2.人员信息

源码:

https://github.com/gyw666/javaDesign2

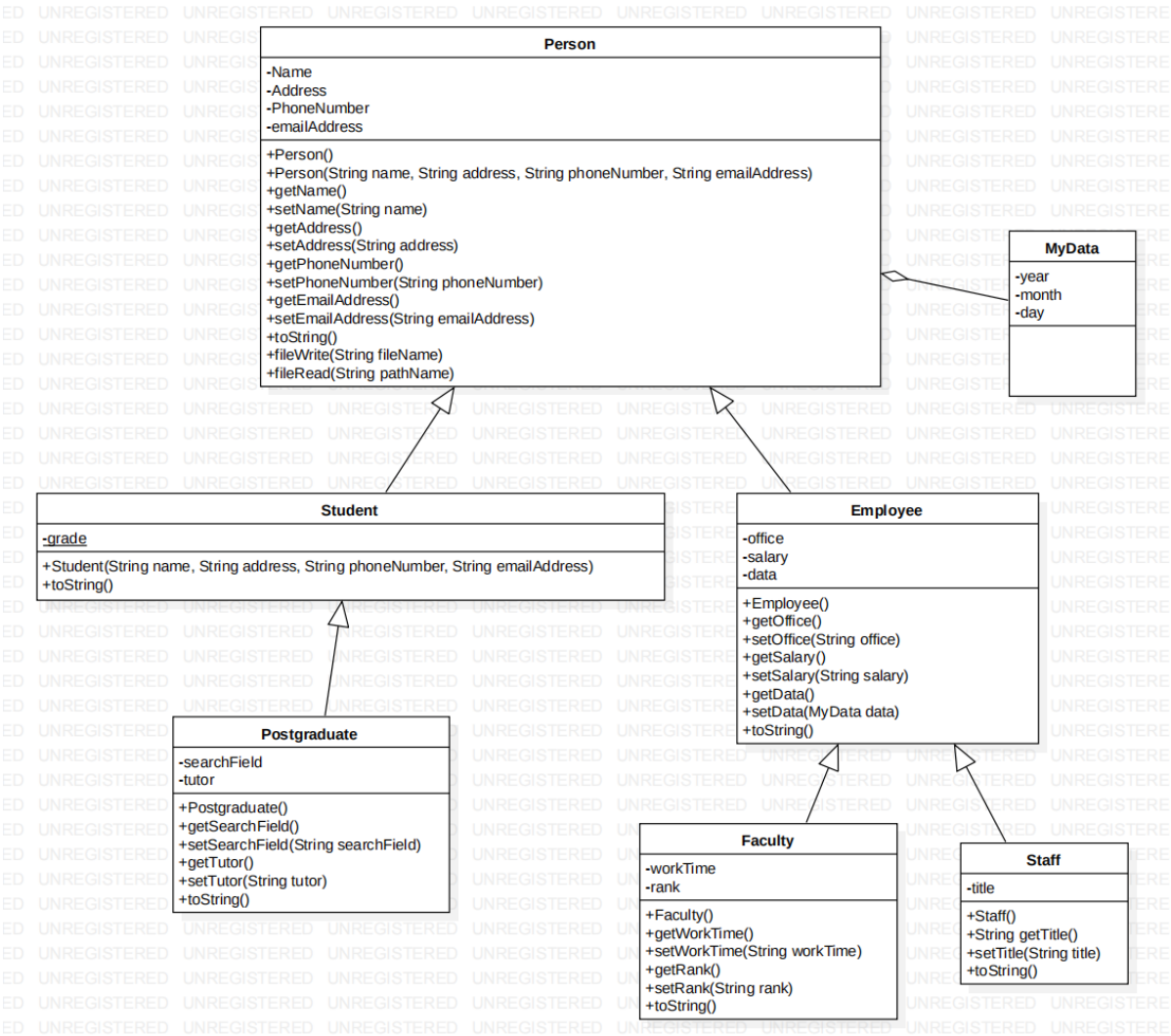
version 1

题目要求:

设计 Person 类，有姓名、地址、电话号码和电子邮件等属性。其子类为 Student 和 Employee，其中 Employee 类又有教员类 Faculty 和职员类 staff。学生类有班级状态（大一、大二、大三或大四），这些状态为常量。Employee 类有办公室、工资和受聘日期。定义一个名为 MyDate 的类，包含 year、month 和 day 数据域。Faculty 类有办公时间和级别。Staff 类有职务称号。重写每个类中的 toString 方法，显示相应的类名和人名。

编写测试类 Test1，要求随机产生不同的对象，并将这些对象存储到数组中，并用方法 printMessages(Person[] persons)将对象的信息输出。

思路:



代码

```
package personType;

import java.io.*;

public abstract class Person {
    private String name;
    private String address;
    private String phoneNumber;
    private String emailAddress;

    public Person() {
    }

    public Person(String name, String address, String phoneNumber, String
emailAddress) {
        this.name = name;
        this.address = address;
        this.phoneNumber = phoneNumber;
        this.emailAddress = emailAddress;
    }

    /**
     * 获取
     * @return name
     */
    public String getName() {
        return name;
    }

    /**
     * 设置
     * @param name
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * 获取
     * @return address
     */
    public String getAddress() {
        return address;
    }

    /**
     * 设置
     * @param address
     */
    public void setAddress(String address) {
        this.address = address;
    }
```

```
    /**
     * 获取
     * @return phoneNumber
     */
    public String getPhoneNumber() {
        return phoneNumber;
    }

    /**
     * 设置
     * @param phoneNumber
     */
    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    /**
     * 获取
     * @return emailAddress
     */
    public String getEmailAddress() {
        return emailAddress;
    }

    /**
     * 设置
     * @param emailAddress
     */
    public void setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public String toString() {
        return "personType.Person{name = " + name + ", address = " + address + ",
phoneNumber = " + phoneNumber + ", emailAddress = " + emailAddress + "}";
    }

    public boolean fileWrite(String fileName) {
        //先读取原来文件中的信息,再加上现在对象的信息,最后再重新输出
        String content=fileRead(fileName)+this.toString();
        File writeName = new File(fileName);
        try {
            BufferedWriter out = new BufferedWriter(new FileWriter(writeName));
            out.write(content);
            out.flush();
            out.close();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        return true;
    }

    public  String fileRead(String pathName) {
        File fileName = new File(pathName);
        String line = "";
        try {
```

```
            InputStreamReader reader = new InputStreamReader(new
FileInputStream(fileName));
            BufferedReader br = new BufferedReader(reader);

            String tempLine="";
            while (tempLine != null) {
                tempLine = br.readLine();
                if (tempLine != null) {
                    line += tempLine;
                    line += "\n";
                }

            }

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        return line;
    }
}
```

```
package personType;

public class Student extends Person {
    private final static String grade1="freshman";
    private final static String grade2="sophomore";
    private final static String grade3="junior";
    private final static String grade4="senior";
    public Student(){}
    public Student(String name,String address,String phoneNumber,String
emailAddress){
        super(name,address,phoneNumber,emailAddress);
    }

    public String toString() {
        return "class:personType.Student,name:" + this.getName();
    }
}
```

```
package personType;

public abstract class Employee extends Person {
    private String office;
    private String salary;
    private MyData data;

    public Employee() {
    }

    public Employee(String name, String address, String phoneNumber, String
emailAddress, String office, String salary, MyData data) {
        super(name, address, phoneNumber, emailAddress);
```

```
        this.office = office;
        this.salary = salary;
        this.data = data;
    }

    /**
     * 获取
     * @return office
     */
    public String getOffice() {
        return office;
    }

    /**
     * 设置
     * @param office
     */
    public void setOffice(String office) {
        this.office = office;
    }

    /**
     * 获取
     * @return salary
     */
    public String getSalary() {
        return salary;
    }

    /**
     * 设置
     * @param salary
     */
    public void setSalary(String salary) {
        this.salary = salary;
    }

    /**
     * 获取
     * @return data
     */
    public MyData getData() {
        return data;
    }

    /**
     * 设置
     * @param data
     */
    public void setData(MyData data) {
        this.data = data;
    }

    public String toString() {
        return "personType.Employee{office = " + office + ", salary = " + salary
+ ", data = " + data + "}";
```

```
    }
}
```

```
package personType;

public class Faculty extends Employee {
    private String workTime;
    private String rank;

    public Faculty() {
    }

    public Faculty(String name, String address, String phoneNumber, String
emailAddress, String office, String salary, MyData data, String workTime, String
rank) {
        super(name, address, phoneNumber, emailAddress, office, salary, data);
        this.workTime = workTime;
        this.rank = rank;
    }

    /**
     * 获取
     * @return workTime
     */
    public String getWorkTime() {
        return workTime;
    }

    /**
     * 设置
     * @param workTime
     */
    public void setWorkTime(String workTime) {
        this.workTime = workTime;
    }

    /**
     * 获取
     * @return rank
     */
    public String getRank() {
        return rank;
    }

    /**
     * 设置
     * @param rank
     */
    public void setRank(String rank) {
        this.rank = rank;
    }

    public String toString() {
        return "class:personType.Faculty,name:" + this.getName();
    }
```

```
}
```

```
package personType;

public class Staff extends Employee {
    private String title;

    public Staff() {
    }

    public Staff(String name, String address, String phoneNumber, String
emailAddress, String office, String salary, MyData data, String title) {
        super(name, address, phoneNumber, emailAddress, office, salary, data);
        this.title = title;
    }

    /**
     * 获取
     * @return title
     */
    public String getTitle() {
        return title;
    }

    /**
     * 设置
     * @param title
     */
    public void setTitle(String title) {
        this.title = title;
    }

    public String toString() {
        return "class:personType.Staff,name:" + this.getName();
    }
}
```

```
package personType;

public class Postgraduate extends Student {
    private String searchField;
    private String tutor;

    public Postgraduate() {
    }

    public Postgraduate(String name, String address, String phoneNumber, String
emailAddress, String searchField, String tutor) {
        super(name, address, phoneNumber, emailAddress);
        this.searchField = searchField;
        this.tutor = tutor;
    }

    /**
```

```
     * 获取
     * @return searchField
     */
    public String getSearchField() {
        return searchField;
    }

    /**
     * 设置
     * @param searchField
     */
    public void setSearchField(String searchField) {
        this.searchField = searchField;
    }

    /**
     * 获取
     * @return tutor
     */
    public String getTutor() {
        return tutor;
    }

    /**
     * 设置
     * @param tutor
     */
    public void setTutor(String tutor) {
        this.tutor = tutor;
    }

    public String toString() {
        return "class:personType.Postgraduate,name:" + this.getName();
    }
}
```

```
package personType;

public class MyData {
    private String year;
    private String month;
    private String day;

    public MyData() {
    }

    public MyData(String year, String month, String day) {
        this.year = year;
        this.month = month;
        this.day = day;
    }

    /**
     * 获取
     * @return year
```

```
     */
    public String getYear() {
        return year;
    }

    /**
     * 设置
     * @param year
     */
    public void setYear(String year) {
        this.year = year;
    }

    /**
     * 获取
     * @return month
     */
    public String getMonth() {
        return month;
    }

    /**
     * 设置
     * @param month
     */
    public void setMonth(String month) {
        this.month = month;
    }

    /**
     * 获取
     * @return day
     */
    public String getDay() {
        return day;
    }

    /**
     * 设置
     * @param day
     */
    public void setDay(String day) {
        this.day = day;
    }

    public String toString() {
        return "personType.MyData{year = " + year + ", month = " + month + ", day
= " + day + "}";
    }
}
```

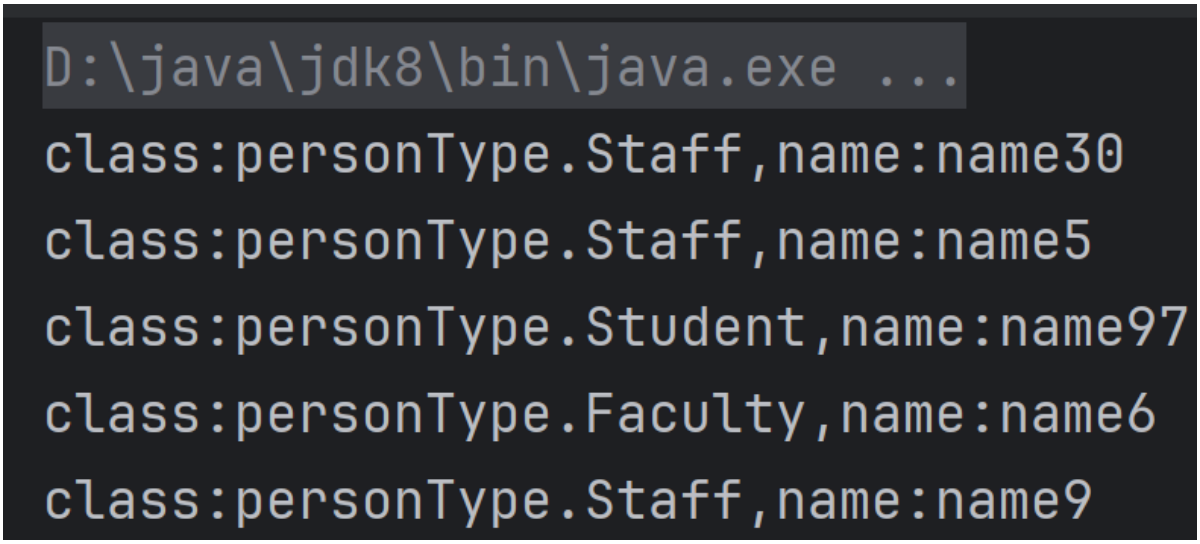
测试类

```
package Test;

import personType.*;

import java.util.Random;


public class Test1 {
    public static final String FILENAME = "person.txt";

    public static void main(String[] args) {
        //personType.Student personType.Faculty staff
        //MyDate(year,month,day)
        Random rand = new Random();
        int capacity = rand.nextInt(500);
        Person[] persons = new Person[capacity];
        for (int i = 0; i < capacity; i++) {
            //1 personType.Student
            //2 personType.Faculty
            //3 personType.Staff
            int type = 1 + rand.nextInt(3);
            switch (type) {
                case 1: {
                    //personType.Student
                    persons[i] = new Student(getName(), getAddress(),
getPhoneNumber(), getEmailAddress());
                    break;
                }
                case 2: {
                    //personType.Faculty
                    persons[i] = new Faculty(getName(), getAddress(),
getPhoneNumber(), getEmailAddress(), getOffice(), getSalary(), getMyData(),
getWorkTime(), getRank());
                    break;
                }
                case 3: {
                    //personType.Staff
                    persons[i] = new Staff(getName(), getAddress(),
getPhoneNumber(), getEmailAddress(), getOffice(), getSalary(), getMyData(),
getTitle());
                    break;
                }
            }
        }
        //打印
        printMessages(persons);
    }

    public static void printMessages(Person[] persons) {
        for (Person person : persons) {
            System.out.println(person);
        }
    }
```

```
    //生成随机日期
    public static MyData getMyData() {
        Random rand = new Random();
        //生成随机年份
        String year = 2000 + rand.nextInt(25) + "";
        int m = rand.nextInt(13);
        String month = m + "";
        String day = "";
        if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12)
{
            //一个月有31天
            day = rand.nextInt(32) + "";
        } else {
            //一个月有30天
            day = rand.nextInt(31) + "";
        }
        return new MyData(year, month, day);
    }

    //生成随机名字
    public static String getName() {
        return "name" + new Random().nextInt(100);
    }

    //生成随机地址
    public static String getAddress() {
        return "address" + new Random().nextInt(100);
    }

    //生成随机电话
    public static String getPhoneNumber() {
        return "phoneNumber" + new Random().nextInt(100);
    }

    //生成随机邮箱
    public static String getEmailAddress() {
        return "emailAddress" + new Random().nextInt(100);
    }

    //升成随机办公室
    public static String getOffice() {
        return "office" + new Random().nextInt(100);
    }

    //生成随机薪资
    public static String getSalary() {
        return "salary" + new Random().nextInt(100);
    }

    //生成随机办公时间
    public static String getWorkTime() {
        return "time" + new Random().nextInt(100);
    }
```

```
    //生成随机等级
    public static String getRank() {
        return "rank" + new Random().nextInt(100);
    }

    //生成随机职称
    public static String getTitle() {
        return "title" + new Random().nextInt(100);
    }


}
```

输出效果



version 2

题目要求:

在上面实现类的基础上，为每个类增加一个将当前对象序列化到指定文件的方法writeToFile(File f)。为Student 类创建一个新的子类
Postgraduate，有研究方向和导师姓名两个新增数据域。编写测试类 Test2，要求随机产生不同的对象，并将这些对象序列化到指

定的文件中，并用方法 printMessages(Person[] persons)将对象的信息输出。

测试类

```
package Test;

import personType.*;

import java.util.Random;


public class Test2 {
    public static final String FILENAME = "person.txt";

    public static void main(String[] args) {
        //personType.Student personType.Faculty staff
        //MyDate(year,month,day)
```

```
        Random rand = new Random();
        int capacity = rand.nextInt(500);
        Person[] persons = new Person[capacity];
        for (int i = 0; i < capacity; i++) {
            //1 personType.Student
            //2 personType.Faculty
            //3 personType.Staff
            //4 personType.Postgraduate
            int type = 1 + rand.nextInt(4);
            switch (type) {
                case 1: {
                    //personType.Student
                    persons[i] = new Student(getName(), getAddress(),
getPhoneNumber(), getEmailAddress());
                    break;
                }
                case 2: {
                    //personType.Faculty
                    persons[i] = new Faculty(getName(), getAddress(),
getPhoneNumber(), getEmailAddress(), getOffice(), getSalary(), getMyData(),
getWorkTime(), getRank());
                    break;
                }
                case 3: {
                    //personType.Staff
                    persons[i] = new Staff(getName(), getAddress(),
getPhoneNumber(), getEmailAddress(), getOffice(), getSalary(), getMyData(),
getTitle());
                    break;
                }
                case 4: {
                    //personType.Postgraduate
                    persons[i] = new Postgraduate(getName(), getAddress(),
getPhoneNumber(), getEmailAddress(), getResearchField(), getTutor());
                }
            }
        }
        //打印
        printMessages(persons);
        //将对象信息输出至person.txt
        for (Person person : persons) {
            person.fileWrite(FILENAME);
        }
        System.out.println(capacity);
    }

    public static void printMessages(Person[] persons) {
        for (Person person : persons) {
            System.out.println(person);
        }
    }


    //生成随机日期
    public static MyData getMyData() {
        Random rand = new Random();
```

```
        //生成随机年份
        String year = 2000 + rand.nextInt(25) + "";
        int m = rand.nextInt(13);
        String month = m + "";
        String day = "";
        if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12)
{
            //一个月有31天
            day = rand.nextInt(32) + "";
        } else {
            //一个月有30天
            day = rand.nextInt(31) + "";
        }
        return new MyData(year, month, day);
    }

    //生成随机名字
    public static String getName() {
        return "name" + new Random().nextInt(100);
    }

    //生成随机地址
    public static String getAddress() {
        return "address" + new Random().nextInt(100);
    }

    //生成随机电话
    public static String getPhoneNumber() {
        return "phoneNumber" + new Random().nextInt(100);
    }

    //生成随机邮箱
    public static String getEmailAddress() {
        return "emailAddress" + new Random().nextInt(100);
    }

    //升成随机办公室
    public static String getOffice() {
        return "office" + new Random().nextInt(100);
    }

    //生成随机薪资
    public static String getSalary() {
        return "salary" + new Random().nextInt(100);
    }

    //生成随机办公时间
    public static String getWorkTime() {
        return "time" + new Random().nextInt(100);
    }

    //生成随机等级
    public static String getRank() {
        return "rank" + new Random().nextInt(100);
    }
```

```
    //生成随机职称
    public static String getTitle() {
        return "title" + new Random().nextInt(100);
    }

    //生成随机研究方向
    public static String getResearchField() {
        return "research" + new Random().nextInt(100);
    }

    //生成随机导师姓名
    public static String getTutor() {
        return "tutor" + new Random().nextInt(100);
    }

}
```

输出文件

person.txt

```
class:personType.Faculty,name:name0
class:personType.Staff,name:name46
class:personType.Student,name:name31
class:personType.Staff,name:name41
class:personType.Student,name:name59
class:personType.Faculty,name:name51
class:personType.Staff,name:name50
class:personType.Student,name:name70
class:personType.Postgraduate,name:name10
class:personType.Student,name:name26
class:personType.Student,name:name32
class:personType.Faculty,name:name62
class:personType.Postgraduate,name:name4
class:personType.Postgraduate,name:name19
class:personType.Postgraduate,name:name79
class:personType.Postgraduate,name:name14
class:personType.Staff,name:name1
class:personType.Student,name:name74
class:personType.Faculty,name:name60
class:personType.Staff,name:name9
class:personType.Postgraduate,name:name75
class:personType.Staff,name:name38
class:personType.Postgraduate,name:name90
class:personType.Postgraduate,name:name42
class:personType.Postgraduate,name:name38
class:personType.Postgraduate,name:name11
class:personType.Staff,name:name84
class:personType.Faculty,name:name60
class:personType.Faculty,name:name60
class:personType.Staff,name:name97
class:personType.Student,name:name77
class:personType.Student,name:name23
class:personType.Faculty,name:name93
class:personType.Faculty,name:name57
```

```
class:personType.Faculty,name:name26
class:personType.Faculty,name:name36
class:personType.Staff,name:name70
class:personType.Faculty,name:name45
class:personType.Student,name:name34
```

version 3

题目要求:

在版本 2 的基础上设计实现一个具有 GUI 界面的人员信息管理系统，要求实现基本的人员增、删、改、查的功能，人员信息列表应采用
TableView 组件。

测试类

```
package Test;

import javafx.geometry.Side;
import javafx.scene.control.*;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.layout.*;
import personType.*;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;

import java.util.ArrayList;
import java.util.Random;


public class Test3 extends Application {
    //每个按钮和textField的宽度
    public static final int deleteButtonHeight = 24;

    //需要用到的button
    //为了方便管理可以用HBox/VBox(写这个的时候不知道这个知识点)
    Button delete1 = new Button("删除");
    Button delete2 = new Button("删除");
    Button delete3 = new Button("删除");
    Button delete4 = new Button("删除");
    Button delete5 = new Button("删除");
    Button delete6 = new Button("删除");
    Button delete7 = new Button("删除");
    Button delete8 = new Button("删除");
    Button delete9 = new Button("删除");
    Button delete10 = new Button("删除");
    Button delete11 = new Button("删除");
    Button delete12 = new Button("删除");
    Button delete13 = new Button("删除");
```

```
    Button delete14 = new Button("删除");
    Button delete15 = new Button("删除");

    Button search1 = new Button("Student");
    Button search2 = new Button("Faculty");
    Button search3 = new Button("Staff");
    Button search4 = new Button("Postgraduate");

    Button modify1 = new Button("修改");
    Button modify2 = new Button("修改");
    Button modify3 = new Button("修改");
    Button modify4 = new Button("修改");
    Button modify5 = new Button("修改");
    Button modify6 = new Button("修改");
    Button modify7 = new Button("修改");
    Button modify8 = new Button("修改");
    Button modify9 = new Button("修改");
    Button modify10 = new Button("修改");
    Button modify11 = new Button("修改");
    Button modify12 = new Button("修改");
    Button modify13 = new Button("修改");
    Button modify14 = new Button("修改");
    Button modify15 = new Button("修改");

    Button confirmButton = new Button("确定修改");

    //修改人物信息时需要用到的文本输入框
    TextField nameField = new TextField();
    TextField addressField = new TextField();
    TextField phoneNumberField = new TextField();
    TextField emailAddressField = new TextField();
    TextField tutorField = new TextField();
    TextField searchField = new TextField();
    TextField titleField = new TextField();
    TextField workTimeField = new TextField();
    TextField rankField = new TextField();
    TextField salaryField = new TextField();
    TextField officeField = new TextField();

    //设置为全局变量,存储要修改的行数
    int modifyChoice;


    public static void main(String[] args) {
        //personType.Student personType.Faculty staff
        //MyDate(year,month,day)
        launch(args);

    }

    //获取随机人物列表
    public static ArrayList<Person> gerPersonList() {
        Random rand = new Random();
        int capacity = 30 + rand.nextInt(500);
```

```
        ArrayList<Person> persons = new ArrayList<>();
        for (int i = 0; i < capacity; i++) {
            //1 personType.Student
            //2 personType.Faculty
            //3 personType.Staff
            //4 personType.Postgraduate
            int type = 1 + rand.nextInt(4);
            switch (type) {
                case 1: {
                    //personType.Student
                    persons.add(new Student("Student" + getName(), getAddress(),
getPhoneNumber(), getEmailAddress()));
                    break;
                }
                case 2: {
                    //personType.Faculty
                    persons.add(new Faculty("Faculty" + getName(), getAddress(),
getPhoneNumber(), getEmailAddress(), getOffice(), getSalary(), getMyData(),
getWorkTime(), getRank()));
                    break;
                }
                case 3: {
                    //personType.Staff
                    persons.add(new Staff("Staff" + getName(), getAddress(),
getPhoneNumber(), getEmailAddress(), getOffice(), getSalary(), getMyData(),
getTitle()));
                    break;
                }
                case 4: {
                    //personType.Postgraduate
                    persons.add(new Postgraduate("Postgraduate" + getName(),
getAddress(), getPhoneNumber(), getEmailAddress(), getResearchField(),
getTutor()));
                }
            }
        }
        return persons;
    }

    //生成随机日期
    public static MyData getMyData() {
        Random rand = new Random();
        //生成随机年份
        String year = 2000 + rand.nextInt(25) + "";
        int m = rand.nextInt(13);
        String month = m + "";
        String day = "";
        if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12)
{
            //一个月有31天
            day = rand.nextInt(32) + "";
        } else {
            //一个月有30天
            day = rand.nextInt(31) + "";
        }
        return new MyData(year, month, day);
```

```
    }

    //生成随机名字
    public static String getName() {
        return "" + new Random().nextInt(100);
    }

    //生成随机地址
    public static String getAddress() {
        return "address" + new Random().nextInt(100);
    }

    //生成随机电话
    public static String getPhoneNumber() {
        return "" + new Random().nextInt(100000);
    }

    //生成随机邮箱
    public static String getEmailAddress() {
        return "emailAddress" + new Random().nextInt(100);
    }

    //升成随机办公室
    public static String getOffice() {
        return "office" + new Random().nextInt(100);
    }

    //生成随机薪资
    public static String getSalary() {
        return "salary" + new Random().nextInt(100);
    }

    //生成随机办公时间
    public static String getWorkTime() {
        return "time" + new Random().nextInt(100);
    }

    //生成随机等级
    public static String getRank() {
        return "rank" + new Random().nextInt(100);
    }

    //生成随机职称
    public static String getTitle() {
        return "title" + new Random().nextInt(100);
    }

    //生成随机研究方向
    public static String getResearchField() {
        return "research" + new Random().nextInt(100);
    }

    //生成随机导师姓名
    public static String getTutor() {
        return "tutor" + new Random().nextInt(100);
    }
```

```
    @Override
    public void start(Stage primaryStage) throws Exception {
        //获取列表.里面内容为随机产生的人物
        ArrayList<Person> persons = gerPersonList();
        //tableView可以使用的list
        ObservableList<Person> list = FXCollections.observableArrayList();
        //随机产生的人物列表内容加入list
        list.addAll(persons);


        //初始化界面
        //把list传给tableView
        TableView<Person> tableView = new TableView<>(list);
        //生成表头,并加入tableView
        //personType.Person成员
        TableColumn<Person, String> tc_name = new TableColumn<>("姓名");
        tableView.getColumns().add(tc_name);
        TableColumn<Person, String> tc_address = new TableColumn<>("地址");
        tableView.getColumns().add(tc_address);
        TableColumn<Person, String> tc_phoneNumber = new TableColumn<>("电话");
        tableView.getColumns().add(tc_phoneNumber);
        TableColumn<Person, String> tc_emailAddress = new TableColumn<>("邮箱");
        tableView.getColumns().add(tc_emailAddress);
        //personType.Postgraduate成员
        TableColumn<Person, String> tc_tutor = new TableColumn<>("导师");
        tableView.getColumns().add(tc_tutor);
        TableColumn<Person, String> tc_searchField = new TableColumn<>("研究方
向");
        tableView.getColumns().add(tc_searchField);
        //personType.Staff成员
        TableColumn<Person, String> tc_title = new TableColumn<>("职称");
        tableView.getColumns().add(tc_title);
        //personType.Faculty成员
        TableColumn<Person, String> tc_workTime = new TableColumn<>("工作时间");
        tableView.getColumns().add(tc_workTime);
        TableColumn<Person, String> tc_rank = new TableColumn<>("等级");
        tableView.getColumns().add(tc_rank);
        //p]ersonType.employee成员
        TableColumn<Person, String> tc_salary = new TableColumn<>("薪资");
        tableView.getColumns().add(tc_salary);
        TableColumn<Person, String> tc_office = new TableColumn<>("办公室");
        tableView.getColumns().add(tc_office);
        TableColumn<Person, String> tc_data = new TableColumn<>("入职日期");
        tableView.getColumns().add(tc_data);

        //把对应的内容加入相应的列
        //personType.person
        tc_name.setCellValueFactory(new PropertyValueFactory<Person, String>
("name"));
        tc_address.setCellValueFactory(new PropertyValueFactory<Person, String>
("address"));
        tc_phoneNumber.setCellValueFactory(new PropertyValueFactory<Person,
String>("phoneNumber"));
        tc_emailAddress.setCellValueFactory(new PropertyValueFactory<Person,
String>("emailAddress"));
```

```
        //personType.Postgraduate
        tc_tutor.setCellValueFactory(new PropertyValueFactory<Person, String>
("tutor"));
        tc_searchField.setCellValueFactory(new PropertyValueFactory<Person,
String>("searchField"));
        //personType.Staff
        tc_title.setCellValueFactory(new PropertyValueFactory<Person, String>
("title"));
        //personType.Faculty
        tc_workTime.setCellValueFactory(new PropertyValueFactory<Person, String>
("workTime"));
        tc_rank.setCellValueFactory(new PropertyValueFactory<Person, String>
("rank"));
        //personType.employee
        tc_salary.setCellValueFactory(new PropertyValueFactory<Person, String>
("salary"));
        tc_office.setCellValueFactory(new PropertyValueFactory<Person, String>
("office"));
        tc_data.setCellValueFactory(new PropertyValueFactory<Person, String>
("data"));

        //AnchorPane对象
        AnchorPane ap = new AnchorPane();
        //把设置好的tableView加入ap
        ap.getChildren().addAll(tableView);

        //增
        addFunction(list, tableView, ap);
        //删
        deleteFunction(list, tableView, ap);
        //查
        searchFunction(list, tableView, ap);
        //改
        modifyFunction(list, ap, tableView);
        //退出系统
        exitFunction(ap);


        //设置cqut背景

        //创建背景图片
        Image image = new
Image("file:D:\\java\\javaDesign\\src\\Test\\cqut.png");
        //创建背景,放在右下角
        BackgroundImage backgroundImage = new BackgroundImage(image,
                BackgroundRepeat.NO_REPEAT,
                BackgroundRepeat.NO_REPEAT,
                new BackgroundPosition(
                        Side.RIGHT, 0, true, Side.BOTTOM, 0, true),
                BackgroundSize.DEFAULT);

        // 设置背景
        ap.setBackground(new Background(backgroundImage));

        //构造ap的Scene
        Scene scene = new Scene(ap);
```

```
        //设置primaryStage
        primaryStage.setScene(scene);
        primaryStage.setHeight(640);  //高
        primaryStage.setWidth(1200);  //宽
        primaryStage.setAlwaysOnTop(true);//始终显示

        primaryStage.show();

    }

    //查询功能
    private void searchFunction(ObservableList<Person> list, TableView<Person>
tableView, AnchorPane ap) {
        //查询按钮
        Button bu_search = new Button("查找人物");
        //设置位置
        bu_search.setLayoutX(600);
        bu_search.setLayoutY(450);
        //设置点击事件
        bu_search.setOnAction(event -> {
            //隐藏不相关的组件
            setModifyButtonVisual(false);
            setDeleteButtonVisual(false);
            setTextFieldNotVisible();
            //设置四种查询类型按钮的位置,再查询按钮的正下方
            search1.setLayoutX(600);
            search2.setLayoutX(600);
            search3.setLayoutX(600);
            search4.setLayoutX(600);
            search1.setLayoutY(450 + 1 * deleteButtonHeight);
            search2.setLayoutY(450 + 2 * deleteButtonHeight);
            search3.setLayoutY(450 + 3 * deleteButtonHeight);
            search4.setLayoutY(450 + 4 * deleteButtonHeight);
            //显示查询按钮
            setSearchButtonVisual(true);
            //设置点击事件
            search1.setOnAction(event1 -> {
                for (int j = 0; j < 100; j++) {   //不知道为什么一次并不能完全筛选,在这
里我设置筛选100词
                    //查询逻辑就是把不是该类型的删除
                    for (int i = 0; i < list.size(); i++) {
                        if (!(list.get(i) instanceof Student)) {
                            list.remove(i);
                        }
                    }
                }
                tableView.refresh();
                System.out.println("查询成功");
            });

            search2.setOnAction(event1 -> {
                for (int j = 0; j < 100; j++) {
                    for (int i = 0; i < list.size(); i++) {
                        if (!(list.get(i) instanceof Faculty)) {
                            list.remove(i);
```

```
                        }
                    }
                }
                tableView.refresh();
            });

            search3.setOnAction(event1 -> {
                for (int j = 0; j < 100; j++) {
                    for (int i = 0; i < list.size(); i++) {
                        if (!(list.get(i) instanceof Staff)) {
                            list.remove(i);
                        }
                    }
                }
                tableView.refresh();
            });

            search4.setOnAction(event1 -> {
                for (int j = 0; j < 100; j++) {
                    for (int i = 0; i < list.size(); i++) {
                        if (!(list.get(i) instanceof Postgraduate)) {
                            list.remove(i);
                        }
                    }
                }
                tableView.refresh();
            });

            //添加查询按钮
            ap.getChildren().add(search1);
            ap.getChildren().add(search2);
            ap.getChildren().add(search3);
            ap.getChildren().add(search4);
            tableView.refresh();

        });
        ap.getChildren().add(bu_search);
    }

    //修改功能
    private void modifyFunction(ObservableList<Person> list, AnchorPane ap,
TableView<Person> tableView) {
        //修改按钮
        Button bu_modify = new Button("修改信息");
        //确定按钮,修改完后点击
        //设置位置
        confirmButton.setLayoutX(1040);
        confirmButton.setLayoutY(22 + 17 * deleteButtonHeight);
        //设置点击事件
        confirmButton.setOnAction(event -> {
            Person temp = list.get(modifyChoice);
            //姓名
            nameField.setVisible(true);
            temp.setName(nameField.getText());
            //地址
            addressField.setVisible(true);
```

```
            temp.setAddress(addressField.getText());
            //电话
            phoneNumberField.setVisible(true);
            temp.setPhoneNumber(phoneNumberField.getText());
            //邮箱
            emailAddressField.setVisible(true);
            temp.setEmailAddress(emailAddressField.getText());
            if (temp instanceof Postgraduate) {
                //导师
                tutorField.setVisible(true);
                ((Postgraduate) temp).setTutor(tutorField.getText());
                //研究方向
                searchField.setVisible(true);
                ((Postgraduate) temp).setSearchField(searchField.getText());
            }
            if (temp instanceof Employee) {
                //办公室
                officeField.setVisible(true);
                ((Employee) temp).setOffice(officeField.getText());
                //工资
                salaryField.setVisible(true);
                ((Employee) temp).setSalary(salaryField.getText());
            }
            if (temp instanceof Staff) {
                //职称
                titleField.setVisible(true);
                ((Staff) temp).setTitle(titleField.getText());
            }
            if (temp instanceof Faculty) {
                //上班时间
                workTimeField.setVisible(true);
                ((Faculty) temp).setWorkTime(workTimeField.getText());
                //等级
                rankField.setVisible(true);
                ((Faculty) temp).setRank(rankField.getText());
            }
            System.out.println("修改成功");
            list.set(modifyChoice, temp);
        });

        //设置按钮不可见,只有当行修改按钮点击后才可见
        confirmButton.setVisible(false);
        ap.getChildren().add(confirmButton);

        //设置总修改按钮位置
        bu_modify.setLayoutX(700);
        bu_modify.setLayoutY(450);

        //每行设置修改按钮
        modify1.setLayoutX(975);
        modify2.setLayoutX(975);
        modify3.setLayoutX(975);
        modify4.setLayoutX(975);
        modify5.setLayoutX(975);
        modify6.setLayoutX(975);
        modify7.setLayoutX(975);
```

```
        modify8.setLayoutX(975);
        modify9.setLayoutX(975);
        modify10.setLayoutX(975);
        modify11.setLayoutX(975);
        modify12.setLayoutX(975);
        modify13.setLayoutX(975);
        modify14.setLayoutX(975);
        modify15.setLayoutX(975);

        modify1.setLayoutY(22 + 0 * deleteButtonHeight);
        modify2.setLayoutY(22 + 1 * deleteButtonHeight);
        modify3.setLayoutY(22 + 2 * deleteButtonHeight);
        modify4.setLayoutY(22 + 3 * deleteButtonHeight);
        modify5.setLayoutY(22 + 4 * deleteButtonHeight);
        modify6.setLayoutY(22 + 5 * deleteButtonHeight);
        modify7.setLayoutY(22 + 6 * deleteButtonHeight);
        modify8.setLayoutY(22 + 7 * deleteButtonHeight);
        modify9.setLayoutY(22 + 8 * deleteButtonHeight);
        modify10.setLayoutY(22 + 9 * deleteButtonHeight);
        modify11.setLayoutY(22 + 10 * deleteButtonHeight);
        modify12.setLayoutY(22 + 11 * deleteButtonHeight);
        modify13.setLayoutY(22 + 12 * deleteButtonHeight);
        modify14.setLayoutY(22 + 13 * deleteButtonHeight);
        modify15.setLayoutY(22 + 14 * deleteButtonHeight);


        //设置事件
        setPerModifyButtonEvent(list, modify1, 0);
        setPerModifyButtonEvent(list, modify2, 1);
        setPerModifyButtonEvent(list, modify3, 2);
        setPerModifyButtonEvent(list, modify4, 3);
        setPerModifyButtonEvent(list, modify5, 4);
        setPerModifyButtonEvent(list, modify6, 5);
        setPerModifyButtonEvent(list, modify7, 6);
        setPerModifyButtonEvent(list, modify8, 7);
        setPerModifyButtonEvent(list, modify9, 8);
        setPerModifyButtonEvent(list, modify10, 9);
        setPerModifyButtonEvent(list, modify11, 10);
        setPerModifyButtonEvent(list, modify12, 11);
        setPerModifyButtonEvent(list, modify13, 12);
        setPerModifyButtonEvent(list, modify14, 13);
        setPerModifyButtonEvent(list, modify15, 14);

        //添加行修改按钮
        ap.getChildren().add(modify1);
        ap.getChildren().add(modify2);
        ap.getChildren().add(modify3);
        ap.getChildren().add(modify4);
        ap.getChildren().add(modify5);
        ap.getChildren().add(modify6);
        ap.getChildren().add(modify7);
        ap.getChildren().add(modify8);
        ap.getChildren().add(modify9);
        ap.getChildren().add(modify10);
        ap.getChildren().add(modify11);
        ap.getChildren().add(modify12);
```

```
        ap.getChildren().add(modify13);
        ap.getChildren().add(modify14);
        ap.getChildren().add(modify15);


        setModifyButtonVisual(false);

        nameField.setLayoutX(1025);
        addressField.setLayoutX(1025);
        phoneNumberField.setLayoutX(1025);
        emailAddressField.setLayoutX(1025);
        tutorField.setLayoutX(1025);
        searchField.setLayoutX(1025);
        titleField.setLayoutX(1025);
        workTimeField.setLayoutX(1025);
        rankField.setLayoutX(1025);
        salaryField.setLayoutX(1025);
        officeField.setLayoutX(1025);

        //设置修改内容文本输入框位置
        nameField.setLayoutY(22 + 0 * deleteButtonHeight);
        addressField.setLayoutY(22 + 1 * deleteButtonHeight);
        phoneNumberField.setLayoutY(22 + 2 * deleteButtonHeight);
        emailAddressField.setLayoutY(22 + 3 * deleteButtonHeight);
        tutorField.setLayoutY(22 + 4 * deleteButtonHeight);
        searchField.setLayoutY(22 + 5 * deleteButtonHeight);
        titleField.setLayoutY(22 + 6 * deleteButtonHeight);
        workTimeField.setLayoutY(22 + 7 * deleteButtonHeight);
        rankField.setLayoutY(22 + 8 * deleteButtonHeight);
        salaryField.setLayoutY(22 + 9 * deleteButtonHeight);
        officeField.setLayoutY(22 + 10 * deleteButtonHeight);

        //设置输入框提示背景
        nameField.setPromptText("请输入姓名");
        addressField.setPromptText("请输入地址");
        phoneNumberField.setPromptText("请输入电话");
        emailAddressField.setPromptText("请输入邮箱");
        tutorField.setPromptText("请输入导师姓名");
        searchField.setPromptText("请输入研究方向");
        titleField.setPromptText("请输入职称");
        workTimeField.setPromptText("请输入工作时间");
        rankField.setPromptText("请输入等级");
        salaryField.setPromptText("请输入薪资");
        officeField.setPromptText("请输入办公室");

        //设置文本输入框不可见
        setTextFieldNotVisible();

        //添加文本输入框
        ap.getChildren().add(nameField);
        ap.getChildren().add(addressField);
        ap.getChildren().add(phoneNumberField);
        ap.getChildren().add(emailAddressField);
        ap.getChildren().add(tutorField);
        ap.getChildren().add(searchField);
        ap.getChildren().add(titleField);
```

```
        ap.getChildren().add(workTimeField);
        ap.getChildren().add(rankField);
        ap.getChildren().add(salaryField);
        ap.getChildren().add(officeField);

        //总修改按钮事件
        bu_modify.setOnAction(event -> {
            setModifyButtonVisual(true);
            setDeleteButtonVisual(false);
            setSearchButtonVisual(false);
            tableView.refresh();
        });
        ap.getChildren().add(bu_modify);
    }

    private static void exitFunction(AnchorPane ap) {
        //退出按钮
        Button bu_exit = new Button("退出");
        //设置按钮位置
        bu_exit.setLayoutX(800);
        bu_exit.setLayoutY(450);
        //设置按钮事件
        bu_exit.setOnAction(event -> {
            //结束jvm
            System.exit(0);
        });
        ap.getChildren().add(bu_exit);
    }

    //删除功能
    private void deleteFunction(ObservableList<Person> list, TableView<Person>
tableView, AnchorPane ap) {
        //删除按钮
        Button bu_delete = new Button("删除人物");
        bu_delete.setLayoutX(500);
        bu_delete.setLayoutY(450);
        //设置行删除按钮
        setDelete_bu(list, tableView, ap, 975, 22 + 0 * deleteButtonHeight, 0,
delete1);
        setDelete_bu(list, tableView, ap, 975, 22 + 1 * deleteButtonHeight, 1,
delete2);
        setDelete_bu(list, tableView, ap, 975, 22 + 2 * deleteButtonHeight, 2,
delete3);
        setDelete_bu(list, tableView, ap, 975, 22 + 3 * deleteButtonHeight, 3,
delete4);
        setDelete_bu(list, tableView, ap, 975, 22 + 4 * deleteButtonHeight, 4,
delete5);
        setDelete_bu(list, tableView, ap, 975, 22 + 5 * deleteButtonHeight, 5,
delete6);
        setDelete_bu(list, tableView, ap, 975, 22 + 6 * deleteButtonHeight, 6,
delete7);
        setDelete_bu(list, tableView, ap, 975, 22 + 7 * deleteButtonHeight, 7,
delete8);
        setDelete_bu(list, tableView, ap, 975, 22 + 8 * deleteButtonHeight, 8,
delete9);
```

```
        setDelete_bu(list, tableView, ap, 975, 22 + 9 * deleteButtonHeight, 9,
delete10);
        setDelete_bu(list, tableView, ap, 975, 22 + 10 * deleteButtonHeight, 10,
delete11);
        setDelete_bu(list, tableView, ap, 975, 22 + 11 * deleteButtonHeight, 11,
delete12);
        setDelete_bu(list, tableView, ap, 975, 22 + 12 * deleteButtonHeight, 12,
delete13);
        setDelete_bu(list, tableView, ap, 975, 22 + 13 * deleteButtonHeight, 13,
delete14);
        setDelete_bu(list, tableView, ap, 975, 22 + 14 * deleteButtonHeight, 14,
delete15);
        //设置不可见,当点击总删除按钮时可见
        setDeleteButtonVisual(false);
        //总删除按钮点击事件
        bu_delete.setOnAction(event -> {
            //设置无关组件不可见
            setSearchButtonVisual(false);
            setModifyButtonVisual(false);
            //设置相关组件可见
            setDeleteButtonVisual(true);
            setTextFieldNotVisible();

        });
        ap.getChildren().add(bu_delete);
    }

    //增加功能
    private void addFunction(ObservableList<Person> list, TableView<Person>
tableView, AnchorPane ap) {
        //添加按钮
        Button bu_add = new Button("添加人物");
        //设置按钮位置
        bu_add.setLayoutX(400);
        bu_add.setLayoutY(450);
        //设置按钮事件
        bu_add.setOnAction(event -> {
            //设置其他无关组件不可见
            setSearchButtonVisual(false);
            setDeleteButtonVisual(false);
            setModifyButtonVisual(false);
            setTextFieldNotVisible();
            //获取随机人物对象并添加至list
            list.add(gerPersonList().get(0));
            System.out.println("添加成功");
            tableView.refresh();
        });

        ap.getChildren().add(bu_add);
    }

    private void setTextFieldNotVisible() {
        nameField.setVisible(false);
        addressField.setVisible(false);
        phoneNumberField.setVisible(false);
        emailAddressField.setVisible(false);
```

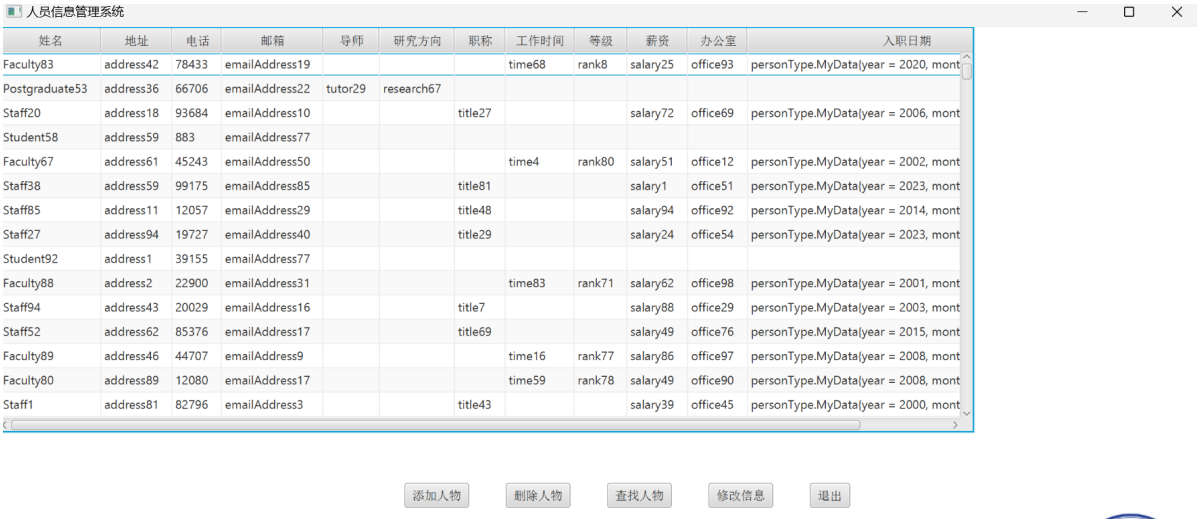
```
        tutorField.setVisible(false);
        searchField.setVisible(false);
        titleField.setVisible(false);
        workTimeField.setVisible(false);
        rankField.setVisible(false);
        salaryField.setVisible(false);
        officeField.setVisible(false);
        confirmButton.setVisible(false);
    }

    //行修改按钮功能
    private void setPerModifyButtonEvent(ObservableList<Person> list, Button
modify, int line) {
        modify.setOnAction(event -> {
            modifyChoice = line;
            confirmButton.setVisible(true);
            modifyEvent(list, line);
        });
    }

    private void setSearchButtonVisual(boolean value) {
        search1.setVisible(value);
        search2.setVisible(value);
        search3.setVisible(value);
        search4.setVisible(value);

    }

    private void setDeleteButtonVisual(boolean value) {
        delete1.setVisible(value);
        delete2.setVisible(value);
        delete3.setVisible(value);
        delete4.setVisible(value);
        delete5.setVisible(value);
        delete6.setVisible(value);
        delete7.setVisible(value);
        delete8.setVisible(value);
        delete9.setVisible(value);
        delete10.setVisible(value);
        delete11.setVisible(value);
        delete12.setVisible(value);
        delete13.setVisible(value);
        delete14.setVisible(value);
        delete15.setVisible(value);
    }

    private void setDelete_bu(ObservableList<Person> list, TableView<Person>
tableView, AnchorPane ap, double x, double y, int index, Button delete) {
        //设置行删除按钮的位置
        delete.setLayoutX(x);
        delete.setLayoutY(y);
        //设置行删除按钮的事件
        delete.setOnAction(event1 -> {
            list.remove(index);
            tableView.refresh();
            System.out.println("成功删除");
```

```
        });
        ap.getChildren().add(delete);
    }

    private void setModifyButtonVisual(boolean value) {
        modify1.setVisible(value);
        modify2.setVisible(value);
        modify3.setVisible(value);
        modify4.setVisible(value);
        modify5.setVisible(value);
        modify6.setVisible(value);
        modify7.setVisible(value);
        modify8.setVisible(value);
        modify9.setVisible(value);
        modify10.setVisible(value);
        modify11.setVisible(value);
        modify12.setVisible(value);
        modify13.setVisible(value);
        modify14.setVisible(value);
        modify15.setVisible(value);
    }


    //修改具体实现逻辑
    private void modifyEvent(ObservableList<Person> list, int line) {
        Person temp = list.get(line);
        //姓名
        nameField.setVisible(true);
        nameField.setText(temp.getName());
        temp.setName(nameField.getText());
        //地址
        addressField.setVisible(true);
        addressField.setText(temp.getAddress());
        temp.setAddress(addressField.getText());
        //电话
        phoneNumberField.setVisible(true);
        phoneNumberField.setText(temp.getPhoneNumber());
        temp.setPhoneNumber(phoneNumberField.getText());
        //邮箱
        emailAddressField.setVisible(true);
        emailAddressField.setText(temp.getEmailAddress());
        temp.setEmailAddress(emailAddressField.getText());
        if (temp instanceof Postgraduate) {
            //导师
            tutorField.setVisible(true);
            tutorField.setText(((Postgraduate) temp).getTutor());
            ((Postgraduate) temp).setTutor(tutorField.getText());
            //研究方向
            searchField.setVisible(true);
            searchField.setText(((Postgraduate) temp).getName());
            ((Postgraduate) temp).setSearchField(searchField.getText());
        }
        if (temp instanceof Employee) {
            //办公室
            officeField.setVisible(true);
            officeField.setText(((Employee) temp).getOffice());
```

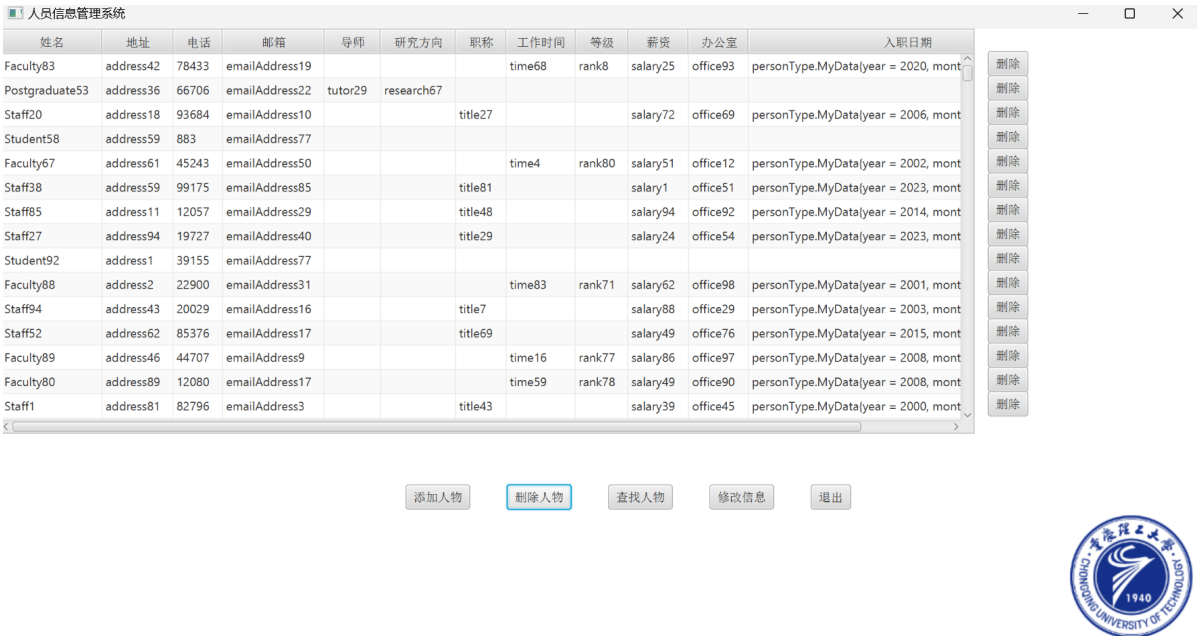
```
            ((Employee) temp).setOffice(officeField.getText());
            //工资
            salaryField.setVisible(true);
            salaryField.setText(((Employee) temp).getSalary());
            ((Employee) temp).setSalary(salaryField.getText());
        }
        if (temp instanceof Staff) {
            //职称
            titleField.setVisible(true);
            titleField.setText(((Staff) temp).getTitle());
            ((Staff) temp).setTitle(titleField.getText());
        }
        if (temp instanceof Faculty) {
            //上班时间
            workTimeField.setVisible(true);
            workTimeField.setText(((Faculty) temp).getWorkTime());
            ((Faculty) temp).setWorkTime(workTimeField.getText());
            //等级
            rankField.setVisible(true);
            rankField.setText(((Faculty) temp).getRank());
            ((Faculty) temp).setRank(rankField.getText());
        }
        System.out.println("修改成功");
        list.set(line, temp);
    }


}
```

输出效果

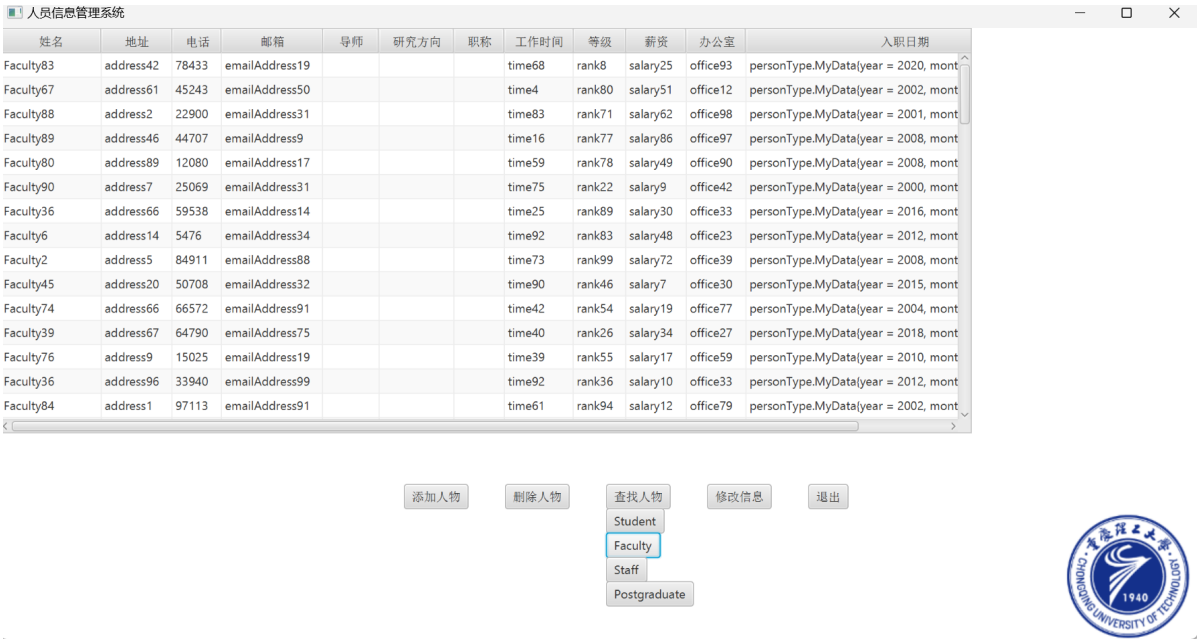
初始界面



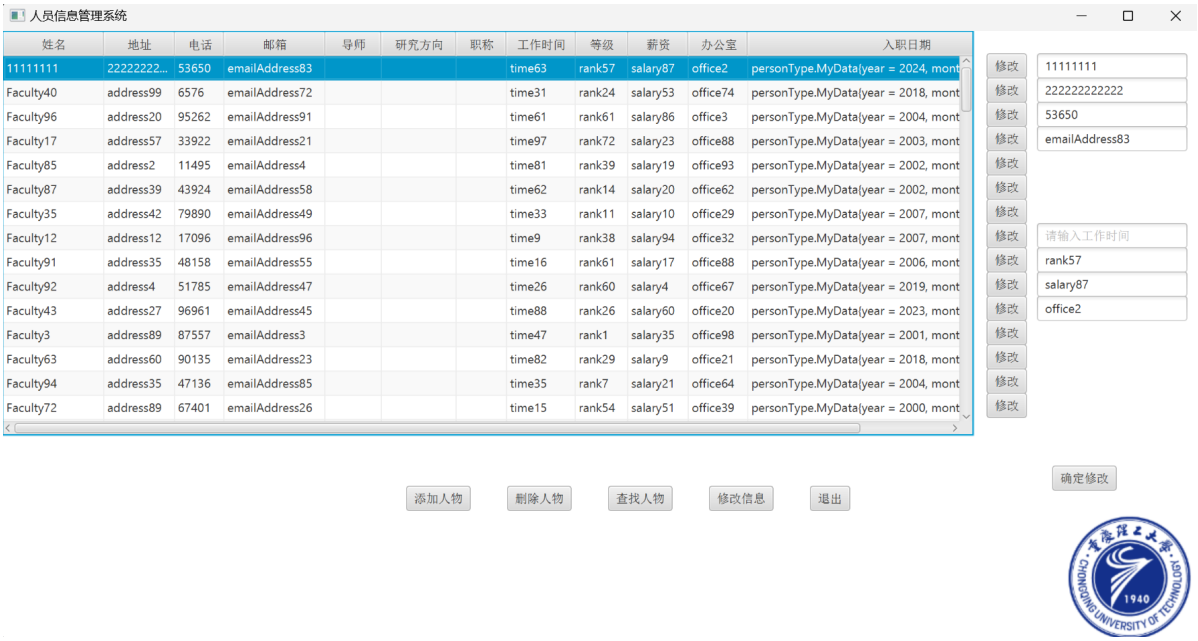
删除界面


人员信息管理系统
姓名 地址 电话 邮箱 导师 研究方向 职称 工作时间 等级 薪资 办公室 入职日期
Faculty83 address42 78433 emailAddress19 time68 rank8 salary25 office93 personType.MyData(year = 2020, mont
Postgraduate53 address36 66706 emailAddress22 tutor29 research67
Staff20 address18 93684 emailAddress10 title27 salary72 office69 personType.MyData(year = 2006, mont
Student58 address59 883 emailAddress77
Faculty67 address61 45243 emailAddress50 time4 rank80 salary51 office12 personType.MyData(year = 2002, mont
Staff38 address59 99175 emailAddress85 title81 salary1 office51 personType.MyData(year = 2023, mont
Staff85 address11 12057 emailAddress29 title48 salary94 office92 personType.MyData(year = 2014, mont
Staff27 address94 19727 emailAddress40 title29 salary24 office54 personType.MyData(year = 2023, mont
Student92 address1 39155 emailAddress77
Faculty88 address2 22900 emailAddress31 time83 rank71 salary62 office98 personType.MyData(year = 2001, mont
Staff94 address43 20029 emailAddress16 title7 salary88 office29 personType.MyData(year = 2003, mont
Staff52 address62 85376 emailAddress17 title69 salary49 office76 personType.MyData(year = 2015, mont
Faculty89 address46 44707 emailAddress9 time16 rank77 salary86 office97 personType.MyData(year = 2008, mont
Faculty80 address89 12080 emailAddress17 time59 rank78 salary49 office90 personType.MyData(year = 2008, mont
Staff1 address81 82796 emailAddress3 title43 salary39 office45 personType.MyData(year = 2000, mont
删除
添加人物
删除人物
查找人物
修改信息
退出

查找界面


人员信息管理系统
姓名 地址 电话 邮箱 导师 研究方向 职称 工作时间 等级 薪资 办公室 入职日期
Faculty83 address42 78433 emailAddress19 time68 rank8 salary25 office93 personType.MyData(year = 2020, mont
Faculty67 address61 45243 emailAddress50 time4 rank80 salary51 office12 personType.MyData(year = 2002, mont
Faculty88 address2 22900 emailAddress31 time83 rank71 salary62 office98 personType.MyData(year = 2001, mont
Faculty89 address46 44707 emailAddress9 time16 rank77 salary86 office97 personType.MyData(year = 2008, mont
Faculty80 address89 12080 emailAddress17 time59 rank78 salary49 office90 personType.MyData(year = 2008, mont
Faculty90 address7 25069 emailAddress31 time75 rank22 salary9 office42 personType.MyData(year = 2000, mont
Faculty36 address66 59538 emailAddress14 time25 rank89 salary30 office33 personType.MyData(year = 2016, mont
Faculty6 address14 5476 emailAddress34 time92 rank83 salary48 office23 personType.MyData(year = 2012, mont
Faculty2 address5 84911 emailAddress88 time73 rank99 salary72 office39 personType.MyData(year = 2008, mont
Faculty45 address20 50708 emailAddress32 time90 rank46 salary7 office30 personType.MyData(year = 2015, mont
Faculty74 address66 66572 emailAddress91 time42 rank54 salary19 office77 personType.MyData(year = 2004, mont
Faculty39 address67 64790 emailAddress75 time40 rank26 salary34 office27 personType.MyData(year = 2018, mont
Faculty76 address9 15025 emailAddress19 time39 rank55 salary17 office59 personType.MyData(year = 2010, mont
Faculty36 address96 33940 emailAddress99 time92 rank36 salary10 office33 personType.MyData(year = 2012, mont
Faculty84 address1 97113 emailAddress91 time61 rank94 salary12 office79 personType.MyData(year = 2002, mont
添加人物
删除人物
查找人物
修改信息
退出
Student
Faculty
Staff
Postgraduate

修改界面


人员信息管理系统
姓名 地址 电话 邮箱 导师 研究方向 职称 工作时间 等级 薪资 办公室 入职日期
11111111 22222222... 53650 emailAddress83 time63 rank57 salary87 office2 personType.MyData(year = 2024, mont
Faculty40 address99 6576 emailAddress72 time31 rank24 salary53 office74 personType.MyData(year = 2018, mont
Faculty96 address20 95262 emailAddress91 time61 rank61 salary86 office3 personType.MyData(year = 2004, mont
Faculty17 address57 33922 emailAddress21 time97 rank72 salary23 office88 personType.MyData(year = 2003, mont
Faculty85 address2 11495 emailAddress4 time81 rank39 salary19 office93 personType.MyData(year = 2002, mont
Faculty87 address39 43924 emailAddress58 time62 rank14 salary20 office62 personType.MyData(year = 2002, mont
Faculty35 address42 79890 emailAddress49 time33 rank11 salary10 office29 personType.MyData(year = 2007, mont
Faculty12 address12 17096 emailAddress96 time9 rank38 salary94 office32 personType.MyData(year = 2007, mont
Faculty91 address35 48158 emailAddress55 time16 rank61 salary17 office88 personType.MyData(year = 2006, mont
Faculty92 address4 51785 emailAddress47 time26 rank60 salary4 office67 personType.MyData(year = 2019, mont
Faculty43 address27 96961 emailAddress45 time88 rank26 salary60 office20 personType.MyData(year = 2023, mont
Faculty3 address89 87557 emailAddress3 time47 rank1 salary35 office98 personType.MyData(year = 2001, mont
Faculty63 address60 90135 emailAddress23 time82 rank29 salary9 office21 personType.MyData(year = 2018, mont
Faculty94 address35 47136 emailAddress85 time35 rank7 salary21 office64 personType.MyData(year = 2004, mont
Faculty72 address89 67401 emailAddress26 time15 rank54 salary51 office39 personType.MyData(year = 2000, mont
修改
11111111
222222222222
53650
emailAddress83
请输入工作时间
rank57
salary87
office2
确定修改
添加人物
删除人物
查找人物
修改信息
退出

6.模拟风扇

源码:

https://github.com/gyw666/javaDesign6

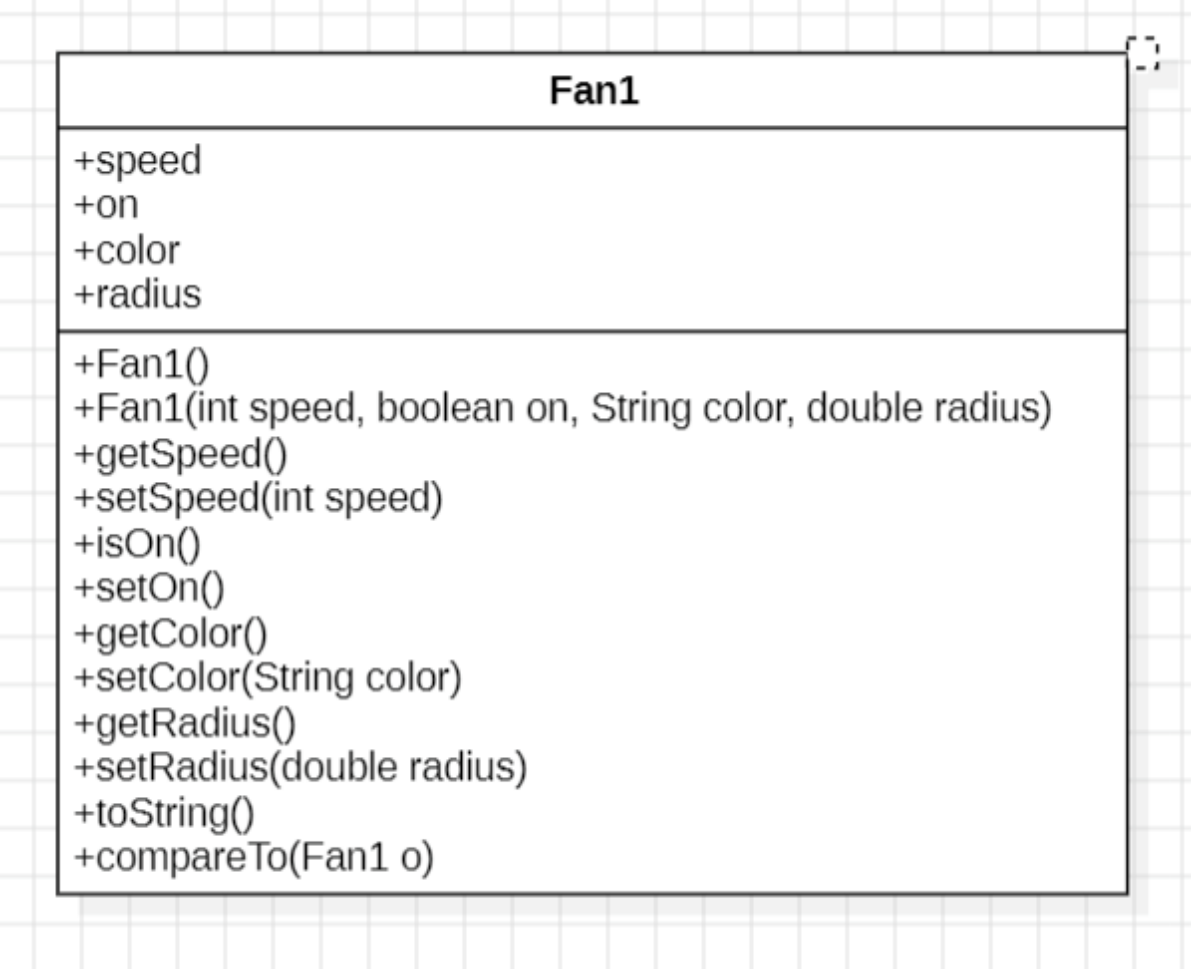
version 1

题目要求:

模拟实现电风扇，可以调 3 档速度（慢速、中速、快速）；开关按钮；定时吹风；描述风扇的扇叶大小、颜色等。

设计 Fan 类，属性包括：3 个常量 SLOW（1）、MEDIUM（2）、FAST（3）代表风扇的速度；1 个 int 属性 speed 指定速度，默认值为 SLOW；1 个 boolean 属性 on 指定开关机，默认值 false；1 个 double 属性 radius 指定风扇扇叶大小；1 个 String 属性 color 指定扇叶颜色，默认值为 blue。方法包括这些属性的访问器、构造函数、重写 Object 类的 toString()和 equals()方法等。

UML



代码

```
package fanType;

public class Fan1 implements Comparable<Fan1>{
    public static final int SLOW = 1;
    public static final int MEDIUM = 2;
    public static final int FAST = 3;

    private int speed;
    private boolean on;
    private String color;
    private double radius;


    public Fan1() {
        speed = SLOW;
        on = false;
        color = "blue";
    }


    public Fan1(int speed, boolean on, String color, double radius) {
        this.speed = speed;
        this.on = on;
        this.color = color;
        this.radius = radius;
    }

    /**
     * 获取
     * @return speed
     */
    public int getSpeed() {
        return speed;
    }

    /**
     * 设置
     * @param speed
     */
    public void setSpeed(int speed) {
        this.speed = speed;
    }

    /**
     * 获取
     * @return on
     */
    public boolean isOn() {
        return on;
    }

    /**
     * 设置
```

```
     * @param on
     */
    public void setOn(boolean on) {
        this.on = on;
    }

    /**
     * 获取
     * @return color
     */
    public String getColor() {
        return color;
    }

    /**
     * 设置
     * @param color
     */
    public void setColor(String color) {
        this.color = color;
    }

    /**
     * 获取
     * @return radius
     */
    public double getRadius() {
        return radius;
    }

    /**
     * 设置
     * @param radius
     */
    public void setRadius(double radius) {
        this.radius = radius;
    }

    public String toString() {
        return "fan1{ speed = " + speed + ", on = " + on + ", color = " + color +
", radius = " + radius + "}";
    }

    @Override
    public int compareTo(Fan1 o) {
        return Double.compare(this.radius, o.radius);
    }
}
```

测试类

```
package Test;
import fanType.Fan1;

public class test1 {
    public static void main(String[] args) {
        Fan1 fan1 = new Fan1();
        fan1.setSpeed(Fan1.FAST);
        fan1.setRadius(10);
        fan1.setColor("yellow");
        fan1.setOn(true);
        System.out.println(fan1.toString());
    }
}
```

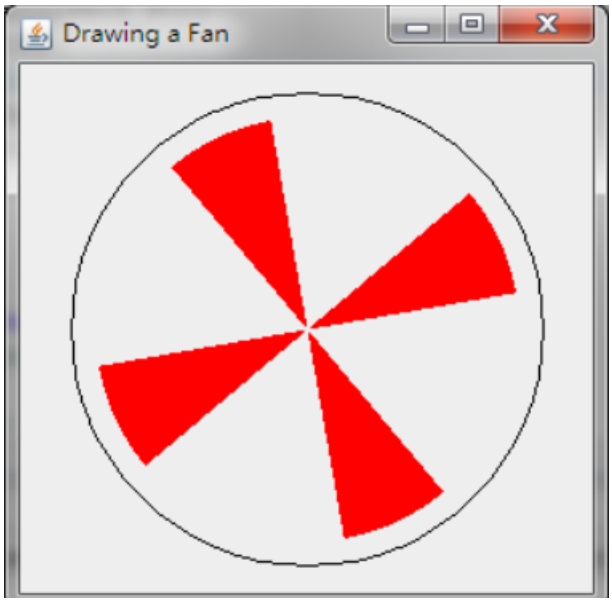
输出效果

```
D:\java\jdk8\bin\java.exe ...
fan1{ speed = 3, on = true, color = yellow, radius = 10.0}
```

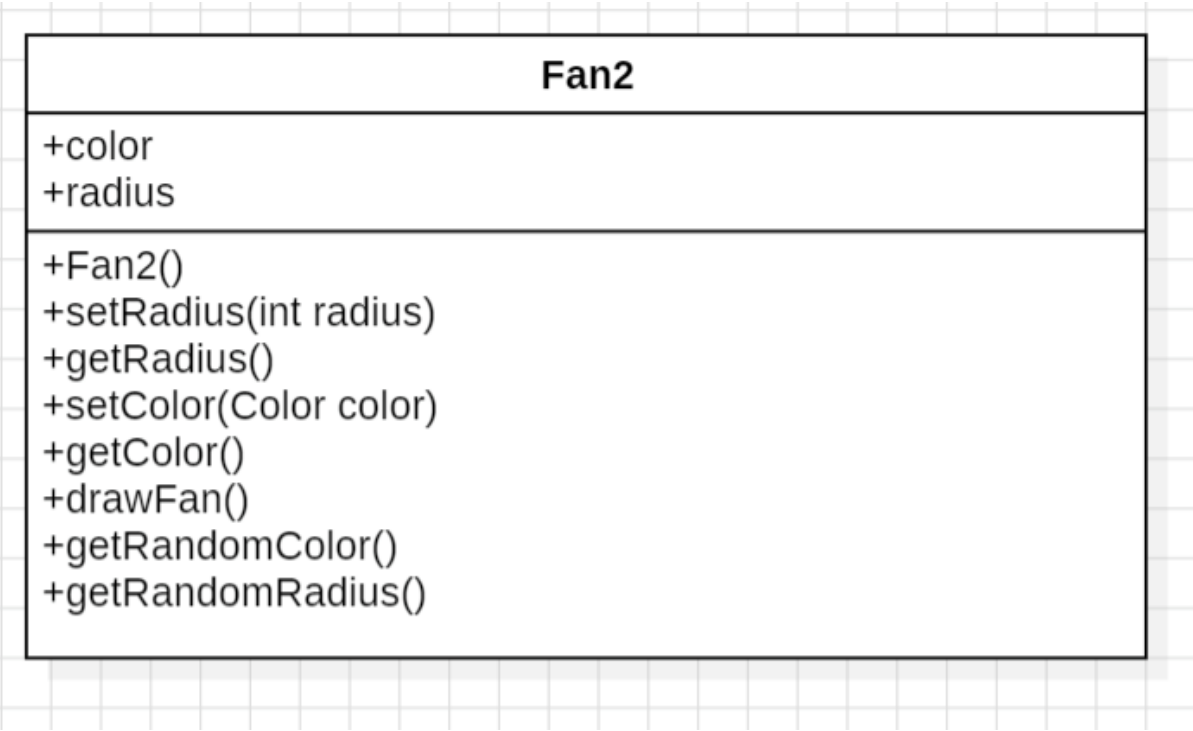
version 2

题目要求:

修改版本 1 中 Fan 类，让其继承 Pane 类，并且把 color 属性设置为 Color 类型，默认属性为 red。随机产生 radius，取值范围为 1-5；随机产生颜色，取值范围为 red、blue、yellow、green、orange；根据 color、radius 属性值绘制风扇。运行如下图：



UML



代码

```
package fanType;

import javafx.scene.layout.Pane;
import javafx.scene.shape.*;
import javafx.scene.paint.Color;

import java.util.Random;


public class Fan2 extends Pane{
    private Color color;
    private int radius;
    public Fan2() {
        color = Color.RED;
        drawFan();
    }

    public void setRadius(int radius) {
        this.radius = radius;
        drawFan();
    }

    public int getRadius() {
        return radius;

    }

    public void setColor(Color color) {
        this.color = color;
        drawFan();
    }
```

```
    public Color getColor() {
        return color;
    }

    public Fan2(Color color, int radius) {
        this.color = color;
        this.radius = radius;
        drawFan();
    }

    private void drawFan() {
        double centerX = 400;
        double centerY = 300;
        for (int i = 0; i < 4; i++) {
            Arc arc = new Arc(centerX,centerY,radius,radius,i*90+30,30);
            arc.setType(ArcType.ROUND);
            arc.setFill(color);
            arc.setStroke(Color.BLACK);
            arc.setStrokeWidth(1);
            this.getChildren().add(arc);
        }
        // 创建Path对象
        Path path = new Path();

        // 起点：在圆弧的顶部（3点钟位置）
        MoveTo moveTo = new MoveTo();
        moveTo.setX(centerX + radius); // 起点X坐标
        moveTo.setY(centerY);          // 起点Y坐标

        // 创建上半圆弧
        ArcTo arcTo1 = new ArcTo();
        arcTo1.setX(centerX - radius); // 上半圆弧终点X坐标
        arcTo1.setY(centerY);          // 上半圆弧终点Y坐标
        arcTo1.setRadiusX(radius);     // X轴方向的半径
        arcTo1.setRadiusY(radius);     // Y轴方向的半径
        arcTo1.setSweepFlag(false);    // 指定弧的方向为逆时针

        // 创建下半圆弧
        ArcTo arcTo2 = new ArcTo();
        arcTo2.setX(centerX + radius); // 下半圆弧终点X坐标
        arcTo2.setY(centerY);          // 下半圆弧终点Y坐标
        arcTo2.setRadiusX(radius);     // X轴方向的半径
        arcTo2.setRadiusY(radius);     // Y轴方向的半径
        arcTo2.setSweepFlag(false);    // 指定弧的方向为逆时针

        // 将起点和圆弧添加到路径
        path.getElements().add(moveTo);
        path.getElements().add(arcTo1);
        path.getElements().add(arcTo2);

        // 设置路径的样式
        path.setStroke(Color.BLACK);
        path.setStrokeWidth(2);
        path.setFill(null);

        // 将路径添加到Pane
```

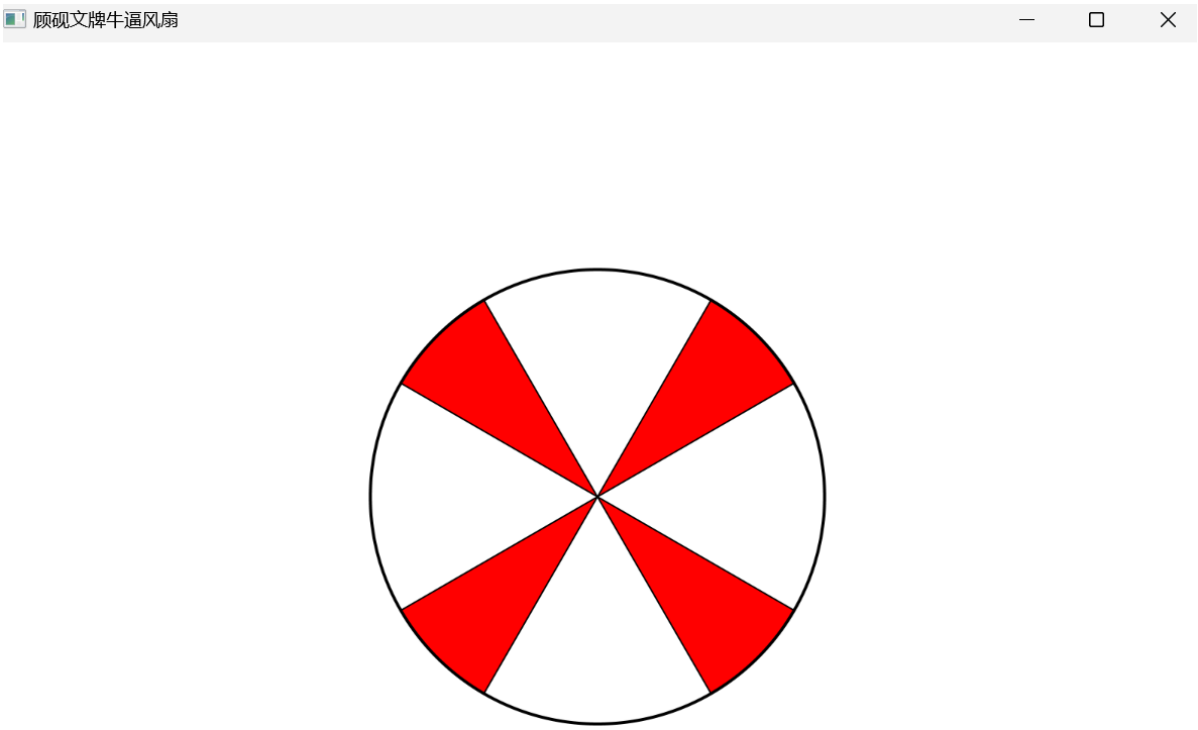
```
        this.getChildren().add(path);

    }


    public static Color getRandomColor() {
        int choice=new Random().nextInt(5);
        switch (choice) {
            case 0:{
                return Color.RED;
            }
            case 1:{
                return Color.BLUE;
            }
            case 2:{
                return Color.YELLOW;
            }
            case 3:{
                return Color.GREEN;
            }
            case 4:{
                return Color.ORANGE;
            }
        }
        return null;
    }

    public static int getRandomRadius(){
        return 100*(1+new Random().nextInt(5));
    }
}
```

测试类

```
package Test;

import fanType.Fan2;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;


public class test2 extends Application {

    public static void main(String[] args) {
        launch(args);
    }


    @Override
    public void start(Stage primaryStage) throws Exception {
        Color color = Fan2.getRandomColor();
        int radius = Fan2.getRandomRadius();
        Fan2 fan = new Fan2(color, radius);
```

```
        Scene scene = new Scene(fan, 800, 600);
        primaryStage.setScene(scene);
        primaryStage.setAlwaysOnTop(true);
        primaryStage.setTitle("顾砚文牌牛逼风扇");
        primaryStage.show();

    }
}
```

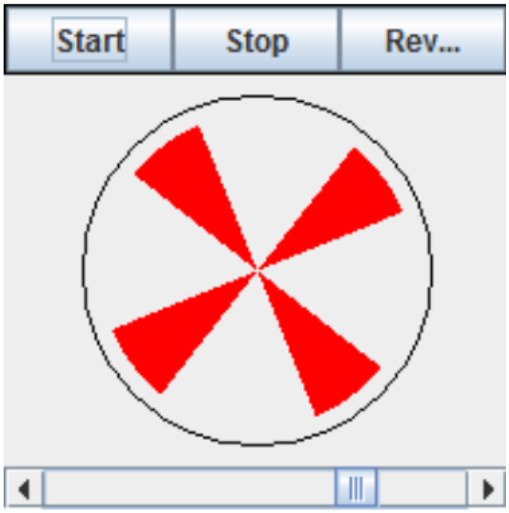
输出效果



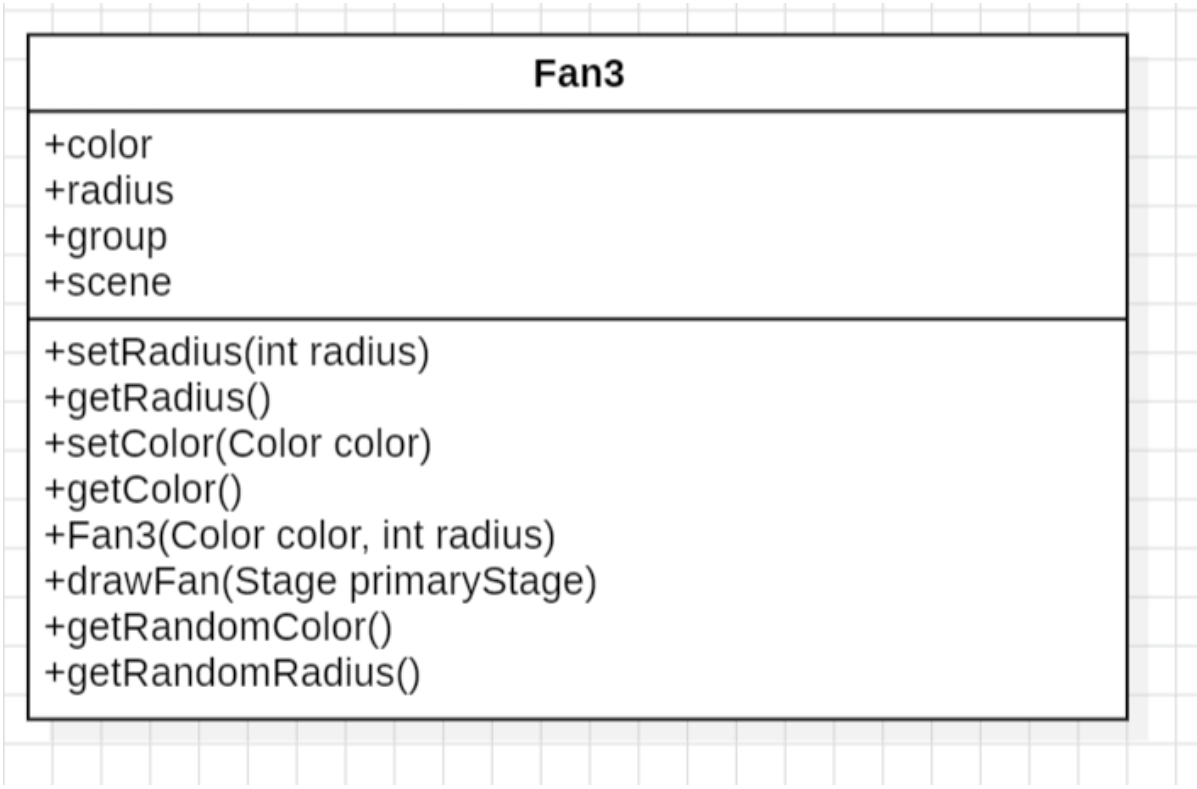
version 3

题目要求:

让版本 2 中的风扇转起来。创建一个 FanControl 类包含以下内容：Start、Stop、Reverse 按钮，用于开启、关闭、反转控制；一个滚动条控制速度。运行示例如下：



UML



代码

```
package fanType;

import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.geometry.Pos;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Slider;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.scene.shape.*;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
```

```java
import javafx.util.Duration;

import java.util.Random;


public class Fan3 extends Pane {
    private Color color;
    private int radius;
    Group group = new Group();
    Scene scene;

    public Fan3() {
        color = Color.RED;
    }

    public void setRadius(int radius) {
        this.radius = radius;
    }

    public int getRadius() {
        return radius;

    }

    public void setColor(Color color) {
        this.color = color;
    }

    public Color getColor() {
        return color;
    }

    public Fan3(Color color, int radius) {
        this.color = color;
        this.radius = radius;
    }

    public void drawFan(Stage primaryStage) {
        double centerX = 400;
        double centerY = 300;
        for (int i = 0; i < 4; i++) {
            Arc arc = new Arc(centerX, centerY, radius, radius, i * 90 + 30, 30);
            arc.setType(ArcType.ROUND);
            arc.setFill(color);
            arc.setStroke(Color.BLACK);
            arc.setStrokeWidth(1);
            group.getChildren().add(arc);
        }
        // 创建Path对象
        Path path = new Path();

        // 起点：在圆弧的顶部（3点钟位置）
        MoveTo moveTo = new MoveTo();
        moveTo.setX(centerX + radius); // 起点X坐标
        moveTo.setY(centerY);          // 起点Y坐标
```

```
        // 创建上半圆弧
        ArcTo arcTo1 = new ArcTo();
        arcTo1.setX(centerX - radius); // 上半圆弧终点X坐标
        arcTo1.setY(centerY);          // 上半圆弧终点Y坐标
        arcTo1.setRadiusX(radius);     // X轴方向的半径
        arcTo1.setRadiusY(radius);     // Y轴方向的半径
        arcTo1.setSweepFlag(false);    // 指定弧的方向为逆时针

        // 创建下半圆弧
        ArcTo arcTo2 = new ArcTo();
        arcTo2.setX(centerX + radius); // 下半圆弧终点X坐标
        arcTo2.setY(centerY);          // 下半圆弧终点Y坐标
        arcTo2.setRadiusX(radius);     // X轴方向的半径
        arcTo2.setRadiusY(radius);     // Y轴方向的半径
        arcTo2.setSweepFlag(false);    // 指定弧的方向为逆时针

        // 将起点和圆弧添加到路径
        path.getElements().add(moveTo);
        path.getElements().add(arcTo1);
        path.getElements().add(arcTo2);

        // 设置路径的样式
        path.setStroke(Color.BLACK);
        path.setStrokeWidth(2);
        path.setFill(null);

        // 将路径添加到Pane
        group.getChildren().add(path);


        Button blueButton = new Button("蓝色");
        Button yellowButton = new Button("黄色");
        Button greenButton = new Button("绿色");
        Button orangeButton = new Button("橙色");
        Button redButton = new Button("红色");

        VBox hc = new VBox(10, blueButton, yellowButton, greenButton,
orangeButton, redButton);
        hc.setAlignment(Pos.BOTTOM_RIGHT);


        Button radius_50 = new Button("50");
        Button radius_100 = new Button("100");
        Button radius_150 = new Button("150");
        Button radius_200 = new Button("200");
        Button radius_250 = new Button("250");

        VBox r = new VBox(10, radius_50, radius_100, radius_150, radius_200,
radius_250);
        r.setAlignment(Pos.BOTTOM_LEFT);

        radius_100.setOnAction(event -> {
            new Fan3(this.color, 100).drawFan(primaryStage);
        });

        radius_150.setOnAction(event -> {
```

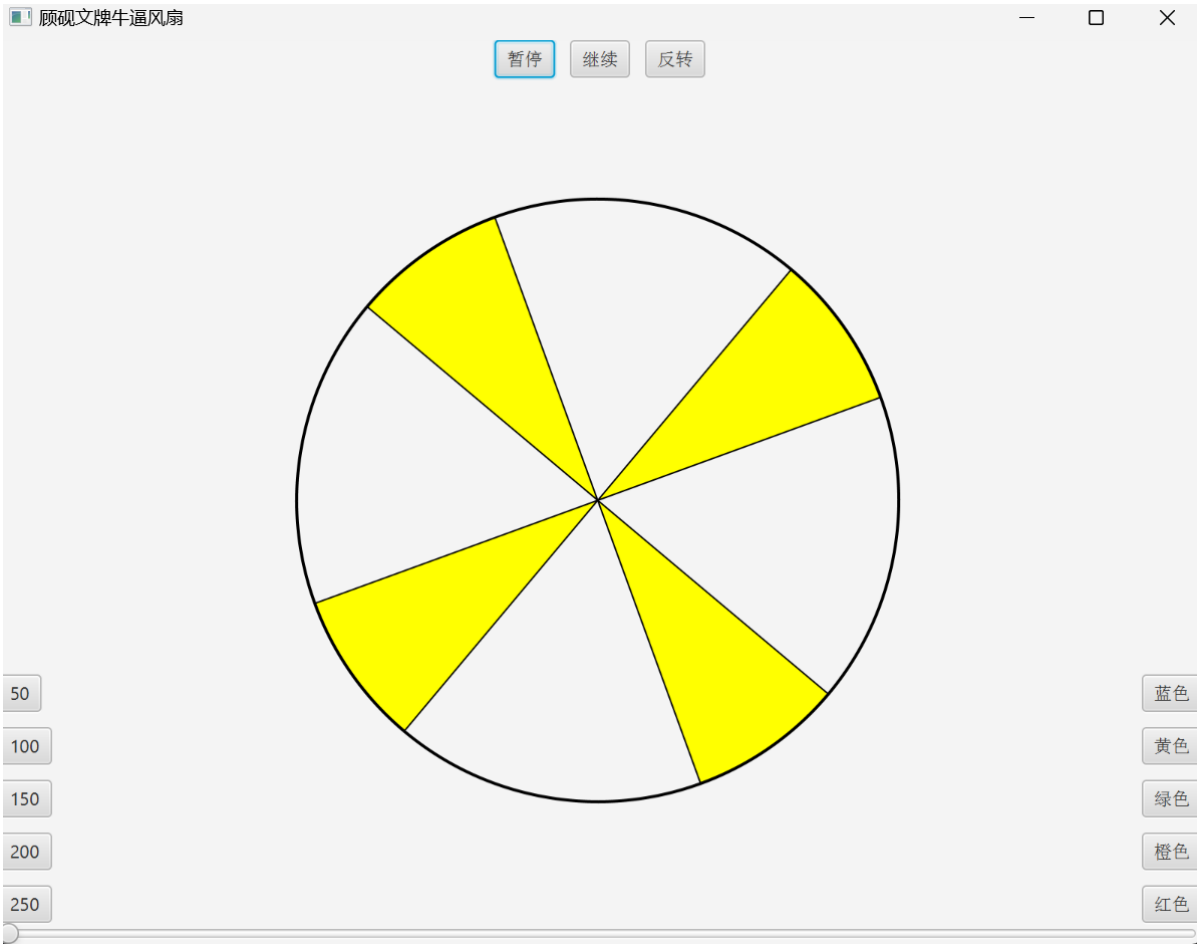
```
            new Fan3(this.color, 150).drawFan(primaryStage);
        });

        radius_200.setOnAction(event -> {
            new Fan3(this.color, 200).drawFan(primaryStage);
        });

        radius_50.setOnAction(event -> {
            new Fan3(this.color, 50).drawFan(primaryStage);
        });

        radius_250.setOnAction(event -> {
            new Fan3(this.color, 250).drawFan(primaryStage);
        });


        Button pause = new Button("暂停"); //暂停按钮
        Button resume = new Button("继续");   //继续按钮
        Button reverse = new Button("反转"); //反转按钮


        HBox hBox = new HBox(10, pause, resume, reverse);
        hBox.setAlignment(Pos.BOTTOM_CENTER);


        redButton.setOnAction(event -> {
            new Fan3(Color.RED, radius).drawFan(primaryStage);
        });

        blueButton.setOnAction(event -> {
            new Fan3(Color.BLUE, radius).drawFan(primaryStage);
        });

        yellowButton.setOnAction(event -> {
            new Fan3(Color.YELLOW, radius).drawFan(primaryStage);
        });

        greenButton.setOnAction(event -> {
            new Fan3(Color.GREEN, radius).drawFan(primaryStage);
        });

        orangeButton.setOnAction(event -> {
            new Fan3(Color.ORANGE, radius).drawFan(primaryStage);
        });

        //关键帧
        KeyFrame keyFrame1 = new KeyFrame(Duration.millis(10), event ->
group.setRotate(group.getRotate() + 1));
        KeyFrame keyFrame2 = new KeyFrame(Duration.millis(10), event ->
group.setRotate(group.getRotate() - 1));
        Timeline animation = new Timeline(keyFrame1);   //时间线动画
        animation.setCycleCount(Timeline.INDEFINITE);   //无限循环次数
        animation.play();   //启动动画

        pause.setOnAction(event -> animation.pause());
        resume.setOnAction(event -> animation.play());
```

```
        reverse.setOnAction(event -> {
            animation.stop();

 animation.getKeyFrames().add(animation.getKeyFrames().remove(0).equals(keyFrame1
) ? keyFrame2 : keyFrame1);
            animation.play();
        });
        Slider slider = new Slider();   //滑动条
        slider.setMax(10);  //滑动条设置最大值
        slider.valueProperty().addListener(observable ->
animation.setRate(slider.getValue())); //滑动条添加监听器

        BorderPane borderPane = new BorderPane(new BorderPane(group));
        borderPane.setTop(hBox);
        borderPane.setRight(hc);
        borderPane.setLeft(r);
        borderPane.setBottom(slider);

        scene = new Scene(borderPane, 800, 600);
        primaryStage.setScene(scene);
        primaryStage.setTitle("顾砚文牌牛逼风扇");
        primaryStage.show();
    }


    public static Color getRandomColor() {
        int choice = new Random().nextInt(5);
        switch (choice) {
            case 0: {
                return Color.RED;
            }
            case 1: {
                return Color.BLUE;
            }
            case 2: {
                return Color.YELLOW;
            }
            case 3: {
                return Color.GREEN;
            }
            case 4: {
                return Color.ORANGE;
            }
        }
        return null;
    }

    public static int getRandomRadius() {
        return 50 * (1 + new Random().nextInt(4));
    }
}
```

测试类

```
package Test;

import fanType.Fan3;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;


public class test3 extends Application {

    public static void main(String[] args) {
        launch(args);
    }


    @Override
    public void start(Stage primaryStage) throws Exception {
        Color color = Fan3.getRandomColor();
        int radius = Fan3.getRandomRadius();
        Fan3 fan = new Fan3(color, radius);
        fan.setRadius(200);
        fan.drawFan(primaryStage);

    }
}
```

输出效果



课程设计总结

在这次的程序设计中，我完成了人员管理系统和模拟风扇的设计开发，收获颇丰。这两个项目使我在实际应用中深入理解了JavaFX的核心组件和动画效果，并提升了编程能力和项目管理技能。

人员管理:

首先，人员管理系统项目让我深入了解了JavaFX中的TableView、Button和TextField等基本组件的使用。TableView作为展示和管理表格数据的重要组件，在这个项目中发挥了核心作用。我学习并实践了如何通过ObservableList来动态更新表格数据，从而实现数据的增删改查功能。通过这些操作，我对数据绑定（data
binding）有了更深的认识，并学会了如何有效地管理和更新界面上的数据。

在设计用户界面时，Button和TextField的结合使用让我掌握了响应用户输入的基本方法。例如，通过监听Button的点击事件，我能够触发相应的动作，如添加新人员、删除选定人员等。而TextField则用于接收用户的输入数据，通过设置合适的事件处理器，可以实现数据的验证和处理。这一过程不仅提高了我的事件处理能力，还让我理解了如何设计用户友好的交互界面。

模拟风扇:

在模拟风扇项目中，我额外使用了Slider和KeyFrame等组件。Slider用于调节风扇速度，这使我对JavaFX的控件交互有了更深的理解。KeyFrame的使用则让我掌握了JavaFX的动画系统，通过设置关键帧和时间轴，我实现了风扇叶片的旋转效果。在这个过程中，我学会了如何创建平滑的动画效果，并理解了动画的性能优化问题。

心得:

这两个项目还让我意识到了代码结构和模块化设计的重要性。在开发过程中，我逐渐养成了将代码分离成多个类和方法的习惯，使代码更加清晰易懂，也便于维护和扩展。此外，我还学会了使用Git进行版本控制。这对团队协作和项目管理非常有帮助。通过Git，我能够记录每次代码的修改历史，方便回溯和查看不同版本之间的变化。这一技能在实际开发中非常重要，尤其是在多人协作的项目中，可以有效避免代码冲突和版本混乱问题。

总的来说，通过这两个项目的开发，我不仅提高了JavaFX编程水平，还在实际应用中锻炼了逻辑思维和问题解决能力。理论知识在项目中的实践应用，使我更加理解了软件开发的全流程。从需求分析、设计实现到测试调优，每一个环节都充满了挑战与收获。这段经历为我未来的学习和工作打下了坚实的基础，也让我对软件开发充满了信心和热情。