I. Purpose of the experiment

As developers, we look to a variety of approaches to help us understand and analyze problems in order to choose the best methodology for programming the solution. Object-oriented analysis and design is a software development practice which aligns with characteristics found in object oriented programming. Let's learn about this.

II. Experimental content

For this assignment, you will use a simple methodology for OOAD analysis and design to code a simple version of the snake game employing basic object oriented programming characteristics:

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

The version of Snake for this assignment is going to be a simple program.

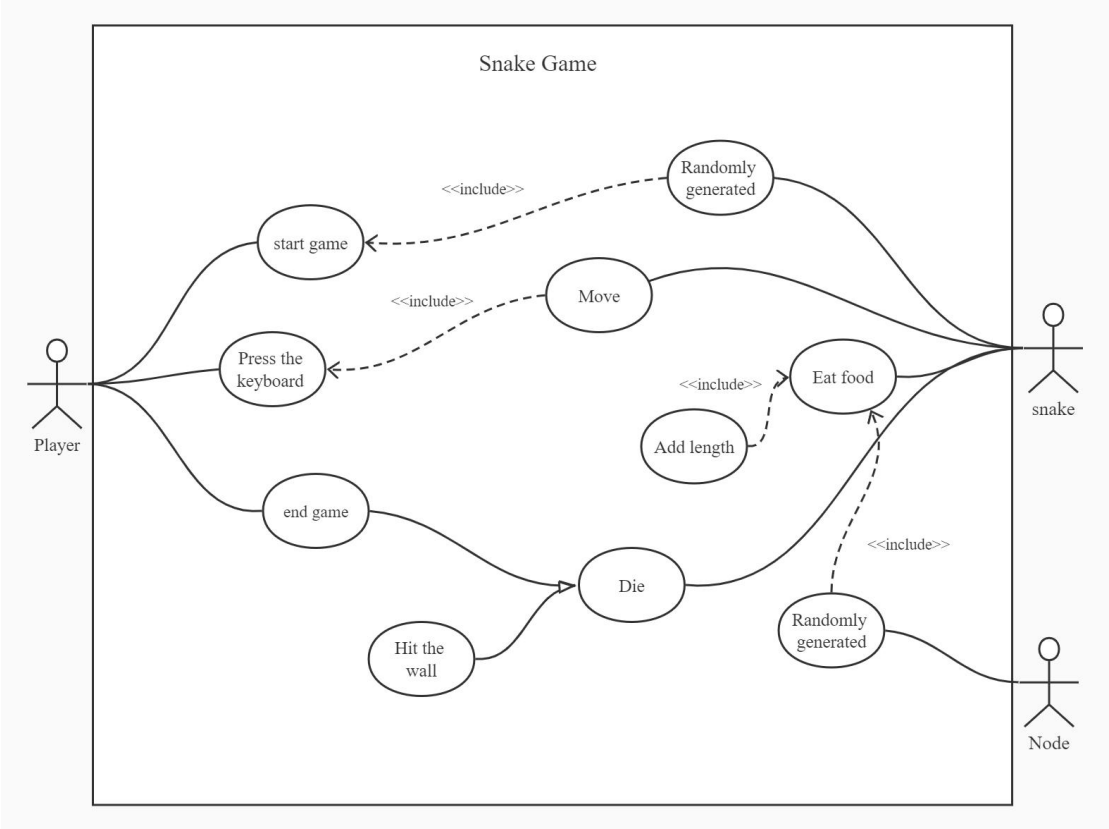
1. The program will run in the terminal
2. Single player
3. The player has to press "enter" every time they want to make a move
4. There is a randomly generated (starting) dot
5. The snake can move up, down, left, or right
6. If the snake eats the dot, it grows and the score goes up by one
7. If the snake runs into the wall or itself, the game resets

III. Experimental steps

Step1: Define Use Cases for snake game.
Step2: Define Domain Model for snake game.
S**tep3:** Assign Object Responsibilities and Draw Interaction Diagrams for snake game.
Step4: Choose a version about smalltalk.
Step5: Use object-oriented thinking to create snake game in smalltalk. Create class for node and sneak which is consist of nodes.

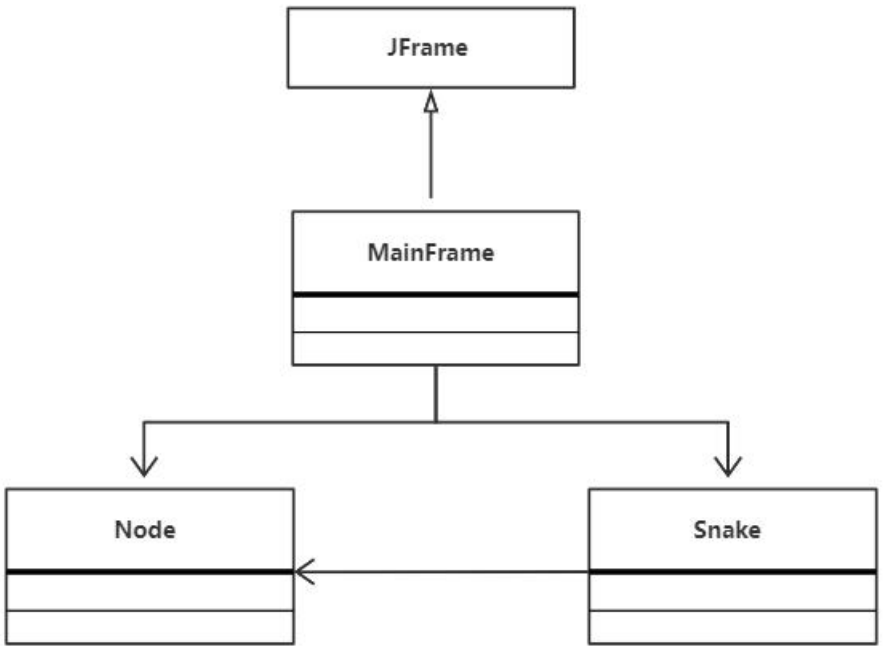
IV. Experimental results and analysis(Discussion)

1. Use Cases



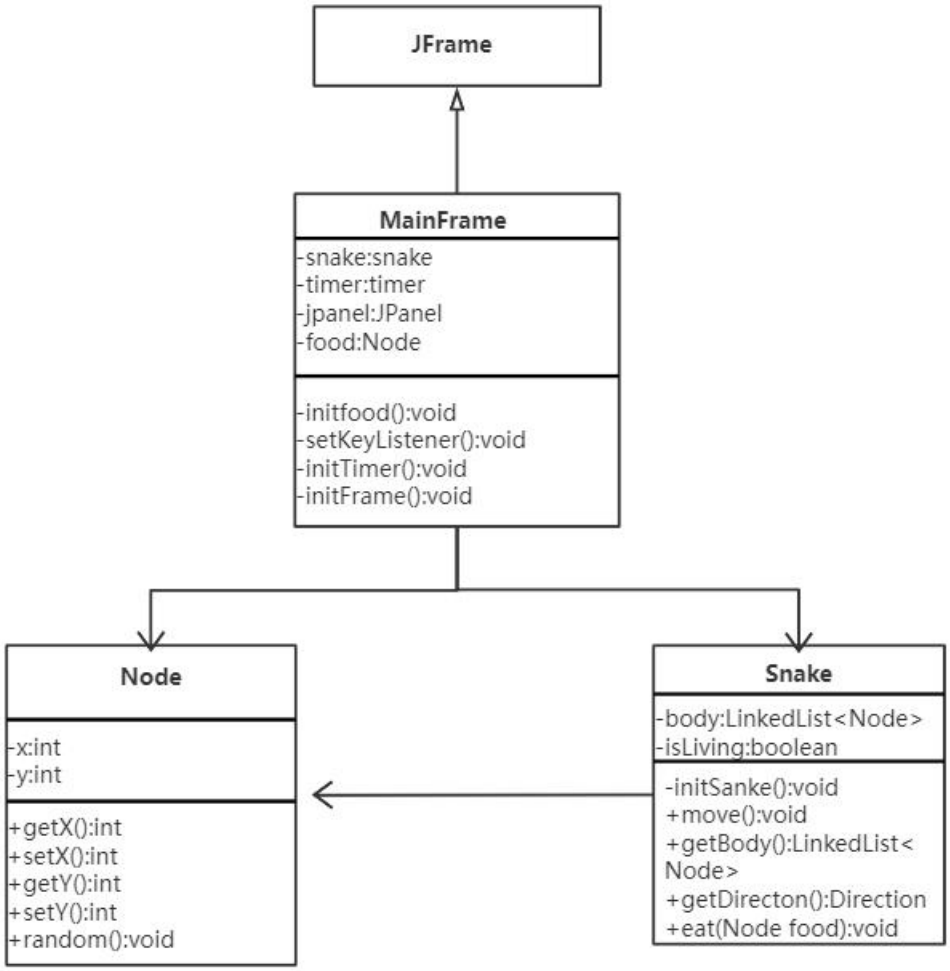
Graph 1 Use case

2. Domain model



Graph 2 Domain model

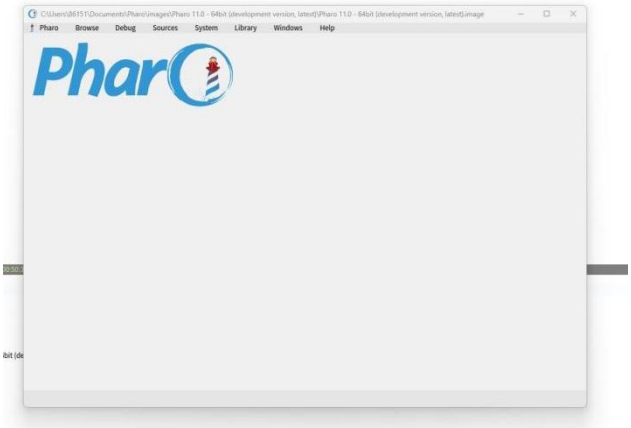
3. Interaction diagram



Graph 3 Interaction diagram(in java)

4. Get smalltalk

Choose pharo as IDE.



Graph 4 IDE

But I have encountered some difficulties in learning this language. Because there are fewer tutorials about this language and fewer cases. Therefore, it was difficult for me to finish this assignment within the time limit and I am sorry for that. In order to maximize my learning of OOD, I chose to complete the next task through a snake

game written in java which I had been exposed to

5. About the role of analysis and design(in java)

I think use case is a great help in helping to build the whole project. By using case, I can clearly see the roles in the project and the interaction between roles and use cases. And, I can use it to sort out the logical relationship between the functions.

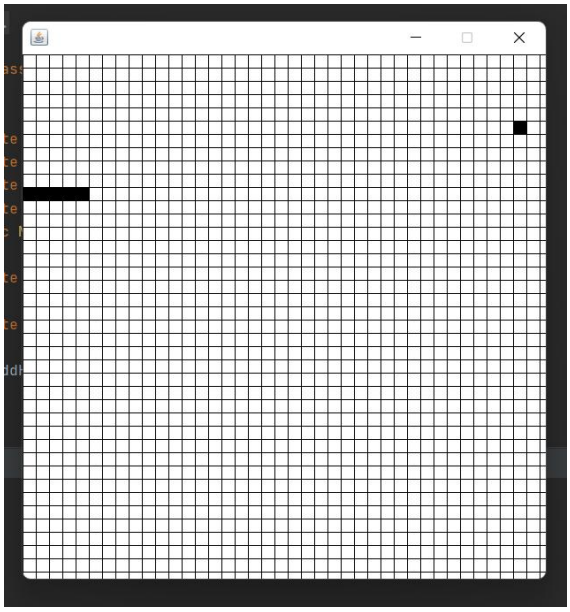
Domain model and Interaction diagram can help me to understand and integrate into the object-oriented thinking very well. With the help of them, I can make my task clear in the process of programming.

6. OOP characteristics in the coding(in java)

Inheritance: In the code, I creat a MainFrame class who is the subclass of JFram

Encapsulation: I creat a class for snake.For the outside world, all properties and methods related to snake are encapsulated in the class.

Abstraction: I abstracted Node as a separate class because we have to use Node in both snake and food, i.e. snake is composed of node and food is a node.



Graph 5 Game running

V. Experimental experience and gains

1. Learned object-oriented programming and object-oriented thinking.
2. Learned and mastered object-oriented analysis and modeling.
3. Learned about the smalltalk.

VI. Core Code

MainFrame:

```
package snakegame;

import java.awt.Graphics;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.util.LinkedList;
```

```
import java.util.Timer;
import java.util.TimerTask;

import javax.swing.JFrame;
import javax.swing.JPanel;

public class MainFrame extends JFrame {

    //蛇
    private sanke sanke;
    private Timer timer;//定时器，在规定时间内调用蛇；
    private JPanel jPanel;//游戏棋盘；
    private Node food;//食物；
    public MainFrame(){
        //初始化窗体的参数

        initFrame();

        //初始化游戏棋盘；

        inintGamePanel();

        //初始化蛇；

        initsanke();

        //初始化食物；

        initfood();

        //初始化定时器

        initTimer();

        //设置键盘监听；

        setKeyListener();

    }

    private void initfood() {

        food = new Node();
        food.random();
```

```
}

private void setKeyListener() {

    addKeyListener(new KeyAdapter(){
        //当键盘按下时自动调用此方法
        @Override
        public void keyPressed(KeyEvent e){
            //键盘中每一个见都有一个编号

            switch(e.getKeyCode()){

                case KeyEvent.VK_UP://上
                    //改变蛇的运动方向；
                    if(sanke.getDirecton()!=Directon.DOWN){
                        sanke.setDirecton(Directon.UP);
                    }
                    break;
                case KeyEvent.VK_DOWN://XIA
                    if(sanke.getDirecton()!=Directon.UP){
                        sanke.setDirecton(Directon.DOWN);
                    }
                    break;
                case KeyEvent.VK_LEFT://左
                    if(sanke.getDirecton()!=Directon.RIGHT){
                        sanke.setDirecton(Directon.LEFT);
                    }
                    break;
                case KeyEvent.VK_RIGHT://右
                    if(sanke.getDirecton()!=Directon.LEFT){
                        sanke.setDirecton(Directon.RIGHT);
                    }
                    break;
            }

        }
    });


}

//初始化定时器
private void initTimer() {
```

```
        //创建定时器对象；
        timer = new Timer();

        //初始化定时任务；

        TimerTask timertask = new TimerTask() {

            @Override
            public void run() {

                sanke.move();
                //判断是否迟到食物
                Node node = sanke.getBody().getFirst();
                if(node.getX()==food.getX()&&node.getY()==food.getY()){
                    sanke.eat(food);
                    food.random();

                }
                //重新绘制棋盘；

                jPanel.repaint();
            }
        };
        //100 毫秒执行一次；
        timer.scheduleAtFixedRate(timertask,0,100);
    }
    private void initsanke() {

        sanke = new sanke();

    }
    //初始化游戏棋盘；
    private void inintGamePanel() {

        jPanel =    new JPanel(){
            //绘制游戏棋盘中的内容
            @Override
            public void paint(Graphics g) {

                //清空棋盘；

                g.clearRect(0, 0, 600, 600);

                //Graphics g 看作是一个画笔提供了很多方法可以绘制一些基本的图形
```

```
                //绘制四十条横线；

                for(int i=0; i<=40; i++){
                    g.drawLine(0, i*15, 600, i*15);
                }
                //绘制四十条竖线；
                for(int i=0; i<=40; i++){
                    g.drawLine(i*15, 0,i*15, 600);
                }
                //绘制蛇；

                LinkedList<Node> body = sanke.getBody();

                for(Node node : body){

                    g.fillRect(node.getX()*15, node.getY()*15, 15, 15);
                }
                //绘制食物；

                g.fillRect(food.getX()*15, food.getY()*15, 15, 15);

            }

        };

        //棋盘添加到窗体中；
        add(jPanel);

    }

    private void initFrame() {
        //创建窗体宽和高

        setSize(610,640);

        //设置窗体的位置；

        setLocation(700,200);

        //设置关闭按钮的作用；

        setDefaultCloseOperation(EXIT_ON_CLOSE);
```

```
        //固定窗体大小

        setResizable(false);
    }


    public static void main(String[] args){
        //创建窗体对象并显示

        new MainFrame().setVisible(true);
    }

}
```

Node:

```
package snakegame;

import java.util.Random;

//节点类每一条蛇有由若干个节点组成的
public class Node {

    private int x;
    private int y;

    public Node(){

    }

    public Node(int x,int y){
        this.x = x;
        this.y = y;
    }

    public int getX(){
        return x;
    }

    public void setX(int x){
        this.x = x;
    }

    public int getY(){
        return y;
```

```
    }

    public void setY(int y){
        this.y = y;
    }

    //随机生成位置（食物）；
    public void random(){
        //创建 random 对象；

        Random r = new Random();

        //随机生成横坐标；

        this.x = r.nextInt(40);

        //随机生成纵坐标

        this.y = r.nextInt(40);
    }
}
```

Snake:

```
package snakegame;

import java.util.LinkedList;

public class sanke {

    //蛇的身体

    private LinkedList<Node> body;

    //蛇的运动方向；
    //默认向左
    private Directon directon = Directon.LEFT;
    //蛇是否活着；
    private boolean isLiving = true;

    //构造方法在 sanke 对象是时执行；

    public sanke(){
        //初始化射身体；
```

```
        initSanke();
    }

    private void initSanke() {

        //创建集合

        body = new LinkedList<>();

        //创建六个节点放到集合中

        body.add(new Node(16,10));
        body.add(new Node(17,10));
        body.add(new Node(18,10));
        body.add(new Node(19,10));
        body.add(new Node(20,10));
        body.add(new Node(21,10));
    }

    //蛇随着蛇头的方向移动
    //控制蛇移动
    public void move(){
        if(isLiving){
            //获取蛇头；

            Node head = body.getFirst();

            switch(directon){

                case UP:

                    //在蛇头上方添加一个节点；

                    body.addFirst(new Node(head.getX(),head.getY()-1));

                    break;
                case DOWN:

                    body.addFirst(new Node(head.getX(),head.getY()+1));

                    break;
                case LEFT:

                    body.addFirst(new Node(head.getX()-1,head.getY()));
```

```
                    break;
                case RIGHT:

                    body.addFirst(new Node(head.getX()+1,head.getY()));

                    break;
            }
            //删除最后的节点；

            body.removeLast();

            //判断蛇是否活着；
            head = body.getFirst();
            if(head.getX()<0||head.getY()<0||head.getX()>=40||head.getY()>=40){
                isLiving = false;
            }

            //判断蛇是否碰到自己；
            for(int i=0;i<body.size();i++){
                Node node = body.get(i);
            /*if(head.getX()==node.getX()&&head.getY()==node.getY()){
                isLiving = false;
            }*/
            }

            //判断是否迟到食物
        }
    }

    public LinkedList<Node> getBody(){
        return body;
    }

    public void setBody(LinkedList<Node> body){

    }

    public Directon getDirecton(){
        return directon;
    }

    public void setDirecton(Directon directon){
        this.directon = directon;
```

```
    }

    public void eat(Node food) {
        // 吃食物；
        Node head = body.getFirst();

        switch(directon){

            case UP:

                //在蛇头上方添加一个节点；

                body.addFirst(new Node(head.getX(),head.getY()-1));

                break;
            case DOWN:

                body.addFirst(new Node(head.getX(),head.getY()+1));

                break;
            case LEFT:

                body.addFirst(new Node(head.getX()-1,head.getY()));

                break;
            case RIGHT:

                body.addFirst(new Node(head.getX()+1,head.getY()));

                break;
        }


    }
}
```

Direction:

```
package snakegame;

//枚举类

public enum Directon {

    UP,DOWN,LEFT,RIGHT;
```

}

VII. Reference

1. https://blog.csdn.net/qq_44433261/article/details/107598788?spm=1001.2101.3001.6650.7&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-7-107598788-blog-116146124.pc_relevant_multi_platform_whitelistv3&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-7-107598788-blog-116146124.pc_relevant_multi_platform_whitelistv3&utm_relevant_index=8
2. https://gitcode.net/mirrors/wojiushimogui/Snake/-/tree/master/Snake2.3/src/model
3. https://github.com/Huldoser/console-snake
4. https://blog.csdn.net/Freedom0000000/article/details/118220808?spm=1001.2101.3001.6650.11&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-11-118220808-blog-116146124.pc_relevant_multi_platform_whitelistv3&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-11-118220808-blog-116146124.pc_relevant_multi_platform_whitelistv3&utm_relevant_index=12
5. https://blog.csdn.net/luoyaxing0812/article/details/112004318