Announcements

- HW2 is posted
- Get a copy of Ch. 2
- Textbook will come out this week, I'll post details
- HW1 is graded. Expect an email w/ your grade in next 1-2 days.
- Lab tomorrow - prelab is due by 10am
A word on cheating.

- Do **NOT** look at another student's code.
- Do **NOT** look at webpages for solutions.

(I only allow course materials plus plus plus. com.)
More on Classes:

Destructors:

If your class opens files or allocates memory, then can't just use delete.

Must create a destructor:

~ClassName() - no arguments, no return type

~Point()  

delete &commands
Copy Constructors:

Previously:
Point a;
Point b(a);

Consider the following:
Vect a (100);
Vect b (a);
What does this do?

(in C++, by default)

copies each element of a to b:

\[
\text{vectSize} = a.\text{vectSize};
\]

\[
\text{theVect} = a.\text{theVect};
\]

\[
\begin{array}{c}
\text{a: Vect} \\
\text{vectSize = 100} \\
\text{theVect}
\end{array}
\]

\[
\begin{array}{c}
\text{b: Vect} \\
\text{vectSize = 100} \\
\text{theVect}
\end{array}
\]

Shallow copy!

not good - so we'll override this
To fix, write our own copy constructor:

```cpp
Vect b(a);

// copy constructor
 Vect(const Vect &a) &
{ vectSize = a.vectSize; // copy size

    Vect = new int[vectSize]; // new vector

    for (int i = 0; i < vectSize; i++)
        Vect[i] = a.Vect[i];
}
```
Another problem:

\[
\text{Vect a (100);} \\
\text{Vect c;} \\
\text{c = a;} \\
\]

What does this do? Shallow copy by default, copies each parameter:
\[
\text{c.vectSize = a.vectSize;} \\
\text{c.theVect = a.theVect;} \\
\]

Write operator = to make deep copy of data.

\[\text{3rd housekeeping function}\]
Enum: user defined types

enum Color {RED, BLUE, GREEN};

Color sky = BLUE;
Color grass = GREEN;

Convention: write in all capital letters
(not enforced by compiler)
Structures: (legacy from C)
useful for holding collections of objects

Ex:
enum MealType {NO_PREF, REGULAR, VEG};

struct Passenger {
  string name;
  MealType mealPref;
  bool isFreqFlyer;
  String FreqFlyerNo;
};

Could use a class.
No functions, no public/private — stripped down class.
Using Structures

Structures can then be used inside the program:

```
Passenger pass = { "John Smith", VEG, true, "12345" };
pass.mealPref = REGULAR;
```
Another example:

```java
Passenger *p; // can't say p. name = "Trin"

p = new Passenger;
    p -> name = "Barbara Wright";
    p -> mealPref = NO_PREFERENCE;
    p -> isFrequentFlyer = false;
    p -> frequentFlyerNo = "NONE";

(*p). mealPref = VEG;
```

```java
if (p -> mealPref == REGULAR) // test if meal preference is regular
```
Function templates:

```
template <typename T>
T min(T a, T b) {
  if (a < b)
    return a;
  else
    return b;
}
```

Important: Will work for any class, as long as "<" has been defined!
Example:

```cpp
int x = 53;
int y = 69;
int z = \min(x, y);
```

```cpp
string a = "Hello";
string b = "Goodbye";
```

```cpp
cout << \min(a, b) << endl;
Goodbye (!)
```
Class templates: a vector example

```cpp
template <typename T>
class BasicVector {
private:
    T* a;
    size_t capacity; // length of array a

public:
    BasicVector(int c = 10) : capacity(c) { // constructor
        a = new T[capacity]; // allocate storage
    }

    T& elementAtRank(int r) { // access r^th element
        return a[r];
    }
};
```
Back to BasicVector: usage

Basic Vector <int> intvec(5); //vector of 5 ints
Basic Vector <string> strvec(10); //vector of 10 strings

intvec.elementAtRank(3) = 8; //sets 4th element = 8
strvec.elementAtRank(7) = "hello"; //sets 8th elt = "hello"

Or even:

Basic Vector < Basic Vector <int> > myvec(5); //vector of 5 Basic Vectors of integers

myvec.elementAtRank(2).elementAtRank(8) = 15; // myvec[2][8] = 15
Error Handling

In C++, we do error handling by throwing exceptions.

(These are really just classes themselves.)

What exceptions do we have in Python?

- TypeError
- NameError
- SyntaxError
- ValueError

3 classes
Exceptions in C++

The book uses its own error classes. (at end of Ch2)
Most of mine are based on C++
default exceptions.

So:

#include <stdexcept> ← (at top of file)
Example:

In BasicVector, might want to allow you to access the i-th element:

```cpp
operator[](int index) {
    if (index >= capacity || index < 0)
        throw out_of_range("Index out of range");
    return a[index];
}
```
Basic Vect <int> myvec;

// fill in vector

try {
    cout << myvec[73] << endl;
}

catch {
    next time
}