姓名：刘睿哲

学号：202220008

开发目标

我写这个游戏是为了尝试实现课上学到的东西，但因为时间匆忙加自己拖延，所以就没有精进扩展，只为了完成基本内容和练习课上基本知识点。

设计理念

关于玩法

游戏思路：本次游戏设计主要是在给的框架和ui下进行修改扩展。我在creature里设计了boss类型的怪物，有更高的血量攻击，并且它的ai也和普通不同，会主动找玩家打架。游戏流程是清理完四个小怪后boss会出现和玩家决斗，从游戏开始到杀死boss通关所用的时间为玩家得分，用时越短越好。

玩家具备的特点：

- 可以自由移动以及攻击
- 可以查看已经猎杀的普通怪物数量

普通monster应当具备以下特点

- 可以攻击玩家，但会主动逃离玩家
- 血少攻低，基本碰了就死，所以他们一直在逃跑
- 杀死四个普通monster会召唤黄色boss

boss应当具备以下特点

- 可以攻击玩家
- 会主动接近玩家去猎杀
- 杀死boss后通关游戏

关于UI

UI我选用了jw04的默认UI，只是在怪物选择上面进行了颜色修改和图形选择

关于服务器

多人联机为了简便操作，设计成玩家自行单机游玩，在通关后的win界面可以和服务器联机，发送自己的分数，可以查看自己分数是否在服务器排名前10。写了一个服务器Server需要在打开游戏时候额外窗口打开，接受玩家得分数据并且写在txt里进行记录。

关于多线程

除了主线程，我还创建了以下线程：

- 用于刷新屏幕的线程，其每一秒刷新十次画面（经过试验，不影响输出效果）
- 玩家线程，其用于接收输入（移动和攻击），并更新玩家信息（hp和得分）
- 各个bot线程，每个线程对应一个bot，设计在了怪物的AI里

关于怪物AI

我在怪物的AI里设计了线程

在creatureAI里定义了run

```
@Override
   public void run() {
   }
```

然后在普通怪物和boss的ai里继承了creatureai，然后进行override

普通怪物的AI会进行随机走位

```
@Override

    public void run(){
        while(this.creature.hp()>0){
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            Random rand = new Random();
            switch(rand.nextInt(4)){
                case 0:
                    this.creature.moveBy(1,0);
                    break;
                case 1:
                    this.creature.moveBy(-1,0);
                    break;
                case 2:
                    this.creature.moveBy(0,1);
                    break;
                case 3:
                    this.creature.moveBy(0,-1);
                    break;
            }
        }
    }
```

boss的ai会主动找寻玩家

```
@Override

    public void run(){
        while(this.creature.hp()>0){
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            int dx, dy;
            dx = getdirection(this.creature.x(), player.x());
            dy = getdirection(this.creature.y(), player.y());
            if (dx == 0 || dy == 0) {
                this.creature.moveBy(dx, dy);
            }
            else {
                Random rand = new Random();
                switch (rand.nextInt(2)) {
                case 0:
                    this.creature.moveBy(dx, 0);
                    break;
                case 1:
                    this.creature.moveBy(0, dy);
                    break;
                }
            }
        }
    }
```

技术问题

关于线程和屏幕刷新

在主函数部分进行刷屏

```
 public void keyTyped(KeyEvent e) {
    }
    public static void main(String[] args) {
        ApplicationMain app = new ApplicationMain();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        app.setVisible(true);

        new Thread(
                ()->{
                        while (true){
                            try {
                                Thread.sleep(50);
                            } catch (InterruptedException e) {
                                e.printStackTrace();
                            }
                            app.repaint();
                        }
                }
        ).start();

    }
```

根据jw04的模板，主函数调用一个screen类，一开始是startscreen，然后根据操作返回playscreen。在playscreen中，我调用所有怪物的AI进行更新。

```
private  Creature[] creaturelist;
    private void createCreatures(CreatureFactory creatureFactory) {
        this.player = creatureFactory.newPlayer(this.messages);

        creaturelist = new Creature[4];
        for (int i = 0; i < 4; i++) {
            creaturelist[i] = creatureFactory.newMonster();
            Thread t = new Thread(creaturelist[i].getAI());
            t.start();
        }


    }
```

关于通信部分

在最后通关后进入winscreen，在里面我调用一个客户端client和服务器server进行沟通。

```
private void getresult() {
        try {
            Client client = new Client(score);
            this.result = client.getResult();
        } catch (IOException e) {
            e.printStackTrace();
```

```
            return;
        }
    }
```

client发送现在得分，之后接受server的返回，会返回自己是否是top10中的一个。

```
public Client(long score) throws IOException {
        String s = String.valueOf(score);
        byte[] data = s.getBytes();
        InetAddress add = InetAddress.getByName("localhost");
        int port = 8800;
        DatagramPacket packet = new DatagramPacket(data, data.length, add, port);
        DatagramSocket socket = new DatagramSocket();
        socket.send(packet);
        byte[] data2 = new byte[100];
        DatagramPacket packet2 = new DatagramPacket(data2, data2.length);
        System.out.println("Waiting");
        socket.receive(packet2);
        result = new String(data2, 0, packet2.getLength());
        System.out.println("Receive");
        socket.close();
    }
```

服务器部分开始时候会先input一下本地的数据，在收到客户端信息后进行比较然后返回排名信息

```
public static void main(String[] args) throws IOException {

        DatagramSocket socket = new DatagramSocket(Port);
        byte[] newscore = new byte[100];
        DatagramPacket packet = new DatagramPacket(newscore, newscore.length);
        System.out.println("Waiting");
        socket.receive(packet);
        String info = new String(newscore, 0, packet.getLength());
        long score = Long.parseLong(info);
        System.out.println(score);
        byte[] data;
        if (score < a[0]) {
            data = "You are NO.1".getBytes();
        } else {
            data = "You are not NO.1".getBytes();
        }
        InetAddress add = packet.getAddress();
        int cport = packet.getPort();

        DatagramPacket packet2 = new DatagramPacket(data, data.length, add, cport);
        socket.send(packet2);
        System.out.println("Close");
        socket.close();
    }
```

关于保存和加载

保存是在游戏中按下enter即可保存，会在本地写入txt，里面记载了现在的位置信息等。

```
 private void Savegame() throws IOException {
        File file = new File("save.txt");
        BufferedWriter fileout = new BufferedWriter(new FileWriter(file));
```

```
        int[] a = new int[3];
        a[0] = player.x();
        a[1] = player.y();
        a[2] = (int) this.startTime;
        String line = "";
        for (int i = 0; i < 3; i++) {
            line = line + String.valueOf(a[i]) + "\r\n";
        }
        fileout.write(line);
        fileout.close();
    }
```

加载是在playscreen加入前判断是否有信息要加载。先修改startscreen里面，给playerscreen带参数，要加载了flag设置为0，否则1

playscreen部分

```
public PlayScreen(int flag) throws IOException {
        this.screenWidth = 40;
        this.screenHeight = 21;
        this.startTime = System.currentTimeMillis();
        this.score = 0;
        createWorld();
        this.messages = new ArrayList<String>();
        this.oldMessages = new ArrayList<String>();
        this.isboss = false;

        CreatureFactory creatureFactory = new CreatureFactory(this.world);
        createCreatures(creatureFactory);
        if (flag == 0) {
            Loadgame();
        }
    }

    private void Loadgame() throws IOException {
        File file = new File("save.txt");
        FileInputStream fileinput = new FileInputStream(file);
        InputStreamReader reader = new InputStreamReader(fileinput);
        BufferedReader br = new BufferedReader(reader);
        int[] a = new int[3];
        String line = "";
        for (int i = 0; i < 3; i++) {
            line = br.readLine();
            a[i] = Integer.valueOf(line).intValue();
        }
        this.startTime = a[2];
        player.setX(a[0]);
        player.setY(a[1]);
        br.close();
    }
```

restartscreen部分

```
@Override
    public Screen respondToUserInput(KeyEvent key) {
        switch (key.getKeyCode()) {
            case KeyEvent.VK_ENTER:
                try {
                    return new PlayScreen(1);
                } catch (IOException e) {
```

```
                    e.printStackTrace();
                }
            case KeyEvent.VK_UP:
                try {
                    return new PlayScreen(0);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            default:
                return this;
        }
    }
```

工程问题

我没学过设计模式啥的，不过按照老师上课讲的java面向对象和范式设计，我尽量让设计的结构有面向对象性质。

一点感受

上述的过程看起来并不是十分复杂，但是我却花费了大量的时间来实现。感觉自己已经尽可能的简化要求，只想搞个简单的玩意交差，但发现即使再简单也有很多小坑等着我。最后debug了好久，也对课堂知识印象更加深刻。也不说以后是否有空会完善吧，我想应该是没有了，不过我以后一定会经常想起java作业这个，搞了好久才搞通的毫无难度但又很繁琐的小玩意，会让我更努力的去写任何程序，即使第一眼看起来很简答。