Day 28 (Mon 04/04)

- exercise 27 review
- day 28 recap questions
- exercise 28

Announcement/reminders

- Submit "Midterm Project Contributions" survey on Gradescope by Friday 4/8 (note: do not assume that late submissions will be accepted for credit)

- HW5: due Wednesday 4/6 by 11pm

- HW6: due Wednesday 4/13 by 11pm
  - written assignment, no late submissions

Exercise 27

Part 2: mean and median member functions

```
double GradeList::mean(void) {
  assert(!grades.empty());
  double sum = 0.0;
  for (std::vector<double>::const_iterator i = grades.cbegin();
       i != grades.cend();
       i++) {
    sum += *i;
  }
  return sum / grades.size();
}

double GradeList::median(void) {
  return percentile(50.0);
}
```

Part 3

in main2.cpp:

```
  double min_so_far = 100.0;
  for (size_t i = 0; i < gl.grades.size(); i++) {
    if (gl.grades[i] < min_so_far) {
      min_so_far = gl.grades[i];
    }
  }
```

This does not work because grades is a private member of GradeList, so a main function (which is not a member function of GradeList) cannot access it directly.

Part 3:

One possible solution:

In grade_list.h (adding a new member function):

```
  size_t get_num_grades() const { return grades.size(); }
  double get_grade(size_t i) const { return grades[i]; }
```

In main2.cpp:

```
  double min_so_far = 100.0;
  for (size_t i = 0; i < gl.get_num_grades(); i++) {
    if (gl.get_grade(i) < min_so_far) {
      min_so_far = gl.get_grade(i);
    }
  }
```

Part 3:

Another possible solution: add to grade_list.h (in GradeList class)

```
  const std::vector<double> &get_grades() const { return grades; }
```

The code in main2.cpp can call get_grades() on the GradeList object to access a const reference to the internal grades vector.

This doesn't violate encapsulation because a const reference can't be used to modify the internal data of the GradeList object.

Part 4:

```
#include <iostream>
#include "grade_list.h"

int main() {
  GradeList gl;
  for (int i = 0; i <= 100; i += 2) {
    gl.add(double(i));
  }
  std::cout << "Minimum: " << gl.percentile(0.0) << std::endl;
  std::cout << "Maximum: " << gl.percentile(100.0) << std::endl;
  std::cout << "Median: " << gl.median() << std::endl;
  std::cout << "Mean: " << gl.mean() << std::endl;
  std::cout << "7th percentile: " << gl.percentile(75.0) << std::endl;
}
```

Day 28 recap questions:

1. What is a non-default (or “alternative”) constructor?
2. If we define a non-default constructor, will C++ generate an implicitly defined default constructor?
3. When do we use the this keyword?
4. What is a destructor?
5. A destructor will automatically release memories that are allocated in the constructor- true or false?