EECS 402 Discussion 5

Project 2, More on classes (ctors/dtors)

Announcements

- P1 regrades are open (close Thursday at 7pm)

- P2 is out!! (due Oct 12)

Project 2 Overview

ColorClass

- Stores red, green, blue values as attributes
- Member functions/methods: setToRed, setToBlack, setToGreen, setToBlue, setToWhite, setTo, addColor, subtractColor, adjustBrightness, printComponentValues

RowColumnClass

- Stores rowInd and colInd as attributes
- Member functions/methods: setRowCol, setRow, setCol, getRow, getCol, addRowColTo, printRowCol

ColorImageClass

- Stores matrix of ColorClass objects (10 by 18)
- Member functions/methods: initializeTo, addImgTo, addImages, setColorAtLocation, getColorAtLocation,

Tips

- Think about adding helper functions
  - What things are you doing over and over that can be written in a function?
  - This avoids duplicate code
- Start early!
- Read the spec carefully (especially addImages in colorImageClass)

Back to classes

Scope Resolution

Used to define functions outside of the class

They must first be declared in the class

```
class Cup {
    private:
        int ounces;
        string color;

    public:
        void fill(int& addedOunces){
            ounces += addedOunces;
            addedOunces = 0;
        }

        bool isEmpty();

};

bool Cup::isEmpty(){
    return ounces == 0;
}
```

Scope Resolution

```
5   class Cup {
6       private:
7           int ounces;
8           string color;
9
10      public:
11          void fill(int& addedOunces){
12              ounces += addedOunces;
13              addedOunces = 0;
14          }
15
16          bool isEmpty(){
17              return ounces == 0;
18          }
19
20  };
```

==

```
5   class Cup {
6       private:
7           int ounces;
8           string color;
9
10      public:
11          void fill(int& addedOunces){
12              ounces += addedOunces;
13              addedOunces = 0;
14          }
15
16          bool isEmpty();
17
18  };
19
20  bool Cup::isEmpty(){
21      return ounces == 0;
22  }
```

Ctors/Dtors

Constructors

- Always called when a new instance is created, but cannot be directly called
- Name is the same as the class
- Cannot return a value
- Can be overloaded (happens in P2)
- Has a default constructor (until overloaded)

```
class Cup {
  private:
    int ounces;

  public:
    Cup(){
      ounces = 16;
    }

    Cup(int ouncesIn){
      ounces = ouncesIn;
    }

};
```

Constructors

- Always called when a new instance is created
- Can be overloaded (happens in P2)

```
int main(){
  Cup soloCup(); // initializes cup of 16 ounces

  Cup mug(10); // initializaes cup of 10 ounces
}
```

```
class Cup {
  private:
    int ounces;

  public:
    Cup(){
      ounces = 16;
    }

    Cup(int ouncesIn){
      ounces = ouncesIn;
    }

};
```

Initializer Lists

```
Cup(int ouncesIn) {
    ounces = ouncesIn;
}
```

These do (almost) the exact same thing!

- The second version is needed for:
  - **const** member variables
  - Invoking specific constructors of member objects
- Think: why might this be?

```
Cup(int ouncesIn):ounces(ouncesIn)
{ ; }
```

Copy Constructors

- Always called when a copy of a class is made

- Can be overloaded- more on this later...

```
Cup(const Cup &myCup) {
    ounces = myCup.ounces;
    cout << "copy ctor!!" << endl;
}
```

Destructors

```
// this is a class!
class BankAccount {
  private:
    double bill = 100; //represents credit card bill
    double balance = 0; // represents your balance


  public:

    ~BankAccount() {
      cout << "This is a dtor" << endl;
    }
```

- Always called when an object is destroyed
- Name is the same as the class with a ‘~’
- Cannot return a value
- More on this when we get to dynamic memory

Example

CPP file:
https://drive.google.com/file/d/1jLrtwu8ZjOfFKK8oBi986aMzazXaSVdh/view?usp=sharing

Git:
$ git clone https://github.com/emolson16/oop-example.git

```cpp
// this is a class!
class BankAccount {
  private:
    double bill; //represents credit card bill
    double balance; // represents your balance

  public:

    // initializes bank account to 0
    BankAccount() {
      bill = 0;
      balance = 0;
    }

    // TODO: initializes bill to 0 and balance to initialAmount
    BankAccount(int initialAmount) {}

    //TODO: deposit the given amount into your balance
    void deposit(double amount) {}

    void charge(double amount) {
      bill += amount;
    }

    //TODO if you have enough money, withdraw the given amount and return ture
    // if you don't have enough money, just return false
    bool withdraw(double amount) {}

    //TODOL Pay your credit card bill (you can go into debt here)
    // try to do this one using scope resolution!
    void payBill() {}

    // Challenge problem- don't worry if you can't get it yet
    // TODO pay your freind the given amount to their account
    // You cannot go into debt here (return false if you don't have enough money)
    bool payFriend(BankAccount& friendAccount, double amount) {}

    // prints current balance
    void printBalance(){
      cout << "Current balance is: $" << balance << endl;
    }

    ~BankAccount() {
      cout << "dtor!!" << endl;
    }

};
```

Example Solution

```cpp
// this is a class!
class BankAccount {
  private:
    double bill; //represents credit card bill
    double balance; // represents your balance

  public:

    // initializes bank account to 0
    BankAccount() {
      bill = 0;
      balance = 0;
    }

    // TODO: initializes bill to 0 and balance to initialAmount
    BankAccount(int initialAmount) {
      bill = 0;
      balance = initialAmount;
    }

    //TODO: deposit the given amount into your balance
    void deposit(double amount) {
      balance += amount;
    }

    void charge(double amount) {
      bill += amount;
    }

    //TODO if you have enough money, withdraw the given amount and return ture
    // if you don't have enough money, just return false
    bool withdraw(double amount) {

      if(balance >= amount) {
        balance -= amount;
        return true;
      }

      return false;
    }
```

```cpp
    //TODOL Pay your credit card bill (you can go into debt here)
    void payBill();

    // Challenge problem- don't worry if you can't get it yet
    // TODO pay your freind the given amount to their account
    // You cannot go into debt here (return false if you don't have enough money)
    bool payFriend(BankAccount& friendAccount, double amount) {

      if(balance >= amount) {
        balance -= amount;
        friendAccount.deposit(amount);
      }

      return false;
    }

    // prints current balance
    void printBalance(){
      cout << "Current balance is: $" << balance << endl;
    }

    ~BankAccount() {
      cout << "dtor!!" << endl;
    }

};
```

```cpp
void BankAccount::payBill() {
    balance -= bill;
    bill = 0;
}
```