Zombie Apocalypse

By Lotanna Akukwe, Sophie Byrne, and Brodie Gibson

About our Project

- We created a zombie apocalypse simulation that tracks the spread of the zombie infection
- The four districts in Simville: The Burbs, Downtown, Medical Hill, & The University
- There are four types of Citizens: Ignorant, Alarmed, Zombies, and Doctors
  - 2000 total
- Each Time Click:
  - Each citizen has a chance to move to an adjacent district and take an action against another citizen
  - At the end of each time click the number of zombies, ignorant, alarmed, and doctors are re-evaluated

Coolness Factors:

- Added Doctor class that can heal Zombies
- Infection rate dependent on Age
- Survival rate dependent on type of Citizen

Important Methods/Functions

- populate()
- createDistricts()
- subDivide()
- setQueue()
- getInfectionRate() & getSurvivalRate()
- moveCitizen()
- action() [for Zombie]

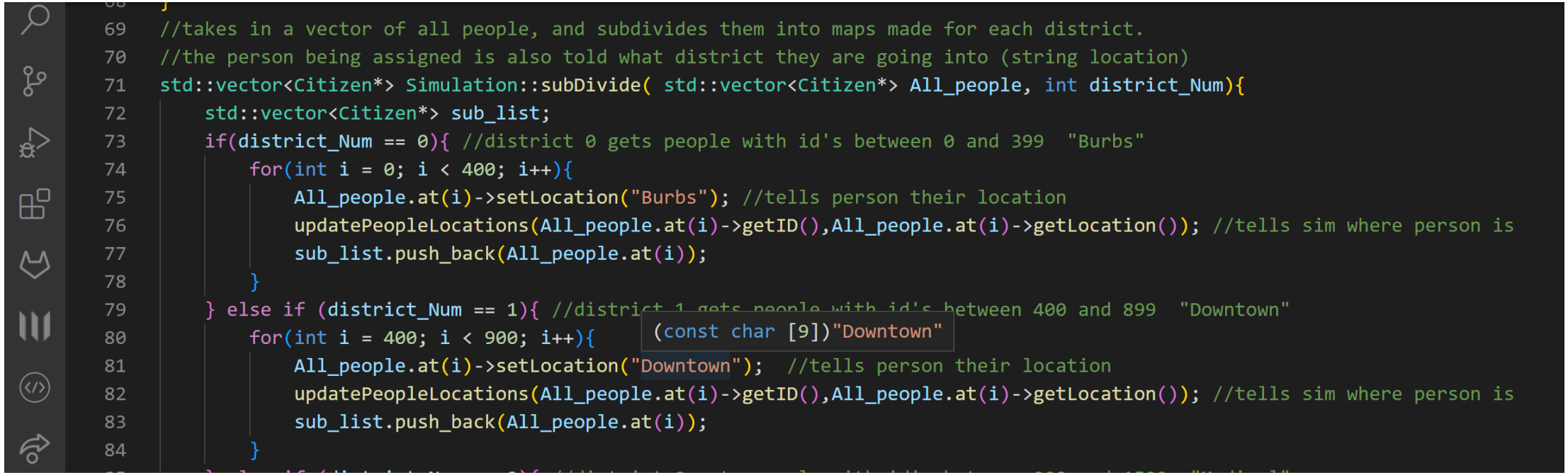
populate()

```
159  void Simulation::populate(string file_name){
160      std::ifstream filepath;
161      filepath.open(file_name);
162      string name;// holds name of person
163      int age; //holds age of person
164      for(int i = 0; i < 2000; i++){
165          filepath >> name; //reads in name from file
166          filepath >> age; //reads in age from file
167          if(i>900 && i<910){
168              Citizen* temp2 = new Doctor(name,age); //makes a citizen* and points it to a new ignorant
169              allCitizens.push_back(temp2); // add citiZen to member variable vector
170          }
171          else if(i>911 && i<915){
172              Citizen* temp2 = new Alarmed(name,age); //makes a citizen* and points it to a new ignorant
173              allCitizens.push_back(temp2); // add citiZen to member variable vector
174          }
175          else{
176              Citizen* temp2 = new Ignorant(name,age); //makes a citizen* and points it to a new ignorant
177              allCitizens.push_back(temp2); // add citiZen to member variable vector
178          }
179          //we may need to delete temp2 here to prevent a data leak, but im not sure
180      }
181      return;
182  }
183
```

createDistricts()

```
59  void Simulation::createDistricts(){
60      std::multimap<int,Citizen*> tempMap;
61      string names[4]{"Burbs","Downtown","Medical","College"}; //0 Burbs, 1 Downtown, 2 Medical, 3 College
62
63      for(int i = 0; i <4; i++){ //loop to subdivide totalpeople into district vectors to be made into maps
64          tempMap = make_maps(subDivide(allCitizens,i));
65          District temp {names[i], tempMap.size(), tempMap};
66          districts.push_back(temp);
67      }
68  }
```

subDivide()



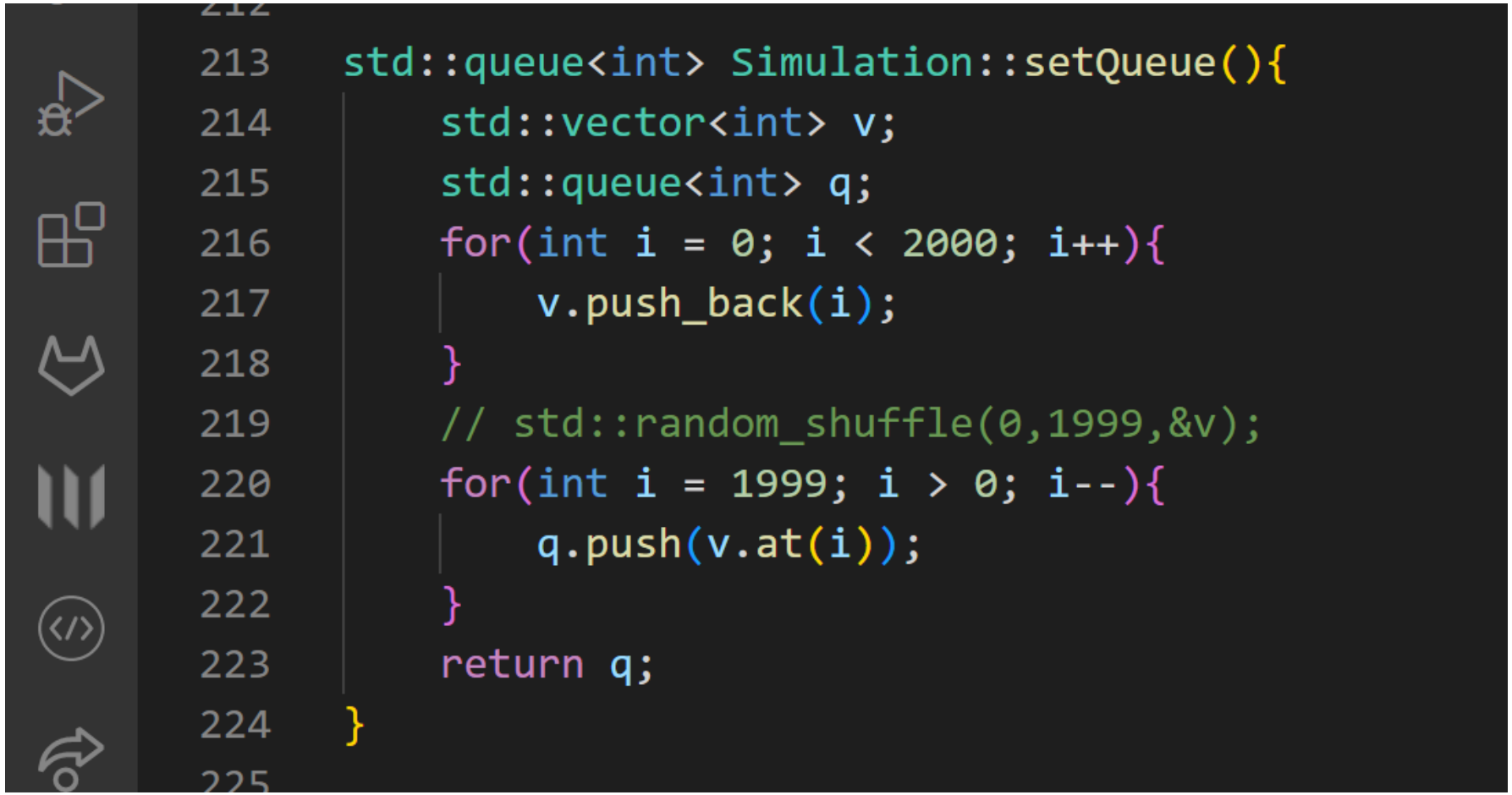
```
69  //takes in a vector of all people, and subdivides them into maps made for each district.
70  //the person being assigned is also told what district they are going into (string location)
71  std::vector<Citizen*> Simulation::subDivide( std::vector<Citizen*> All_people, int district_Num){
72      std::vector<Citizen*> sub_list;
73      if(district_Num == 0){ //district 0 gets people with id's between 0 and 399  "Burbs"
74          for(int i = 0; i < 400; i++){
75              All_people.at(i)->setLocation("Burbs"); //tells person their location
76              updatePeopleLocations(All_people.at(i)->getID(),All_people.at(i)->getLocation()); //tells sim where person is
77              sub_list.push_back(All_people.at(i));
78          }
79      } else if (district_Num == 1){ //district 1 gets people with id's between 400 and 899  "Downtown"
80          for(int i = 400; i < 900; i++){
81              All_people.at(i)->setLocation("Downtown");  //tells person their location
82              updatePeopleLocations(All_people.at(i)->getID(),All_people.at(i)->getLocation()); //tells sim where person is
83              sub_list.push_back(All_people.at(i));
84          }
```

(const char [9])"Downtown"

moveCitizen()

```
252  //d represents district citizen is moving from, c represents the citizen themselves
253  void Simulation::moveCitizen(District d, Citizen*& c){
254      std::vector<string> s = dist_access.at(c->getLocation()); //the vector of available locations
255      string temp;
256      if(c->getID()%2 == 0){
257          temp = s.at(0);
258      }else{
259          temp = s.at(1);
260      }
261      //District newD =
262      matchDistrict(temp).addCitizen(c,c->getID()); //adds citizen to new map
263      c->setLocation(temp);
264      updatePeopleLocations(c->getID(), temp);
265      d.deleteCitizen((c->getID())); //removes citizen from old map
266      return;
267  }
268
```

setQueue()



```
213  std::queue<int> Simulation::setQueue(){
214      std::vector<int> v;
215      std::queue<int> q;
216      for(int i = 0; i < 2000; i++){
217          v.push_back(i);
218      }
219      // std::random_shuffle(0,1999,&v);
220      for(int i = 1999; i > 0; i--){
221          q.push(v.at(i));
222      }
223      return q;
224  }
```

getInfectionRate() & getSurvivalRate()

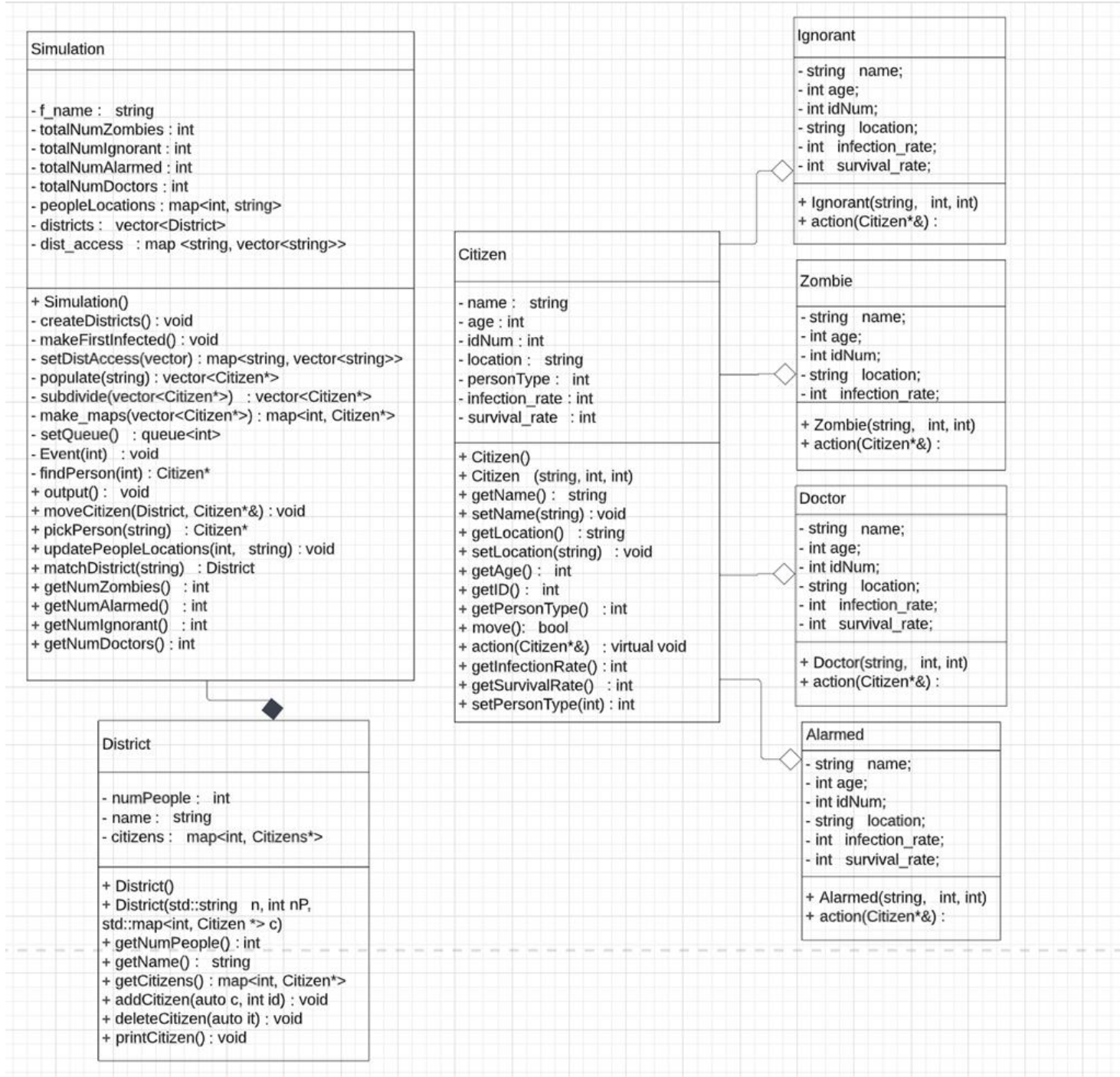
```
///chance of infecting/effecting others
int Citizen::getInfectionRate(){
    if((this->getAge() > 50)||(this->getAge()< 14)){
        //make lower chance of infection
        return -20;
    }else{
        //make higher chance of infection
        return 20;
    }
}

//chance of surviving/resisting a event
int Citizen::getSurvivalRate(){
    if(this->personType == 1){ //ignorant survival rate
        return 0;
    } else if(this->personType == 2){ //alarmed survival rate
        return 40;
    } else if(this->personType == 4){ //doctor survival rate
        return 35;
    } else if(this->personType == 3){ //zombie survival rate
        return 0;
        //against doctor
    }
}
```

action() for Zombie

```
87      void action(Citizen*& person2){ // action for zombie is infecting another person
88          std::cout << "RAUUUGH" << std::endl;
89          if(person2->getPersonType() == 4 || person2->getPersonType() == 2 || person2->getPersonType() == 1){
90              // use infection rate and survival rate to see chances of infection
91              int totalChance = person2->getInfectionRate() + person2->getSurvivalRate();
92              if(rand()%100-1 <= totalChance){
93                  std::string name = person2->getName();
94                  int age = person2->getAge();
95                  int ID = person2->getID();
96                  delete person2;
97                  person2 = new Zombie(name, age, ID);
98                  person2->setLocation(this->getLocation());
99              }
100         }
101     }
102 };
103
```

UML Diagrams


Simulation
- f_name : string
- totalNumZombies : int
- totalNumIgnorant : int
- totalNumAlarmed : int
- totalNumDoctors : int
- peopleLocations : map<int, string>
- districts : vector<District>
- dist_access : map <string, vector<string>>
+ Simulation()
- createDistricts() : void
- makeFirstInfected() : void
- setDistAccess(vector) : map<string, vector<string>>
- populate(string) : vector<Citizen*>
- subdivide(vector<Citizen*>) : vector<Citizen*>
- make_maps(vector<Citizen*>) : map<int, Citizen*>
- setQueue() : queue<int>
- Event(int) : void
- findPerson(int) : Citizen*
+ output() : void
+ moveCitizen(District, Citizen*&) : void
+ pickPerson(string) : Citizen*
+ updatePeopleLocations(int, string) : void
+ matchDistrict(string) : District
+ getNumZombies() : int
+ getNumAlarmed() : int
+ getNumIgnorant() : int
+ getNumDoctors() : int
District
- numPeople : int
- name : string
- citizens : map<int, Citizens*>
+ District()
+ District(std::string n, int nP,
std::map<int, Citizen *> c)
+ getNumPeople() : int
+ getName() : string
+ getCitizens() : map<int, Citizen*>
+ addCitizen(auto c, int id) : void
+ deleteCitizen(auto it) : void
+ printCitizen() : void
Citizen
- name : string
- age : int
- idNum : int
- location : string
- personType : int
- infection_rate : int
- survival_rate : int
+ Citizen()
+ Citizen (string, int, int)
+ getName() : string
+ setName(string) : void
+ getLocation() : string
+ setLocation(string) : void
+ getAge() : int
+ getID() : int
+ getPersonType() : int
+ move(): bool
+ action(Citizen*&) : virtual void
+ getInfectionRate() : int
+ getSurvivalRate() : int
+ setPersonType(int) : int
Ignorant
- string name;
- int age;
- int idNum;
- string location;
- int infection_rate;
- int survival_rate;
+ Ignorant(string, int, int)
+ action(Citizen*&) :
Zombie
- string name;
- int age;
- int idNum;
- string location;
- int infection_rate;
+ Zombie(string, int, int)
+ action(Citizen*&) :
Doctor
- string name;
- int age;
- int idNum;
- string location;
- int infection_rate;
- int survival_rate;
+ Doctor(string, int, int)
+ action(Citizen*&) :
Alarmed
- string name;
- int age;
- int idNum;
- string location;
- int infection_rate;
- int survival_rate;
+ Alarmed(string, int, int)
+ action(Citizen*&) :

Challenges

- Keeping track of Pointers
- Using inheritance and polymorphism correctly throughout
- Testing as we go