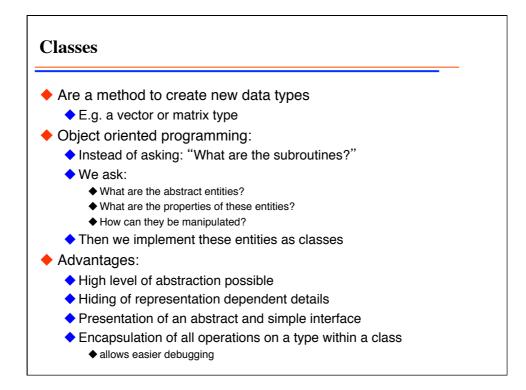
An Introduction to C++

Part 4

Introduction to classes



A first simulation: biological aging
a simple model for death: dN = -λNdt
what about aging?

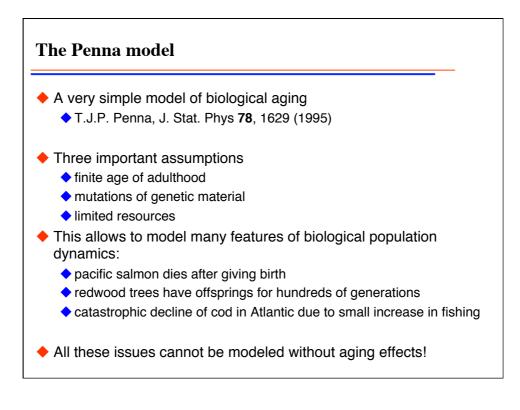
the remaining lifetime does not depend on current age.
true for radioactive decay
not true for biology

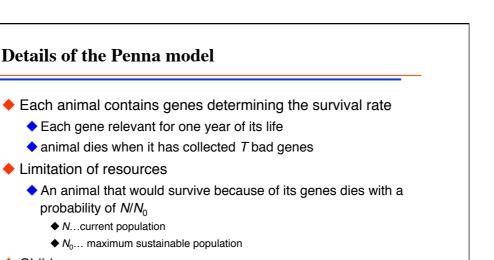
what about age distribution?

exponential distribution!
also not what is seen in nature!

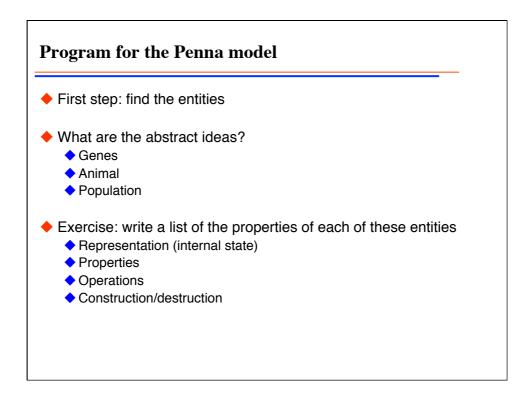
what is missing?

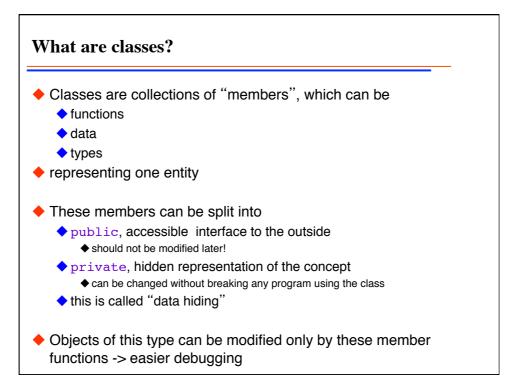
some kind of aging
we need to develop a model containing aging

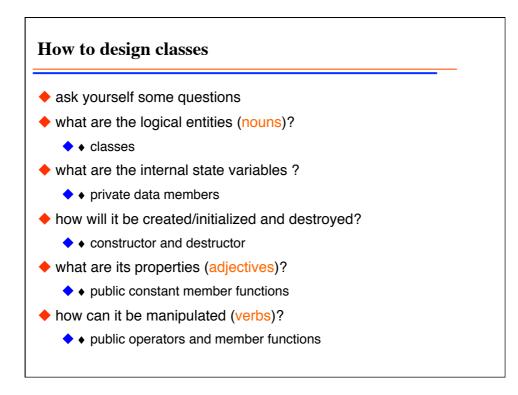


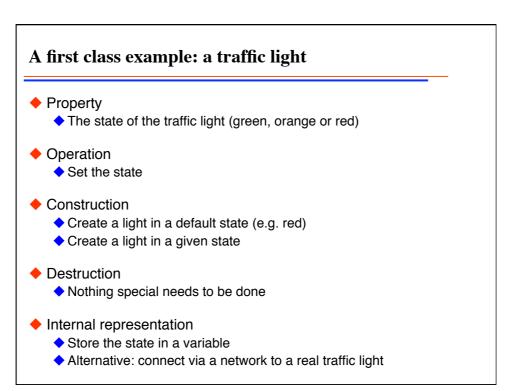


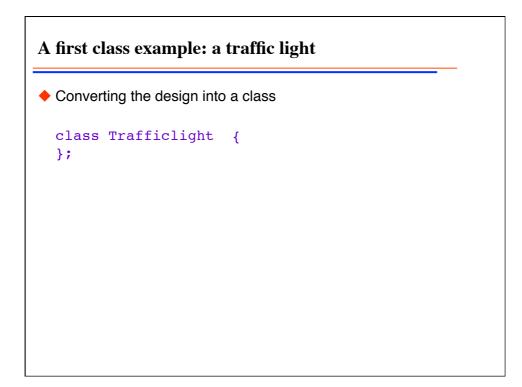
- Children
 - from an age of *R* years an animal gets a child asexually with a probability *b* (birthrate)
- Mutations
 - The children have the genes of the parents but with *M* random mutations

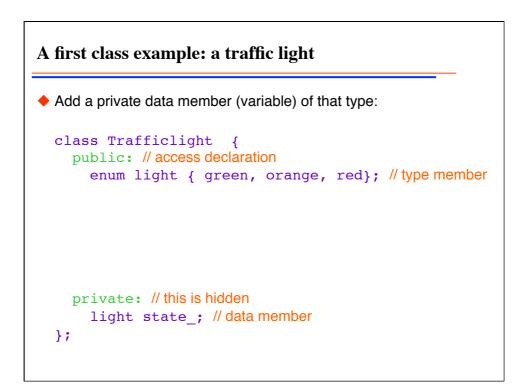




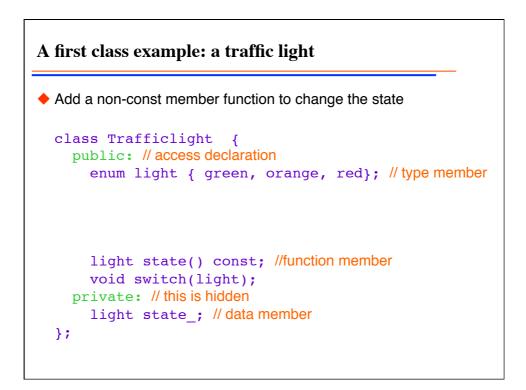




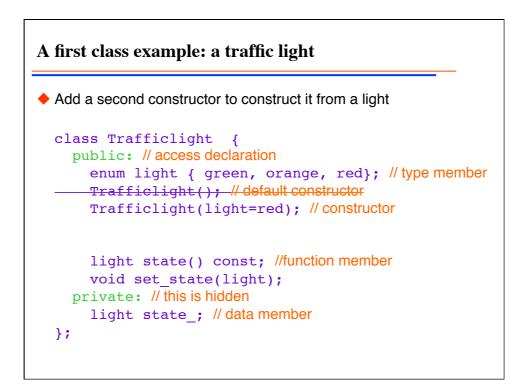




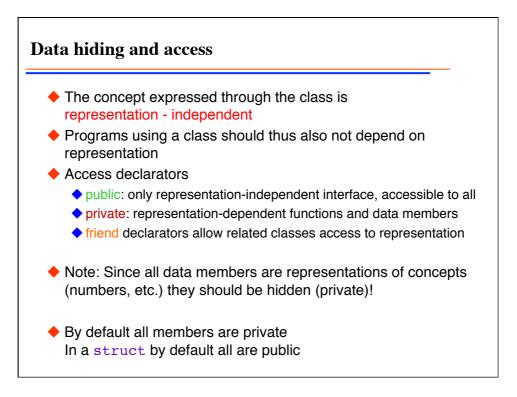
A first class example: a traffic light
Add a const member function to access the state
class Trafficlight {
 public: // access declaration
 enum light { green, orange, red}; // type member
 light state() const; //function member
 private: // this is hidden
 light state_; // data member
};

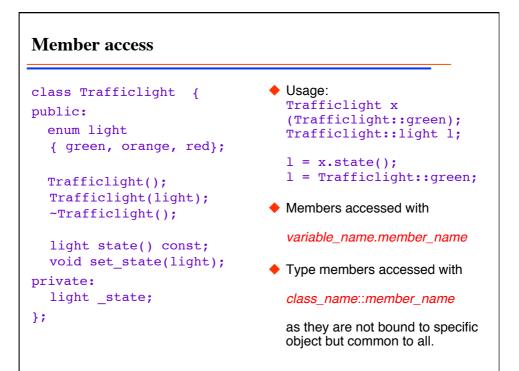


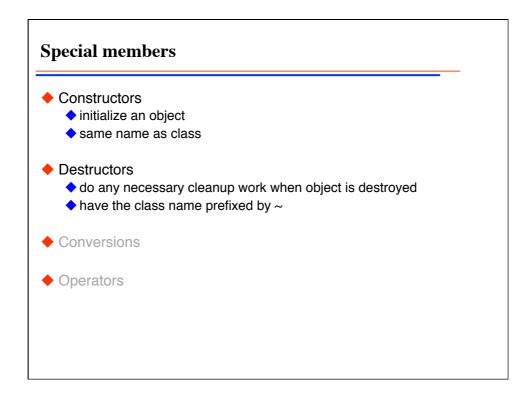
A first class example: a traffic light
Add a default constructor to initialize it in the default way a constructor has the same name as the class class Trafficlight { public: // access declaration enum light { green, orange, red}; // type member Trafficlight(); // default constructor
light state() const; //function member void set_state(light); private: // this is hidden light state_; // data member };

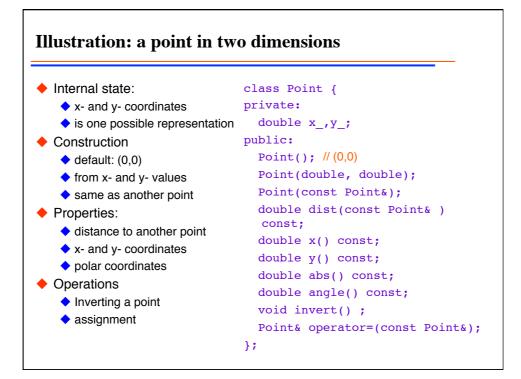


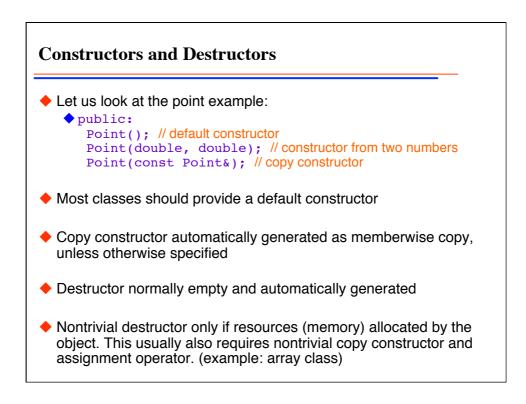
A first class example: a traffic light
And finish by adding a destructor (called to cleanup at destruction) a destructor has the same name as the class, prefixed by ~ class Trafficlight { public: // access declaration enum light { green, orange, red}; // type member Trafficlight(); // default constructor Trafficlight(light); // constructor ~ Trafficlight(light); // constructor ~ Trafficlight(); // destructor
light state() const; //function member void set_state(light); private: // this is hidden light state_; // data member };

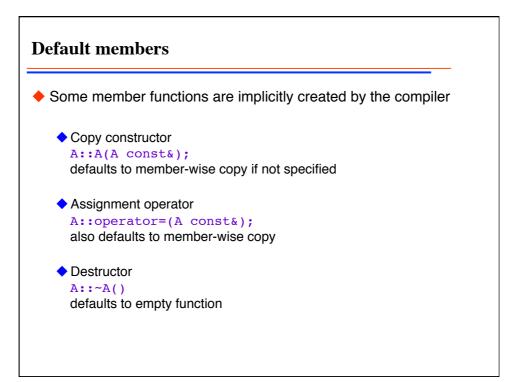


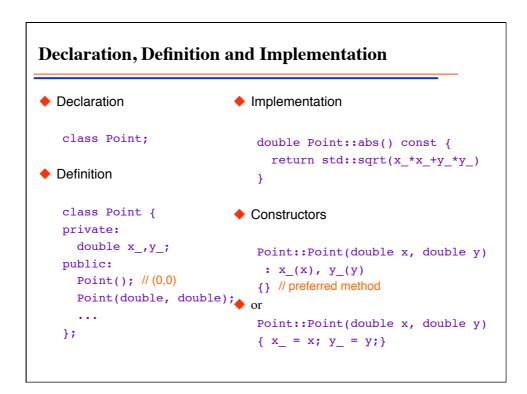


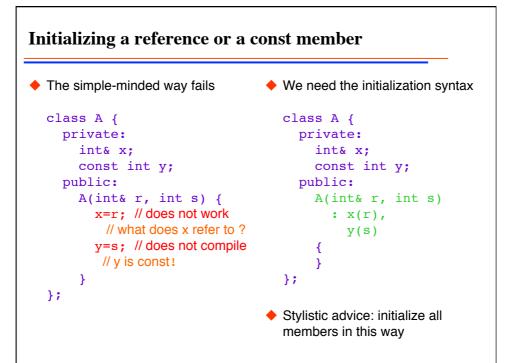


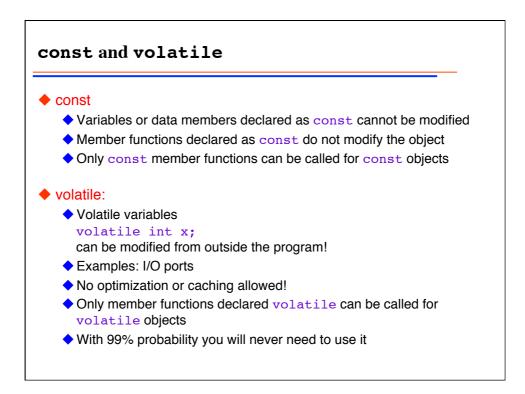








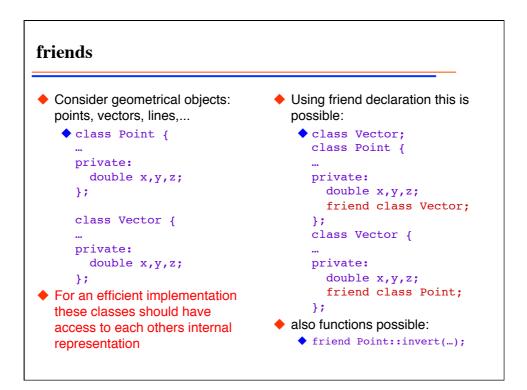




```
mutable
                                   • Solution:
Problem:
    want to count number of calls to age

    mutable qualifier allows

      () function of animal
                                         modification of member even in
                                         const object!
Original source:
                                   Modified source:
   class Animal() {
                                      class Animal() {
   public:
      age_t age() const;
                                         ....
                                       private:
   private:
                                        mutable long cnt_;
      long
              cnt_;
      age_t age_;
                                         ....
                                       };
   };
                                       age t Animal::age() const {
   age_t Animal::age() const {
                                         cnt_++; // now OK!
      cnt ++; // error: const!
                                         return age_;
      return age ;
                                       }
   }
```



Week 4

