Qifan Luo(1010945), Xinyi Chen(1582439)

Program Structures & Algorithms

Spring 2021

Final Assignment

- **Task**

Your task is to simulate the spread of a virus such as SARS-CoV-2, the pathogen behind COVID-19.
• Your simulations will take into account:
• The R factor of the disease (bonus points for including the k factor);
• The usage and effectiveness of masks;
• The prevalence of testing and contact tracing;
• The availability and efficacy of the vaccine;
• Any barriers to entry (including quarantining) into the subject area.

- **Aim of the project**

We hope to be able to simulate the movement of people and the spread of viruses through algorithms.

From the simulation, we will compare the difference in the spread of the virus between people without any protection and when they have protection.

We hope to find the best way to solve the epidemic in these different situations.

Qifan is to do simulation and algorithms while Xinxi is to do GUI to visualize the spread of the virus.

- **Complete project details**

The first step is to simulate the action of people. We will first create a city and the people living in it(PersonPool). When we think about how people in a crowd should act, we will naturally find that our daily life always revolves around three places: home, work place, and restaurant.

Therefore, we have determined the rules of people's actions according to this rule: continuous circulation among the three points, and can't leave the city.

The next step is to simulate the spread of virus. We divide people into five states in total: normal, infected but latent, confirmed and ill, quarantined, and dead. We will initialize some infected people in the crowd. Next, we will analyze the state of each person.

The first type is normal people. For normal people, they can move freely in the city. But they face the risk of infection when they move. If the distance from the infected person is less than the safe distance, then they will have a chance of being infected.

For people in the shadow period, the time of confirmed is random. People in the shadow period will not be detected by the hospital and quarantined, but if the tracking program is turned on, the situation will be different.

If the tracking procedure is activated, the tracking will begin when the confirmed person is admitted to the hospital. People who are diagnosed will be recorded when they were diagnosed and the place where they were diagnosed. After that, we will return to that place, find all people within a certain range to determine whether they are infected/in the incubation period, and then be sent to the hospital for isolation. This approach does not actually find out who infected him, but in real life we can also only find possible potential infections by tracking the whereabouts of the patient.

For those who have been diagnosed, we will first determine whether there is a location in the hospital/quarantine area. If there is a location and the hospital is given enough time to treat the patient, then the confirmed person will be sent to the hospital. There is also a probability of death in the hospital, but the probability of death will decrease. People who have not been admitted to the hospital have a greater chance of dying than those who are treated in the hospital.

Next are people in quarantine/hospital. People here will not move and have a higher probability of being treated. Here we will mention the details about

masks and vaccines. People wearing masks may still be infected, but the probability of infection will decrease. People who have been vaccinated may also be infected, but the probability of infection will be greatly reduced, and people who have been vaccinated will not die from the virus.

Finally, there are people who have died. They will no longer take any action.

- **Constant Explain**

R factor: We model the R factor as the probability of one person infecting the next. Wearing a mask and getting a vaccine will affect the probability of a patient infecting others. Different virus has different probability of infecting.

Incubation period: According to recent news reports, we set the incubation period to 14 days.

Case fatality rate: According to the overall situation of the world, the case fatality rate is roughly 0.02. But for small cities or where the infection is so dense that there is no hospital for treatment, the case fatality rate will increase significantly. In our simulation, in order to make the situation very obvious and easy to observe, we increased the fatality rate.

- **Implementation**

In this part, I will show the code related to the movement trajectory and virus spread. The display of the GUI will be shown in the conclusion evidence

and video.

```
this.city = city;
//没必要进行正态分布，随机分布即可
home[0] = x;
home[0] = y;

MoveTarget home = new MoveTarget(x,y);
moveTargets.add(home);
//代表着person只能在三个点中进行运动，已经确定好了
//第一次是单位的地址，第二次是食堂的地址；
for(int i = 0;i<2;i++){
    int x_point = random.nextInt(Constants.CITY_WIDTH);
    int y_point = random.nextInt(Constants.CITY_HEIGHT);
    MoveTarget t = new MoveTarget(x_point,y_point);
    moveTargets.add(t);
}
```

This is used to initialize the 3 point of the people should go.

```
double length = Math.sqrt(Math.pow(dX, 2) + Math.pow(dY, 2));//与目标点的距离

if (length < 1) {
    //判断是否到达目标点
    moveTarget.setArrived(true);

    //判断这个人位置的状态 以便于移动到下一个状态
    if(location == 0){
        location++;
        isBack = false;
    }else if(location == 1){
        if(isBack){
            location--;
        }else{
            location++;
        }
    }else if(location == 2){
        location--;
        isBack = true;
    }

    return;
}
```

This part is used to judge where a person came and where to go. The process is home->work place-> restaurant -> work place -> home.

```java
//这个函数用来判断健康人士是否会感染
public void normalAction(){
    List<Person> people = personPool.getPersonList();

    //如果小于安全距离且这个人感染了，那么有一定概率将会被感染
    for (Person person : people) {
        //如果这个人是正常人或者已经死去，那么将不会被感染
        if (person.getState() == State.NORMAL || person.getState() == State.DEATH) {
            continue;
        }
        //戴口罩和打疫苗都会降低感染的概率

        float broad_rate = Constants.broadRate;

        if(isMask){
            broad_rate = broad_rate*0.8f;
        }

        if(isVaccine){
            broad_rate = broad_rate*0.2f;
        }

        float random = new Random().nextFloat();
        if (random < broad_rate && distance(person) < SAFE_DIST) {
            //记录下被感染的时间和地点
            this.beInfected();
            break;
        }
    }

}
```

This part is used to determine whether a healthy person will be infected.

```java
//这个函数用来判断潜伏人士
public void shadowAction(){

    //增加一个正态分布用于潜伏期内随机发病时间
    double stdRnShadowtime = MathUtil.stdGaussian( sigma: 25,  u: constants.getShadowTime() / 2);
    //处理发病的潜伏期感染者
    if (myPanel.getWorldTime() - infectedTime > stdRnShadowtime) {
        state = State.CONFIRMED;//潜伏者发病
        confirmedTime = myPanel.getWorldTime();//刷新时间
    }

}
```

This part is used to determine when a person in the incubation period becomes ill.

```
if ( dieMoment == 0) {

    int destiny = new Random().nextInt( bound: 10000) + 1;//[1,10000]随机数
    //如果落在死亡区间，那么确定一个死亡时刻
    if (1 <= destiny && destiny <= (int) (constants.getFatalityRate() * 10000)) {
        int dieTime = (int) MathUtil.stdGaussian(constants.getDieVariance(), constants.getDieTime());
        dieMoment = confirmedTime + dieTime;//发病后确定死亡时刻(确诊时间+还能活几天)
    } else {
        dieMoment = -1;//逃过了死神的魔爪

    }
}
```

This is used to judge whether the person who has been diagnosed will die and the time of death.

```
//医院需要一定的时间来收治病人
if (myPanel.getWorldTime() - confirmedTime >= constants.getHospitalReceiveTime()) {
    //如果患者已经确诊，且（世界时刻-确诊时刻）大于医院响应时间，即医院准备好病床了，可以抬走了
    Bed bed = hospital.pickBed();//查找空床位
    if (bed == null) {

        //没有床位了，报告需求床位数

    } else {
        //安置病人
        useBed = bed;
        state = State.FREEZE;
        setX(bed.getX());
        setY(bed.getY());
        bed.setEmpty(false);
    }
}
```

This part is to determine whether the patient can be admitted to the hospital and isolated.

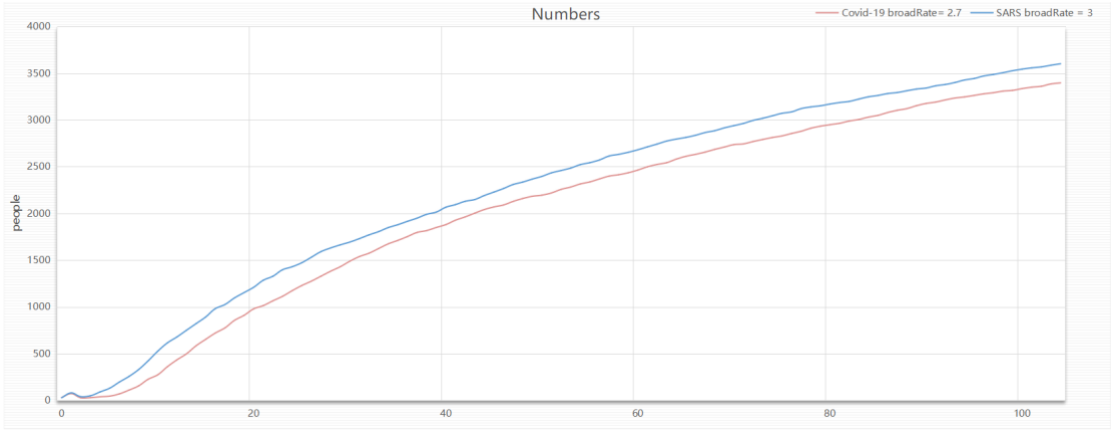
```java
public void trace(){
    //追踪病人的行动轨迹，但是只追踪一次
    if(isTrack){
        List<Person> people = personPool.getPersonList();
        for(Person person :people){
            int dX = person.getX() - confirmLocation[0];
            int dY = person.getY() - confirmLocation[1];

            double length = Math.sqrt(Math.pow(dX, 2) + Math.pow(dY, 2));//与目标点的距离
            //如果距离小于一定距离（称之为能追踪的最大范围且已经感染了或者在潜伏中就进入隔离程序）
            if(length<5 && (person.state == State.CONFIRMED || person.state == State.SHADOW)){
                Bed bed = hospital.pickBed();//查找空床位
                if (bed == null) {

                    //没有床位了，报告需求床位数

                } else {
                    //安置病人
                    person.useBed = bed;
                    person.state = State.FREEZE;
                    setX(bed.getX());
                    setY(bed.getY());
                    bed.setEmpty(false);
                }
            }
        }
```

This part is used to trace whether there are other patients on the trajectory of a certain infected patient.

```java
    //打了疫苗的人不会死亡
    if(isVaccine){
        dieMoment = -1;
    }

    //如果这个人既未能被治愈也没确定死亡
    if ( dieMoment == 0 && cureMoment == 0) {

        int destiny = new Random().nextInt( bound: 10000) + 1;//[1,10000]随机数
        //如果落在死亡区间，那么确定一个死亡时刻
        //在医院中的死亡概率要比在外面的要小
        if (1 <= destiny && destiny <= (int) (constants.getFatalityRateHospital() * 10000)) {
            int dieTime = (int) MathUtil.stdGaussian(constants.getDieVariance(), constants.getDieTime());
            dieMoment = confirmedTime + dieTime;//发病后确定死亡时刻(确诊时间+还能活几天)
        } else {
            dieMoment = -1;//逃过了死神的魔爪
            int cureTime = (int) MathUtil.stdGaussian(constants.getCureVariance(), constants.getCureTime());
            cureMoment = confirmedTime + cureTime;

        }
    }
```

This part is used to judge whether the person undergoing treatment in the hospital will die.

- **Conclusion**

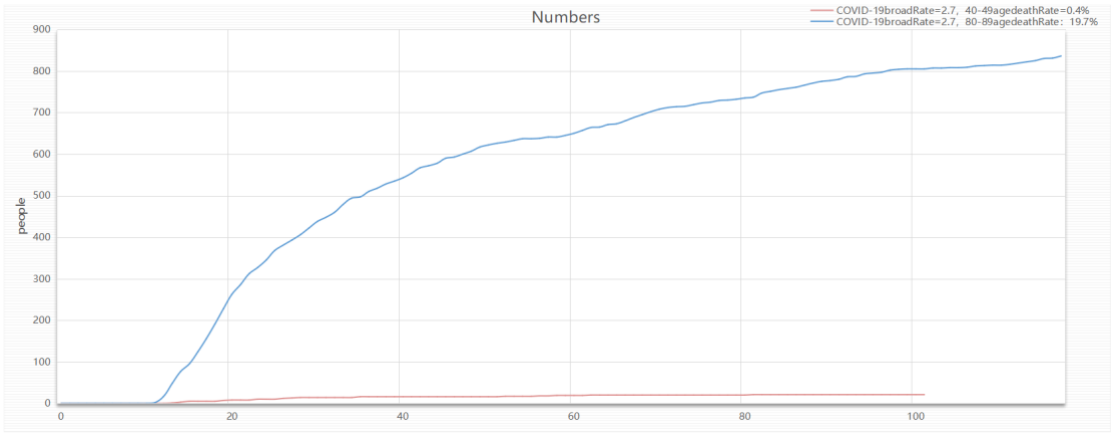


- graph 01

1.From the perspective of dissemination rate ： Covid-19 vs Sars

graph 01 conclusion：

The number of hospital beds is also controlled at 100, and without the mandatory use of masks and vaccination, the R0 value of SARS is 3 and the R0 value of COVID-19 is 2.7. The graph shows that the number of SARS infections is more than that of Covid-19 most of the time, and the increasing trend of the two lines is basically the same.

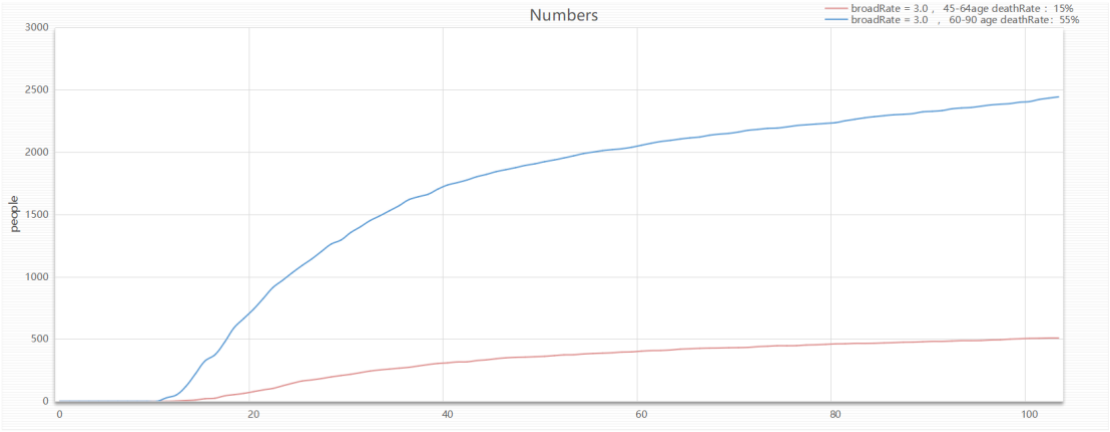


- graph 02

2.Comparing the number of deaths in different age groups

graph 02 conclusion:

The beds are all at 100, and the mortality rate varies by age without mandatory mass masking and vaccination. The mortality rate at age 40-49 is 0.4%, with a flat curve and no significant increase, but the mortality rate at age 80-89 is at 19.7, with a logarithmic function increase in the curve and a very large increase before 50.

- 

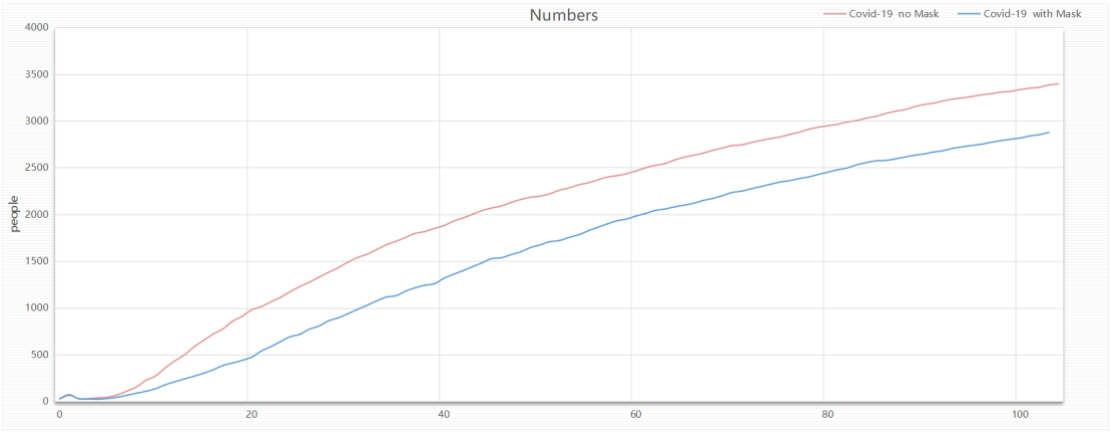
- graph 03

3.Comparing the number of deaths in different age groups

graph03 conclusion :

SARS, like COVID-19, has different age groups and different mortality rates, with a median death rate increasing logarithmically for all middle-aged and older adults over 70 years of age, but SARS has a higher mortality rate than COVID-19 at the same age.

-

- 

    - graph 04

4.Analysis of the effectiveness of masks

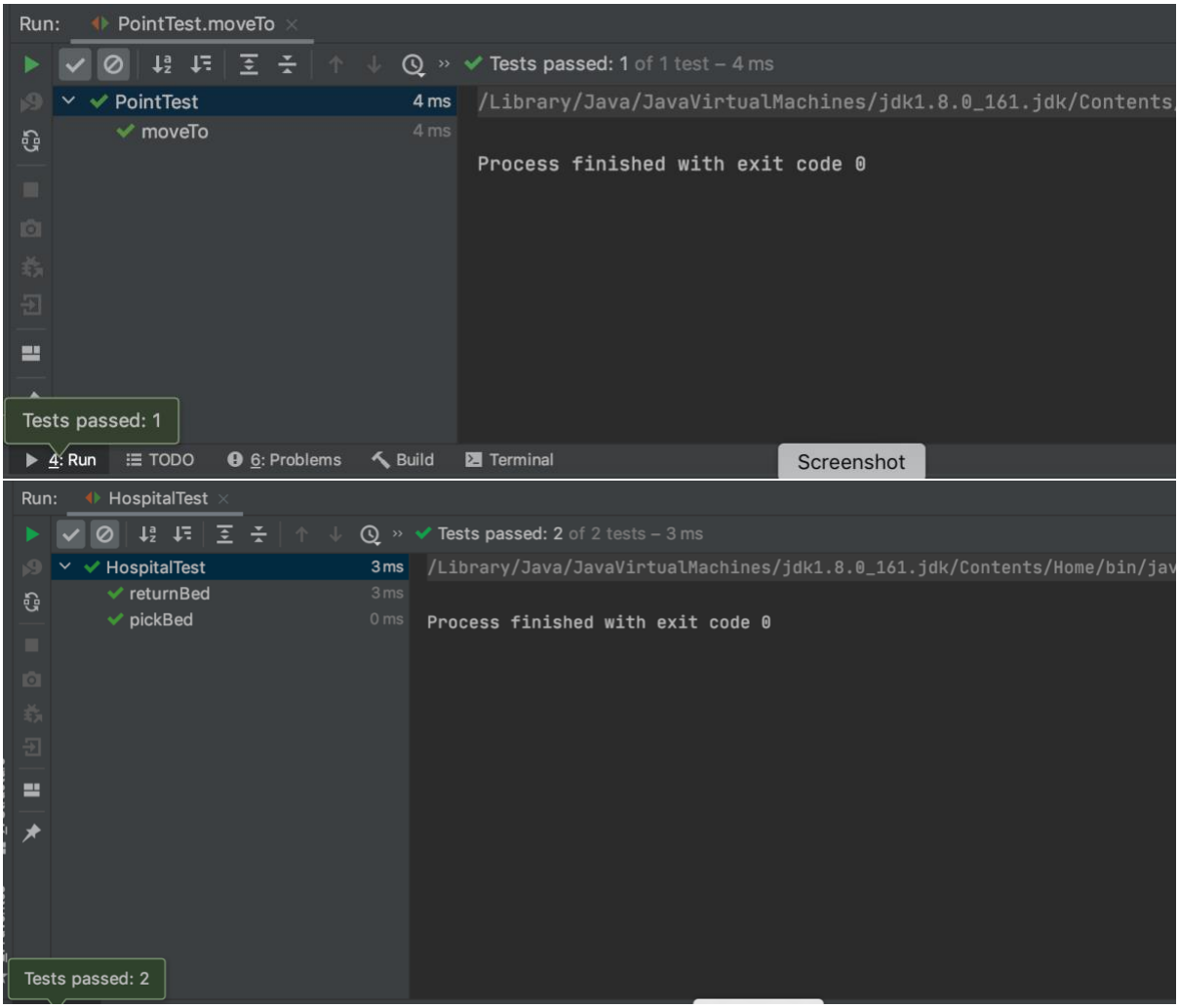
graph 04 conclusion:

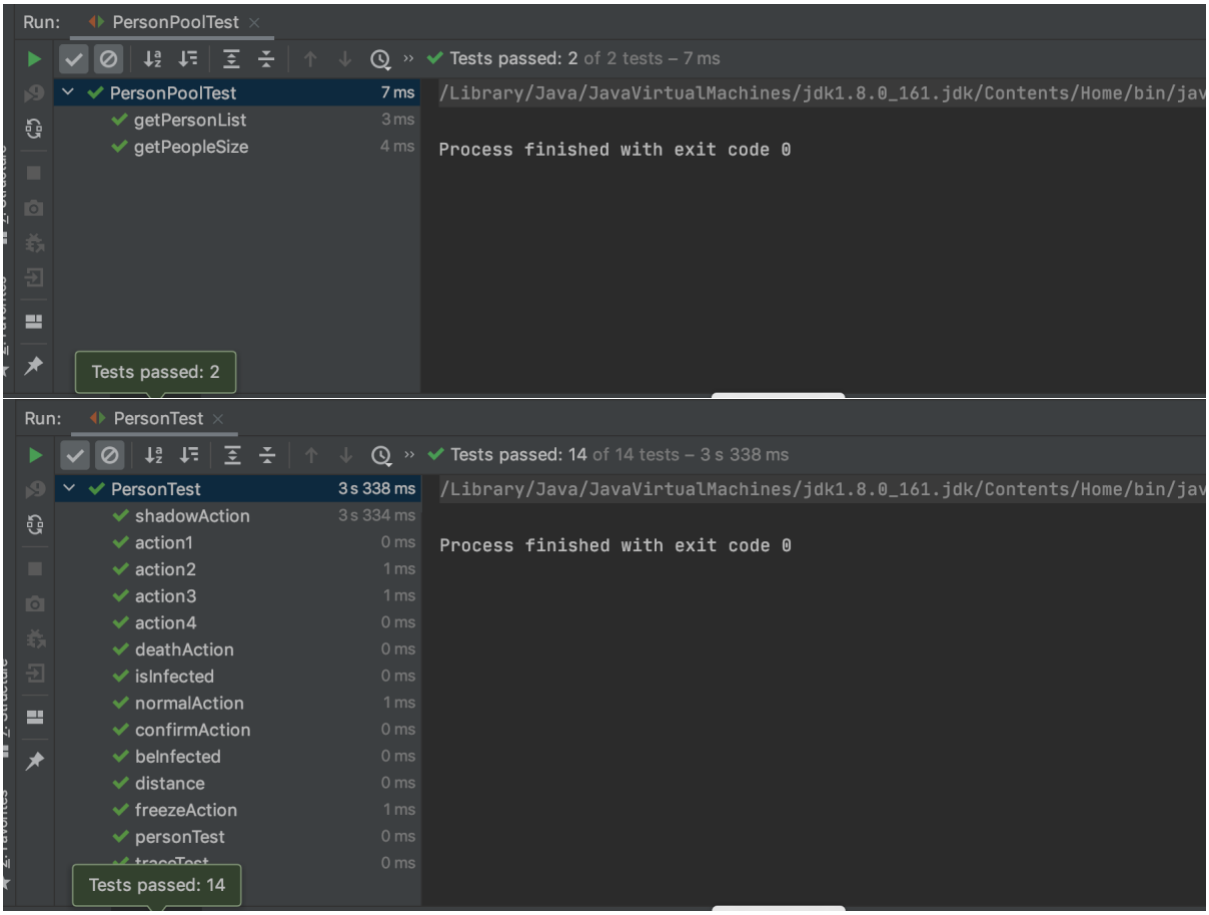
Wearing a mask does reduce the number of people infected, all things being equal: rate of transmission, bed space, availability of vaccination, and lethality of the virus.

- **Evidence to support the conclusion:**
- **Data representation:**

  Our data is in the 4Graph.xls.

- **Unit tests result:**


Run:
PointTest.moveTo
Tests passed: 1 of 1 test – 4 ms
PointTest 4 ms
moveTo 4 ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/
Process finished with exit code 0
Tests passed: 1
4: Run
TODO
6: Problems
Build
Terminal
Screenshot
Run:
HospitalTest
Tests passed: 2 of 2 tests – 3 ms
HospitalTest 3 ms
returnBed 3 ms
pickBed 0 ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/jav
Process finished with exit code 0
Tests passed: 2


Run:
PersonPoolTest
Tests passed: 2 of 2 tests – 7 ms
PersonPoolTest 7 ms
getPersonList 3 ms
getPeopleSize 4 ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/jav
Process finished with exit code 0
Tests passed: 2

Run:
PersonTest
Tests passed: 14 of 14 tests – 3 s 338 ms
PersonTest 3 s 338 ms
shadowAction 3 s 334 ms
action1 0 ms
action2 1 ms
action3 1 ms
action4 0 ms
deathAction 0 ms
isInfected 0 ms
normalAction 1 ms
confirmAction 0 ms
beInfected 0 ms
distance 0 ms
freezeAction 1 ms
personTest 0 ms
0 ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/jav
Process finished with exit code 0
Tests passed: 14