Developing programs

- What's the purpose of a Makefile?
  - Dependencies, recompiles
  - Environment dependencies

- How does a makefile work?
  - Key features: target, dependencies, command(s)
  - When target is invoked, changes in dependencies can cause command(s) to execute
  - What's the silly Makefile tab rule? Continued lines?

- See Makefile guide on class resource page

Multiple developer issues

- How do three people work on the same program?
  - Each works on the whole thing, copy files
  - Agree to work on parts, copy files
  - Make a shared directory, what about simultaneous development?

- Need a versioning system that supports multiple developers
  - Let system worry about simultaneous development
  - Ensure integrity of system, roll back to previous versions

- CVS, RCS, Source safe, Sourceforge, Savannah, others
  - Read resources page, use CVS now!!!

Inheritance issues (see pile.cpp)

- Here's a scenario: is a Deck a Pile? Is a Pile shuffleable
  - Should all piles be shuffleable because a Deck is?
  - Can we declare a Deck * and use it rather than a Pile *?
  - Can we assign a Pile * to a Deck *? Vice versa?

- We can leverage multiple inheritance and create a Shuffleable class either as standalone or as a subclass of Pile
  - Advantages? Methods for doing this?
  - Tradeoffs in multiple inheritance?

- What about pure virtual functions?

Exceptional situations

- Do we always need to check if we can add a card to a pile?
- Do we need to check that we can take a card?
- What do we return from `takeCard` when there are none?
  - Require user to check precondition, but then what?

- We can use exceptions
  - Throw exceptions
  - Catch exceptions
  - See pile.cpp for details