messages between objects

- messages (method calls) indicated by arrow to other object
  - write message name and arguments above arrow

Messages, continued

- messages (method calls) indicated by arrow to other object
  - dashed arrow back indicates return
  - different arrowheads for normal / concurrent (asynchronous) calls

Lifetime of objects

- creation: arrow with 'new' written above it
  - notice that an object created after the start of the scenario appears lower than the others
- deletion: an X at bottom of object's lifetime
  - Java doesn't explicitly delete objects; they fall out of scope and are garbage-collected
Indicating method calls

- **activation**: thick box over object’s life line; drawn when object’s method is on the stack
  - either that object is running its code,
  - or it is on the stack waiting for another object’s method to finish

Selection and loops

- **frame**: box around part of diagram to indicate if or loop
  - if -> (opt)
    - [condition]
  - if/else -> [alt]
    - [condition],
      separated by horizontal dashed line
  - Loop -> (loop)
    - [condition or items to loop over]

Linking sequence diagrams

- If one diagram is too large or refers to another, indicate with:
  - an unfinished arrow and comment,
  - or a “ref” frame that names the other diagram

sd Example

```
Controller

*Actual recursion*

Activation

ref
```
Forms of system control

- What can you say about the control flow of each system?
  - Is it centralized?
  - Is it distributed?

Why not just code it?

- Sequence diagrams can be somewhat close to the code level.
- So why not just code up that algorithm rather than drawing it as a sequence diagram?
  - A good sequence diagram is still a bit above the level of the real code (not all code is drawn on diagram)
  - Sequence diagrams are language-agnostic (can be implemented in many different languages)
  - Non-coders can do sequence diagrams
  - Easier to do sequence diagrams as a team
  - Can see many objects/classes at a time on same page (visual bandwidth)

Sequence diagram exercise

- Let's do a sequence diagram for the following poker casual use case, **Start New Game Round**

The scenario begins when the player chooses to start a new round in the UI. The UI asks whether any new players want to join the round; if so, the new players are added using the UI.

All players’ hands are emptied into the deck, which is then shuffled.

The player left of the dealer supplies an ante bet of the proper amount. Next each player is dealt a hand of two cards from the deck in a round-robin fashion; one card to each player, then the second card.

If the player left of the dealer doesn’t have enough money to ante, he/she is removed from the game, and the next player supplies the ante. If that player also cannot afford the ante, this cycle continues until such a player is found or all players are removed.
Sequence diagram question

- Consider the possible poker use case, *Betting Round*
- Why is it hard to diagram this case as a sequence diagram?

The scenario begins after the *Start New Round* case has completed. The UI asks the first player for a bet. That player chooses to either bet a given amount, or check (no bet).

The next player is asked what to do. If the prior player placed a bet, the next player must either match ("see") it, or match it plus add an additional bet ("raise"), or choose not to match and exit the round ("fold"). This continues around the table until an entire pass is made in which all players have either matched all other players' bets or folded.

If the next player doesn't have enough money to match the current bet, the player is allowed to bet all of their money. But they can then win only up to the amount they bet; the rest is a "side pot" among the more wealthy players remaining in the round.

Questions?