Assignment 1

- Biological viral infection simulation
  - Tissues, cells, cell membranes
  - Infection spreading in tissue
- Write simulation using given interface
- Examples of
  - Attribute-oriented interface design
  - Value types
  - Notifications

4

Object Model

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

5

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

6

Input / Output

- Input - text file with commands

```
Tissue tissueNew Tissue1
Tissue Tissue1 helpCellNew x y z
Cell Tissue1 x y z cloneNew north
Tissue Tissue1 infectionStartLocationIs x y z east
```

- Output for each round of infection, # infected cells, # infection attempts, etc.
- Make sure exceptions do not propagate back to the shell - extra output regarded as wrong!

7

Attribute-oriented interface

- Client-facing (public) interface of an object
- Client are only concerned with attributes of an object
- Declarative semantics
  - 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(X)`

8

Attribute Examples

- **`color`** attribute of a car

```
Color c = car->color();
car->colorIs(Color::Red());
```

- **`passenger`** collection attribute of an airplane

```
Passenger::Ptr p = plane->passenger("Jack");
plane->passengerIs("Tony", q);
```


9

Attribute Examples

- Accessors

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

- Mutators

```
mem->antibodyStrengthIs(AntibodyStrength(40));
tissue->cellIs(cell); // indexed by cell location
cell->membraneNew("north",
                  CellMembrane::SideInstance("north"));
```

- Object instantiation
  - Factory methods, Mutators

```
static Tissue::Ptr TissueNew(Fwk::String _name);
```


10

Value types

- AntibodyStrength - wrapper for a U32
  - Range-checking during construction
  - Enable type checking by compiler
  - Operators: ==, !=, <, >, <=, >=, <<
  - Source-code representation of valid range of AntibodyStrength values

11

Notifications

- Object `foo` (notifiee) of type `Foo` interested in changes to object `bar` (notifier) of type `Bar`
- `foo` registers with `bar` as notifiee
- Changes to `bar` using `bar->attrIs()`, which calls `foo->onAttr()`
- `Bar::Notifiee` interface
  - Defines virtual `onAttr()` for each notifying attribute of `Bar`
  - Inherited by `Foo` which overrides `onAttr()`

12

Notifications Example

```
 1   class Account {
 2   public:
 3      class Notifiee;
 4      U32 balance() const { return balance_; }
 5      void balanceIs( U32 newBalance );
 6      void notifieeIs( Account::Notifiee *n ) {
 7         notifiee_ = n;
 8      }
 9   protected:
10      U32 balance_;
11      Account::Notifiee *notifiee_;
12   };
13
14   class Account::Notifiee {
15   public:
16      virtual void onBalance() = 0;
17      Notifiee( Account * a ):account_(a) {}
18   protected:
19      Account *account_;
20   };
```


13

```
 1   void Account::balanceIs( U32 newBalance ) {
 2      if( balance_ == newBalance ) return;
 3      balance_ = newBalance;
 4      if( notifiee_ ) try {
 5        notifiee_->onBalance();
 6      }
 7      catch(...) { /* Exception processing*/ }
 8   }
 9
10   class AccountReactor : public Account::Notifiee {
11   public:
12      void onBalance() {
13         // handle changes to balance
14      }
15      static AccountReactor::Ptr
16      AccountReactorIs( Account* a ) {
17         AccountReactor *m = new AcccountReactor(a);
18         return m;
19      }
20   protected:
21      AccountReactor( Account* a ):
22                  Account::Notifiee(a) {}
23   }
```


14

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 pod machines!

15