Inheritance and Virtual Methods

Instructor Example

For this problem, you will design C++ classes suitable for storing information about university instructors. The goal of the example is to demonstrate arranging classes in a hierarchy for maximum code-sharing. There are three types of instructors: Faculty, Lecturer, and Grad Student. At any time, an instructor can best be described by three quantities: number of unread e-mail messages, age, and number of eccentricities. An instructor should be initialized with their age (and by setting them to have no unread mail and no eccentricities).

There are two measures of an instructor's current mood: stress and respect.

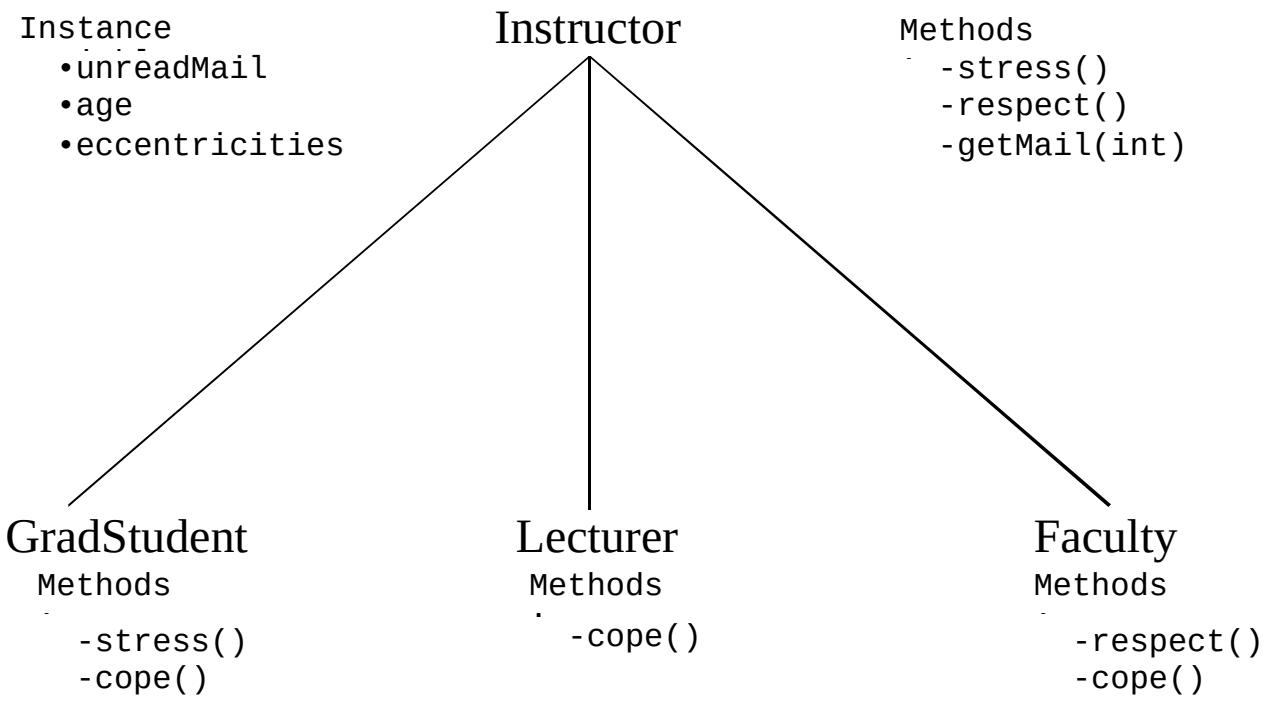
stress: an instructor's stress level is the number of unread messages. However, stress is never more than 1000. Grad students are the exception. Their stress is double the number of unread messages and their maximum stress is 2000.

respect: generally an instructor's level of respect in the community is their age minus the number of eccentricities. Respect can never be negative. Faculty are the exception— faculty eccentricities are regarded as "signs of a troubled genius" thereby increasing respect. So for faculty respect is age **plus** number of eccentricities.

Everything in an instructor's life is driven by receiving e-mail. Several things happen when an instructor gets new, unread e-mail:

- the amount of unread-mail is increased.
- 90% of the time, there will be no change in the number of eccentricities. 10% of the time the number of eccentricities will randomly go up or down by one.
- Third, the new unread mail may cause the instructor's stress factor to become larger than their respect factor, which makes the instructor unhappy. In this case, instructors have different coping mechanisms. Faculty react by gaining 10 eccentricities. Lecturers react by "accidentally" deleting half their unread mail. Grad students react by reading all of their unread mail. The resulting mental anguish causes the Grad student's eccentricities to go up or down by one randomly. The coping mechanisms may or may not bring the stress and respect factors back in line. The coping mechanism is activated at most one time for each batch of incoming mail.

This is the sort of design drawing you might make to help think about your solution:



The Code

You can also find this code on Leland at /usr/class/cs193d/section/7-20/

stress.h

```
#include <string>
#include <iostream>


class Instructor    // abstract class
{

public:
  Instructor(const std::string & aName, int anAge);
  virtual ~Instructor();  // destructor should be virtual

  virtual int stress() const;   // virtual methods shouldn't be inline
  virtual int respect() const;
  void getMail(int);
  const std::string & getName() const;

  friend std::ostream & operator<<(std::ostream & os, const Instructor &
i);

protected:
  static const int kMaxStress = 1000;

  std::string name;
  int unreadMail;
  int age;
  int eccentricities;
```

```
  // private helper which changes eccentricities by +-1
  void randomEccentric();

  virtual void cope() = 0;    // pure virtual function

private:
  static double randomPercent();
};

class GradStudent : public Instructor
{

public:
  GradStudent(const std::string & aName, int anAge)
    : Instructor(aName, anAge) {}
  virtual ~GradStudent();

  virtual int stress() const;   // must override with same arguments

protected:
  virtual void cope();
};


class Professor : public Instructor
{

public:
  Professor(const std::string & aName, int anAge)
    : Instructor(aName, anAge) {}
  virtual ~Professor();

  virtual int respect() const;

protected:
  virtual void cope();
};


class Lecturer : public Instructor
{

public:
  Lecturer(const std::string & aName, int anAge)
    : Instructor(aName, anAge) {}
  virtual ~Lecturer();

protected:
  virtual void cope();
};
```

stress.c

```
#include <cstdlib>
using namespace std;

#include "stress.h"
```

```
Instructor::Instructor(const string & aName, int anAge)
{
  name = aName;
  age = anAge;
  unreadMail = 0;
  eccentricities = 0;
}

Instructor::~Instructor()
{
}

int Instructor::stress() const
{
  if (unreadMail <= kMaxStress)
    return unreadMail;

  return kMaxStress;
}

int Instructor::respect() const
{
  int respect = age - eccentricities;

  if(respect >= 0)
    return respect;

  return 0;
}

void Instructor::getMail(int numMessages)
{
  unreadMail += numMessages;
  if(randomPercent() <= 0.10)
    randomEccentric();

  if(stress() > respect())
    cope();
}

const string & Instructor::getName() const
{
  return name;
}

void Instructor::randomEccentric()
{
  int delta;

  if(randomPercent() < 0.5)
    delta = 1;
  else
    delta = -1;

  eccentricities += delta;
```

```
  if (eccentricities < 0)
    eccentricities = 0;
}

double Instructor::randomPercent()
{
  return double(rand())/double(RAND_MAX);
}

ostream & operator<<(ostream & os, const Instructor & i)
{
  return os << i.name << " :"
    << " age = " << i.age << ";"
    << " unread mail = " << i.unreadMail << ";"
    << " ecc = " << i.eccentricities << ";"
    << " stress = " << i.stress() << ";"
    << " respect = " << i.respect();
}


GradStudent::~GradStudent()
{
  cout << getName() << " flames out of graduate school" << endl;
}

int GradStudent::stress() const
{
  return 2 * Instructor::stress();
}

void GradStudent::cope()
{
  cout << getName() << " decides to cope by reading all unread mail" <<
endl;
  unreadMail = 0;
  randomEccentric();
}


Lecturer::~Lecturer()
{
  cout << getName() << " leaves Stanford for a consulting job" << endl;
}

void Lecturer::cope()
{
  cout << getName()
    << " decides to cope by deleting half of unread mail" << endl;
  unreadMail = unreadMail / 2;
}


Professor::~Professor()
{
  cout << getName() << " gets tenure" << endl;
}

int Professor::respect() const
{
```

```
  return age + eccentricities;
}

void Professor::cope()
{
  cout << getName()
    << " decides to cope by becoming more eccentric" << endl;
  eccentricities += 10;
}
```

main.cc

```
#include <vector>
#include <iostream>
#include <cstdlib>
using namespace std;

#include "stress.h"


void
applyStress(Instructor & t, int numEmails)
{
  cout << t << endl;

  cout << "sending " << t.getName() << " " << numEmails << " emails" <<
endl;

  t.getMail(numEmails);

  cout << t << endl;

  cout << endl;
}


int
main(int argc, char *argv[])
{
  int days = 10;
  int maxMail = 40;

  if(argc > 1)
  {
    days = atoi(argv[1]);
    if(argc > 2)
      maxMail = atoi(argv[2]);
  }


  vector<Instructor *> teachers;
  // No connections to real people are implied by the names below.
  // Also, the ages are made up.
  teachers.push_back( new GradStudent("Dan", 24) );
  teachers.push_back( new Lecturer("Jerry", 31) );
  teachers.push_back( new Professor("Daphne", 33) );
```

```
  for(int i=0; i<days; ++i)
  {
    Instructor * poorsoul = teachers[rand() % teachers.size()];
    int mail = rand() % maxMail;
    applyStress(*poorsoul, mail);
  }


  for(unsigned i=0; i<teachers.size(); ++i)
    // virtual destructors ensure that this does the right thing
    delete teachers[i];
}
```

Sample Output of Program

Run with default values of `days` and `maxMail`.

```
Daphne : age = 33; unread mail = 0; ecc = 0; stress = 0; respect = 33
sending Daphne 38 emails
Daphne decides to cope by becoming more eccentric
Daphne : age = 33; unread mail = 38; ecc = 10; stress = 38; respect = 43

Jerry : age = 31; unread mail = 0; ecc = 0; stress = 0; respect = 31
sending Jerry 11 emails
Jerry : age = 31; unread mail = 11; ecc = 0; stress = 11; respect = 31

Dan : age = 24; unread mail = 0; ecc = 0; stress = 0; respect = 24
sending Dan 19 emails
Dan decides to cope by reading all unread mail
Dan : age = 24; unread mail = 0; ecc = 1; stress = 0; respect = 23

Jerry : age = 31; unread mail = 11; ecc = 0; stress = 11; respect = 31
sending Jerry 7 emails
Jerry : age = 31; unread mail = 18; ecc = 0; stress = 18; respect = 31

Dan : age = 24; unread mail = 0; ecc = 1; stress = 0; respect = 23
sending Dan 25 emails
Dan decides to cope by reading all unread mail
Dan : age = 24; unread mail = 0; ecc = 0; stress = 0; respect = 24

Dan : age = 24; unread mail = 0; ecc = 0; stress = 0; respect = 24
sending Dan 17 emails
Dan decides to cope by reading all unread mail
Dan : age = 24; unread mail = 0; ecc = 0; stress = 0; respect = 24

Jerry : age = 31; unread mail = 18; ecc = 0; stress = 18; respect = 31
sending Jerry 15 emails
Jerry decides to cope by deleting half of unread mail
Jerry : age = 31; unread mail = 16; ecc = 0; stress = 16; respect = 31

Dan : age = 24; unread mail = 0; ecc = 0; stress = 0; respect = 24
sending Dan 14 emails
Dan decides to cope by reading all unread mail
Dan : age = 24; unread mail = 0; ecc = 1; stress = 0; respect = 23

Jerry : age = 31; unread mail = 16; ecc = 0; stress = 16; respect = 31
sending Jerry 16 emails
Jerry decides to cope by deleting half of unread mail
```

```
Jerry : age = 31; unread mail = 16; ecc = 0; stress = 16; respect = 31

Daphne : age = 33; unread mail = 38; ecc = 10; stress = 38; respect = 43
sending Daphne 34 emails
Daphne decides to cope by becoming more eccentric
Daphne : age = 33; unread mail = 72; ecc = 20; stress = 72; respect = 53

Dan flames out of graduate school
Jerry leaves Stanford for a consulting job
Daphne gets tenure
```