Objectives

- Midterm prep
- Defining our own classes
  - Some more tricks
- Designing our own classes

Midterm Prep

- Midterm: Friday
- Prep document online
- Similar problems to last exam
  - Very short answer
  - Short answer
  - Reading code (what's the output)
  - Writing code, comments
- Slightly more emphasis on writing code

Creating a Counter Class

- Has a fixed range
- Starts at some low value, increments by 1, loops back around to low value if gets beyond some maximum value
- Example application of the counter: Caesar cipher for letters 'a' to 'z'

What is the *API* for this object/class? → Object o of type Counter

- What are the *attributes* of an object in the class?
- What *data* should be used to *represent* an object in the class?

Creating a Counter Class

- Data: Instance variables that represent
  - High, Low, Current Value
- Methods (API)
  - Counter(low, high)
  - increment([amount]) ← Defaults to 1,
  - decrement([amount]) ← -1
  - setValue(value)
  - getValue()
  - getLow()
  - getHigh()

Applying the Counter Class

- To the Caesar Cipher program
- Compare implementations, with and without using the counter
- Any drawbacks from using Counter class?

Applying the Counter Class

- Creating a Clock class
  - Model the hours, minutes, seconds
  - Default: starts at 12:00:00
- Operations:
  - Ticking
  - Set the Time

Discussing the Clock Class

- Do we have to worry about user setting hours to > 12?
  - Add test

Nov 5, 2007 Sprenkle - CS111 7

Discussing the Clock Class

- Do we have to worry about user setting hours to > 12?
  - No. Counter object handles.
- Separation of functionality
  - Building code on top of other classes
  - Smaller chunk of code, well-tested that handles some set of functionality

Nov 5, 2007 Sprenkle - CS111 8

Summary: Designing Classes

- What does the object/class represent?
- How to model/represent the class's *data*?
  - Instance variable
  - Data type
- What *functionality* should objects of the class have?
  - How will others want to use the class?
  - Put into methods for others to call (API)

Nov 5, 2007 Sprenkle - CS111 9

Benefits of Classes

- Package/group related data into one object
- Reusing code
  - E.g., Don't need to check if user put in valid time
- Provide interface, can change underlying implementation
  - e.g., Counter's increment -- could implement like in Caesar Ciphers instead

Nov 5, 2007 Sprenkle - CS111 10

Considerations for using Classes

- Only use class if you're using most of its functionality/information
  - Don't use Counter for validating if a number is within the valid range; not using the wrapping/ current value
- Since don't know implementation, may inadvertently duplicate code
  - Redo something done by class
  - Could have efficiency penalties
  - But time saved reusing code is usually worth it

Nov 5, 2007 Sprenkle - CS111 11

Comparing Objects of the Same Type

- Special __cmp__ method
  - Header: _cpm__(self,other)
    - other is another object of the same type
  - Returns
    - Negative integer if self < other
    - 0 if self==other
    - Positive integer if self > other
- Similar to implementing `Comparable` interface in Java
- Can now use objects in comparison expressions
  - <,>,==, etc.

Nov 5, 2007 Sprenkle - CS111 12

Comparing Objects of the Same Type

- Example Code:

```
def __cmp__(self, other):
      """ Compares Card objects by their ranks """
      # Could compare by black jack value or rummy value
     if self.rank < other.getRank():
         return -1
     elif self.rank > other.getRank():
         return 1
     else:
          return 0
```

Lab 7 Feedback

- Good things:
  - Use of functions
  - Closing files
  - Creative pictures, animations
- Efficiency issue: Reading in whole file and saving all words in list
  - Better: line by line reading/processing