CS 103 Unit 14

Classes Revisited

Mark Redekopp
UML (Unified Modeling Language)

• Shows class definitions in a language-agnostic way
• Shows class hierarchy (inheritance, etc.)
• Each class shown in one box with 3 sections
  – Class Name, Member functions, then Data members
  – Precede function/data member with:
    + (public), - (private), # (protected)
  – Functions show name with arguments : return type
  – Data members show name : type

```
class Deck {
    public:
    Deck();   // Constructor
    ~Deck();  // Destructor
    void shuffle();
    void cut();
    int get_top_card();

    private:
    int cards[52];
    int top_index;
};
```

```
class name (e.g. Deck)
+ shuffle() : void
+ cut() : void
+ get_top_card() : int

Member data
- cards[52] : int
- top_index : int
```
• Remember data members live on from one member function call to the next and can be accessed within ANY member function.

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

class ABC
{
public:
    ABC();
    void add_score(int s);
    int get_score(int loc);
private:
    vector<int> scores;
};

// A change to scores here
void ABC::add_score(int s){
    scores.push_back(s);
}

// would be seen by subsequent // calls to member functions
int ABC::get_score(int loc){
    return scores[loc];
}

int main(){
    ABC a;
    a.add_score(95);
    a.get_score(0);
}
```
Class Design

• Class names should be 'nouns'

• To decide what objects/classes you need use
  – Object discovery: Based on the requirements of description of the problem look for the nouns/object
  – Object invention: Objects that simplify management or help glue together the primary objects

• Method/Function names should be 'verbs'

```cpp
class GradeBook {
    public:
        computeAverage();
        int* getScores();
    private:
        int scores[20];
        int _size, _tail;
};

bool GradeBook::computeAverage() {
    double sum = 0.0;
    for (int i=0; i < _size; i++) {
        sum += scores[i];
    }
    return sum / _size;
}

int main() {
    GradeBook gb;
    int* myscores = gb.getScores();
    double sum = 0.0;
    for (int i=0; i < _size; i++) {
        sum += myscores[i];
    }
    ...
}
```

http://www.cprogramming.com/tutorial/class_design.html
Class Design

- Keep the computation where the data is (i.e. in the appropriate class member functions)

```cpp
class GradeBook
{
    public:
        computeAverage();
        int* getScores();
        int size() { return _size; }
    private:
        int scores[20];
        int _size, _tail;
};

bool GradeBook::computeAverage()
{
    double sum = 0.0;
    for(int i=0; i < _size; i++)
    {
        sum += scores[i];
    }
    return sum / _size;
}

int main()
{
    GradeBook gb;
    int* myscores = gb.getScores();
    double sum = 0.0;
    for(int i=0; i < gb.size(); i++)
    {
        sum += myscores[i];
    }
    ...
}
```
Group Activity

• Write the class definitions (and maybe eventually the whole program) for a card game
  – Form groups of 2 or 3
  – Choose BlackJack unless you'd really like to do something else
Part 1

• Write out the rules in Word, Google Docs, etc.
  – Put a box around the nouns...
  – Circle the action verbs
• Use the nouns and verbs to define the classes and member functions at a general level