

CS 249A  
Review Session  
Assignment 1

Anant Bhardwaj

[anantb@cs.stanford.edu](mailto:anantb@cs.stanford.edu)

# Course Info

- Website: <http://cs249a.stanford.edu>
- Staff Email: send questions to this address!  
[cs249a-staff@cs.stanford.edu](mailto:cs249a-staff@cs.stanford.edu)
- Piazza: to interact with TAs and classmates  
<http://piazza.com/stanford/fall2011/cs249a>

# Office Hours

- Prof. Cheriton (Gates 439)  
by appointment
- Anant Bhardwaj (Gates B26A)  
Tue and Thu 6:30-8:30pm
- Aditya Ramesh (Gates B24A)  
Wed 6:15-8:15pm, Fri 10am-12pm
- Use Stanford ID to access basement after 5pm

# Overview

- Biological viral infection simulation
- Tissues, cells, cell membranes
- Infection spreading in tissue
- Write simulation using given interface

# Entities

- Tissue
  - Collection of cells located in  $(x, y, z)$ ,
  - where  $-50 \leq x, y, z \leq 50$  are integers
- Cell
  - Collection of (six) cell membranes
  - At most six neighbors (north  $+y$ , south  $-y$ , east  $+x$ , west  $-x$ , up  $+z$ , down  $-z$ )
- Cell membrane
  - Has antibodies to fight against infection

# Infection Model

- Starts by attacking a cell membrane of a particular cell with given infection strength
- Spreads to neighboring cells using **Breadth-First Search (BFS)**
- If membrane has antibody strength higher than infection strength, infection fails
- Does **not attempt** to infect already infected cells
- Stops when no more attempts needed

# Program Input/Output

- Input - text file with commands
  - Tissue tissueNew Tissue1
  - Tissue Tissue1 helperCellNew x y z
  - Cell Tissue1 x y z cloneNew north
  - Tissue Tissue1 infectionStartLocations x y z east
- Output for each round of infection, # infected cells, # infection attempts, etc. (see the webpage for details)
- Make sure exceptions do not propagate back!

# Concepts

- Attribute-oriented interface design
  - Data Types (ADT)
  - Name (Noun vs. Verb)
  - Operations
    - Accessors
    - Mutators
- Notifications Model

# Attribute-oriented Interface Design

- Client-facing (public) Interface of an object
- Clients are only concerned with attributes of an object
- Tell the object what you want its state to be, not how to reach that state - leave that to object's implementation
- e.g. `dictionary->sortOrderIs(Order::Ascending())`

# Attribute

- Attribute Examples

- color attribute of a car

- Color c = car->color(); //accessor**

- car->colorIs(Color::Red()); //mutator**

- Passenger: a collection attribute of an airplane

- Passenger::Ptr p = plane->passenger("Jack");**

- plane->passengerIs("Anant", seat\_);**

# Attribute Examples

- **Accessors**

```
Cell::Ptr cell = tissue->cell(coords);  
CellMembrane::Ptr mem = cell->membrane(CellMembrane::north());  
U32 antibodyStrength = mem->antibodyStrength();
```

- **Mutators**

```
Tissue::Ptr tissue = Tissue::TissueNew("myTissue");  
Cell::Ptr cell = Cell::CellNew(coord);  
cell->membraneNew("mem_n", CellMembrane::north());  
mem->antibodyStrengthIs(U32(40));  
tissue->cells(cell);  
tissue->cellDel(cell->name());
```

# Notification Model

```
class Account : public NamedInterface {  
public:  
    class Notiffee;  
    Dollar balance() const { return balance_; }  
    void balancel( Dollar );  
    Account::Notiffee * notiffee() const { return notiffee_; }  
protected:  
    Dollar balance_;  
    Account::Notiffee *notiffee_;  
    Void notiffeeels (Account::Notiffee);  
};
```

# Notification Model

```
class Account::Notiffee : public NamedInterface::Notiffee {  
public:  
    virtual void onBalance() {}  
    Notiffee( Account * a ) : NamedInterface::Notiffee( a ) {}  
};
```

```
void Account::notifieels( Account::Notiffee * n ) const {  
    Account* me = const_cast<Account*>(this);  
    me->notiffee_ = n;  
}
```

# Notification Model

```
void Account::balancel( Dollar newBalance ) {  
    if( newBalance == balance_ ) {  
        return;  
    }  
    balance_ = newBalance;  
    if( notifiee_ ) {  
        try {  
            notifiee_->onBalance();  
        }catch(...) {  
        }  
    }  
}
```

# Notification Model

```
class AccountReactor : public Account::Notiffee {  
public:  
    void onBalance() {  
        // handle changes to balance  
    }  
    static AccountReactor::Ptr AccountReactorIs( Account* a ) {  
        AccountReactor *m = new AccountReactor(a);  
        return m;  
    }  
protected:  
    AccountReactor( Account* a ):Account::Notiffee(a) {}  
}
```

# Notification Model

## Client Code

```
Account::Ptr acc = Account::AccountNew("anant")
AccountReactor* ar = AccountReactor::AccountReactorIs(acc.ptr());
acc->balancel(Dollar(5000))
```

## For Tissue Case

```
Tissue::Ptr tissue = Tissue::TissueNew("myTissue");
TissueReactor *tr = TissueReactor::TissueReactorIs(tissue.ptr());
Cell::Coordinates coords = Cell::Coordinates();
tissue->cells(Cell::CellNew(coords, tissue.ptr(), Cell::tCell()));
Cell::Ptr cell = tissue->cell(coords);
tissue->cellDel(cell->name());
```

# Requirements and tips

- Must not have extra output from exceptions
- Must use notifications for
  - constructing cell membranes upon adding new cell to tissue
  - keeping count of # helper cells and # cytotoxic cells in a tissue
- Must not modify any provided types
- Derive from `Tissue::Notifiee` for custom reactor to changes in tissue
- Make sure program runs on corn machines!

# Questions

- Post questions on piazza
- Come to office hours
- Send us an email