# 主干代码

**目录**

[**主干代码 1**](#_Toc73127789)

[**Disk.java 2**](#_Toc73127790)

[**Postion.java 3**](#_Toc73127791)

[**FCFS.java 4**](#_Toc73127792)

[**SSTF.java 6**](#_Toc73127793)

[**SCAN.java 8**](#_Toc73127794)

[**C\_SCAN.java 11**](#_Toc73127795)

[**Out.java 14**](#_Toc73127796)

[**Main.java 18**](#_Toc73127797)

## Disk.java

package model;

import java.util.ArrayList;

public abstract class Disk {

    private int length;

    private Postion currentPostion;

    public ArrayList<Postion> drawFlow = *new* ArrayList<>();

    public Disk(Postion initPostion){

*// 设置初始点*

*this*.currentPostion = initPostion;

*this*.currentPostion.toVisited();

*this*.drawFlow.add(*this*.currentPostion);

*// 设置寻道初始长度*

*this*.length = 0;

    }

    public void addLength(int length){

*this*.length += length;

    }

*// ----------  getter and setter  ------------ //*

    public Postion getCurrentPostion() {

*return* currentPostion;

    }

    public void setCurrentPostion(Postion currentPostion) {

*this*.currentPostion = currentPostion;

    }

    public int getLength(){

*return* *this*.length;

    }

    public abstract void run();

    public abstract Postion getNext();

}

## Postion.java

package model;

public class Postion implements Cloneable ,Comparable<Postion>{

    private int location;

    private boolean visited = false;

    public Postion(int location){

*this*.location = location;

    }

    public int getLocation(){

*return* *this*.location;

    }

    public void setLocation(int location){

*this*.location = location;

    }

    public boolean isVisited(){

*return* visited;

    }

    public void toVisited(){

*this*.visited = true;

    }

    @Override

    public String toString() {

*//        return "Postion{" +*

*//                "location=" + location +*

*//                ", visited=" + visited +*

*//                '}';*

*return* String.valueOf(location);

    }

    @Override

    public int compareTo(Postion o) {

*return* Integer.compare(*this*.getLocation(), o.getLocation());

    }

}

## FCFS.java

package action;

import model.Disk;

import model.Postion;

import java.util.ArrayList;

public class FCFS extends Disk{

    private ArrayList<Postion> postionArrayList = *new* ArrayList<>();

    public FCFS(Postion initPostion, ArrayList<Postion> postionArrayList) {

*super*(initPostion);

*for*(Postion i *:* postionArrayList){

*this*.postionArrayList.add(*new* Postion(i.getLocation()));

        }

    }

    @Override

    public Postion getNext() {

        int i = 0;

*for*(;i<postionArrayList.size() && postionArrayList.get(i).isVisited();i++){

        }

*// postionArrayList.get(i).isVisited() == false || i>=postionArrayList.size()*

*if*(i>=postionArrayList.size()){

*return* null;

        }

*this*.addLength(Math.abs(*this*.getCurrentPostion().getLocation() - postionArrayList.get(i).getLocation()));

        postionArrayList.get(i).toVisited();

*this*.setCurrentPostion(postionArrayList.get(i));

*this*.drawFlow.add(*this*.getCurrentPostion());

*return* postionArrayList.get(i);

    }

    @Override

    public void run(){

*while*(true){

*//System.out.println(drawFlow);*

*if*(*this*.getNext()==null) {

*break*;

            }

        }

    }

}

## SSTF.java

package action;

import model.Disk;

import model.Postion;

import java.util.ArrayList;

public class SSTF extends Disk {

    private ArrayList<Postion> postionArrayList = *new* ArrayList<>();

    public SSTF(Postion initPostion, ArrayList<Postion> postionArrayList) {

*super*(initPostion);

*for*(Postion i *:* postionArrayList){

*this*.postionArrayList.add(*new* Postion(i.getLocation()));

        }

    }

    @Override

    public void run() {

*while*(true){

*//System.out.println(drawFlow);*

*if*(*this*.getNext()==null) {

*break*;

            }

        }

    }

    @Override

    public Postion getNext() {

        int min\_location = Integer.MAX\_VALUE;

        int min\_index = postionArrayList.size();

*for*(int i = 0;i< postionArrayList.size();i++){

*//            System.out.println(!postionArrayList.get(i).isVisited());*

*//            System.out.println(Math.abs(getCurrentPostion().getLocation() - postionArrayList.get(i).getLocation()) < min\_location);*

*if*((!postionArrayList.get(i).isVisited()) && (Math.abs(getCurrentPostion().getLocation() - postionArrayList.get(i).getLocation()) < min\_location)){

                min\_index = i;

                min\_location = Math.abs(getCurrentPostion().getLocation() - postionArrayList.get(i).getLocation());

            }

        }

*// min\_index = postionArrayList.size();*

*if*(min\_index == postionArrayList.size()){

*return* null;

        }

*this*.addLength(min\_location);

        postionArrayList.get(min\_index).toVisited();

*this*.setCurrentPostion(postionArrayList.get(min\_index));

*this*.drawFlow.add(*this*.getCurrentPostion());

*return* postionArrayList.get(min\_index);

    }

}

## SCAN.java

package action;

import model.Disk;

import model.Postion;

import java.util.ArrayList;

import java.util.Collections;

public class SCAN extends Disk {

    private ArrayList<Postion> postionArrayList = *new* ArrayList<>();

*// 方向 1: 增大方向 0: 减小方向*

    private int flag;

    public SCAN(Postion initPostion, ArrayList<Postion> postionArrayList, int flag) {

*super*(initPostion);

*this*.flag = flag;

*for*(Postion i *:* postionArrayList){

*this*.postionArrayList.add(*new* Postion(i.getLocation()));

        }

        Collections.sort(*this*.postionArrayList);

        System.out.println("排序后的序列: " + *this*.postionArrayList); *// 小到大*

    }

    @Override

    public void run() {

*while*(true){

*//System.out.println(drawFlow);*

*if*(*this*.getNext()==null) {

*break*;

            }

        }

*//        // 最小的没有被访问过*

*//        if(!this.postionArrayList.get(0).isVisited()){*

*//            this.addLength(Math.abs(postionArrayList.get(0).getLocation() - getCurrentPostion().getLocation()));*

*//            postionArrayList.get(0).toVisited();*

*//            this.setCurrentPostion(postionArrayList.get(0));*

*//            this.drawFlow.add(this.getCurrentPostion());*

*//            while()*

*//        }*

*this*.flag = (*this*.flag == 1*?*0*:*1);

*while*(true){

*//System.out.println(drawFlow);*

*if*(*this*.getNext()==null) {

*break*;

            }

        }

    }

    @Override

    public Postion getNext() {

*if*(flag == 1) {

*// 增大方向*

            int i = 0;

*for*(;i< postionArrayList.size();i++){

*if*(!postionArrayList.get(i).isVisited() && postionArrayList.get(i).getLocation() >= getCurrentPostion().getLocation()){

*break*;

                }

            }

*// i = postionArrayList.size();*

*if*(i == postionArrayList.size()){

*return* null;

            }

*this*.addLength(Math.abs(postionArrayList.get(i).getLocation() - getCurrentPostion().getLocation()));

            postionArrayList.get(i).toVisited();

*this*.setCurrentPostion(postionArrayList.get(i));

*this*.drawFlow.add(*this*.getCurrentPostion());

*return* postionArrayList.get(i);

        }*else* *if*(flag == 0){

*// 减小方向*

            int i = postionArrayList.size()-1;

*for*(;i >= 0; i--){

*if*(!postionArrayList.get(i).isVisited() && postionArrayList.get(i).getLocation() <= getCurrentPostion().getLocation()){

*break*;

                }

            }

*// i = -1;*

*if*(i == -1){

*return* null;

            }

*this*.addLength(Math.abs(postionArrayList.get(i).getLocation() - getCurrentPostion().getLocation()));

            postionArrayList.get(i).toVisited();

*this*.setCurrentPostion(postionArrayList.get(i));

*this*.drawFlow.add(*this*.getCurrentPostion());

*return* postionArrayList.get(i);

        }*else*{

*return* null;

        }

    }

}

## C\_SCAN.java

package action;

import model.Disk;

import model.Postion;

import java.util.ArrayList;

import java.util.Collections;

public class C\_SCAN extends Disk {

    private ArrayList<Postion> postionArrayList = *new* ArrayList<>();

*// 方向 1: 增大方向 0: 减小方向*

    private int flag;

    public C\_SCAN(Postion initPostion, ArrayList<Postion> postionArrayList, int flag) {

*super*(initPostion);

*this*.flag = flag;

*for*(Postion i *:* postionArrayList){

*this*.postionArrayList.add(*new* Postion(i.getLocation()));

        }

        Collections.sort(*this*.postionArrayList);

        System.out.println("排序后的序列: " + *this*.postionArrayList); *// 小到大*

    }

    @Override

    public void run() {

*if*(flag == 1){

*// 增大方向*

*while*(true){

*//System.out.println(drawFlow);*

*if*(*this*.getNext()==null) {

*break*;

                }

            }

*// 最小的没有被访问过*

*if*(!*this*.postionArrayList.get(0).isVisited()){

*this*.addLength(Math.abs(postionArrayList.get(0).getLocation() - getCurrentPostion().getLocation()));

                postionArrayList.get(0).toVisited();

*this*.setCurrentPostion(postionArrayList.get(0));

*this*.drawFlow.add(*this*.getCurrentPostion());

*while*(true){

*//System.out.println(drawFlow);*

*if*(*this*.getNext()==null) {

*break*;

                    }

                }

            }

        }*else* *if*(flag == 0){

*// 减小方向*

*while*(true){

*//System.out.println(drawFlow);*

*if*(*this*.getNext()==null) {

*break*;

                }

            }

*// 最大的没有被访问过*

*if*(!*this*.postionArrayList.get(*this*.postionArrayList.size()-1).isVisited()){

*this*.addLength(Math.abs(postionArrayList.get(*this*.postionArrayList.size()-1).getLocation() - getCurrentPostion().getLocation()));

                postionArrayList.get(*this*.postionArrayList.size()-1).toVisited();

*this*.setCurrentPostion(postionArrayList.get(*this*.postionArrayList.size()-1));

*this*.drawFlow.add(*this*.getCurrentPostion());

*while*(true){

*//System.out.println(drawFlow);*

*if*(*this*.getNext()==null) {

*break*;

                    }

                }

            }

        }

    }

    @Override

    public Postion getNext() {

*if*(flag == 1) {

*// 增大方向*

            int i = 0;

*for*(;i< postionArrayList.size();i++){

*if*(!postionArrayList.get(i).isVisited() && postionArrayList.get(i).getLocation() >= getCurrentPostion().getLocation()){

*break*;

                }

            }

*// i = postionArrayList.size();*

*if*(i == postionArrayList.size()){

*return* null;

            }

*this*.addLength(Math.abs(postionArrayList.get(i).getLocation() - getCurrentPostion().getLocation()));

            postionArrayList.get(i).toVisited();

*this*.setCurrentPostion(postionArrayList.get(i));

*this*.drawFlow.add(*this*.getCurrentPostion());

*return* postionArrayList.get(i);

        }*else* *if*(flag == 0){

*// 减小方向*

            int i = postionArrayList.size()-1;

*for*(;i >= 0; i--){

*if*(!postionArrayList.get(i).isVisited() && postionArrayList.get(i).getLocation() <= getCurrentPostion().getLocation()){

*break*;

                }

            }

*// i = -1;*

*if*(i == -1){

*return* null;

            }

*this*.addLength(Math.abs(postionArrayList.get(i).getLocation() - getCurrentPostion().getLocation()));

            postionArrayList.get(i).toVisited();

*this*.setCurrentPostion(postionArrayList.get(i));

*this*.drawFlow.add(*this*.getCurrentPostion());

*return* postionArrayList.get(i);

        }*else*{

*return* null;

        }

    }

}

## Out.java

package main;

import action.C\_SCAN;

import action.FCFS;

import action.SCAN;

import action.SSTF;

import model.Postion;

import java.util.ArrayList;

public class Out {

    public static int n = 400;

    public static FCFS outFCFS(Postion initPostion, ArrayList<Postion> postionArrayList) {

*//----------------------------------------------------------------------- FCFS*

        System.out.println("\n-------------  FCFS算法  -------------");

        FCFS fcfs = *new* FCFS(initPostion, postionArrayList);

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* FCFS run start \*\*\*\*\*\*\*\*\*\*\*\*\* //");*

        fcfs.run();

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* FCFS run end \*\*\*\*\*\*\*\*\*\*\*\*\* //\n");*

        System.out.println("绘图序列: " + fcfs.drawFlow);

        System.out.println("寻道长度: " + fcfs.getLength());

        System.out.println("平均寻道长度: " + fcfs.getLength() / n);

*return* fcfs;

    }

    public static SSTF outSSTF(Postion initPostion, ArrayList<Postion> postionArrayList) {

*//--------------------------------------------------------------------- SSTF*

        System.out.println("\n-------------  SSTF算法  -------------");

        SSTF sstf = *new* SSTF(initPostion, postionArrayList);

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* SSTF run start \*\*\*\*\*\*\*\*\*\*\*\*\* //");*

        sstf.run();

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* SSTF run end \*\*\*\*\*\*\*\*\*\*\*\*\* //\n");*

        System.out.println("绘图序列: " + sstf.drawFlow);

        System.out.println("寻道长度: " + sstf.getLength());

        System.out.println("平均寻道长度: " + sstf.getLength() / n);

*return* sstf;

    }

    public static SCAN outSCAN1(Postion initPostion, ArrayList<Postion> postionArrayList) {

*//----------------------------------------------------------------------- SCAN - 增大方向1*

        System.out.println("\n-------------  SCAN算法[增大方向]  -------------");

        SCAN scan1 = *new* SCAN(initPostion, postionArrayList, 1);

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* SCAN-1 run start \*\*\*\*\*\*\*\*\*\*\*\*\* //");*

        scan1.run();

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* SCAN-1 run end \*\*\*\*\*\*\*\*\*\*\*\*\* //\n");*

        System.out.println("绘图序列: " + scan1.drawFlow);

        System.out.println("寻道长度: " + scan1.getLength());

        System.out.println("平均寻道长度: " + scan1.getLength() / n);

*return* scan1;

    }

    public static SCAN outSCAN0(Postion initPostion, ArrayList<Postion> postionArrayList) {

*//------------------------------------------------------------------------ SCAN - 减小方向0*

        System.out.println("\n-------------  SCAN算法[减小方向]  -------------");

        SCAN scan0 = *new* SCAN(initPostion, postionArrayList, 0);

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* SCAN-0 run start \*\*\*\*\*\*\*\*\*\*\*\*\* //");*

        scan0.run();

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* SCAN-0 run end \*\*\*\*\*\*\*\*\*\*\*\*\* //\n");*

        System.out.println("绘图序列: " + scan0.drawFlow);

        System.out.println("寻道长度: " + scan0.getLength());

        System.out.println("平均寻道长度: " + scan0.getLength() / n);

*return* scan0;

    }

    public static C\_SCAN outC\_SCAN1(Postion initPostion, ArrayList<Postion> postionArrayList) {

*//------------------------------------------------------------------------ C-SCAN - 增大方向1*

        System.out.println("\n-------------  C-SCAN算法[增大方向]  -------------");

        C\_SCAN c\_scan1 = *new* C\_SCAN(initPostion, postionArrayList, 1);

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* C-SCAN-0 run start \*\*\*\*\*\*\*\*\*\*\*\*\* //");*

        c\_scan1.run();

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* C-SCAN-0 run end \*\*\*\*\*\*\*\*\*\*\*\*\* //\n");*

        System.out.println("绘图序列: " + c\_scan1.drawFlow);

        System.out.println("寻道长度: " + c\_scan1.getLength());

        System.out.println("平均寻道长度: " + c\_scan1.getLength() / n);

*return* c\_scan1;

    }

    public static C\_SCAN outC\_SCAN0(Postion initPostion, ArrayList<Postion> postionArrayList) {

*//------------------------------------------------------------------------ C-SCAN - 减小方向0*

        System.out.println("\n-------------  C-SCAN算法[减小方向]  -------------");

        C\_SCAN c\_scan0 = *new* C\_SCAN(initPostion, postionArrayList, 0);

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* C-SCAN-0 run start \*\*\*\*\*\*\*\*\*\*\*\*\* //");*

        c\_scan0.run();

*//        System.out.println("// \*\*\*\*\*\*\*\*\*\*\*\*\* C-SCAN-0 run end \*\*\*\*\*\*\*\*\*\*\*\*\* //\n");*

        System.out.println("绘图序列: " + c\_scan0.drawFlow);

        System.out.println("寻道长度: " + c\_scan0.getLength());

        System.out.println("平均寻道长度: " + c\_scan0.getLength() / n);

*return* c\_scan0;

    }

}

## Main.java

package fxui;

import action.C\_SCAN;

import action.FCFS;

import action.SCAN;

import action.SSTF;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.chart.LineChart;

import javafx.scene.chart.NumberAxis;

import javafx.scene.chart.XYChart;

import javafx.scene.control.Button;

import javafx.scene.layout.AnchorPane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import main.Out;

import model.Postion;

import java.util.ArrayList;

import java.util.Random;

import static main.Out.*\**;

import static main.Out.outC\_SCAN0;

public class Main extends Application {

    public static FCFS fcfs;

    public static SSTF sstf;

    public static SCAN scan1;

    public static SCAN scan0;

    public static C\_SCAN c\_scan1;

    public static C\_SCAN c\_scan0;

    public static void main(String[] args){

        launch(args);

    }

    @Override

    public void start(Stage stage) throws Exception {

        LineChart<Number, Number> lineChart =  getView();

        lineChart.setPrefWidth(1500);

        lineChart.setPrefHeight(1000);

*//图形化界面*

        Button button = *new* Button("刷新数据");

        VBox vBox = *new* VBox();

        vBox.setAlignment(Pos.CENTER);

        vBox.getChildren().addAll(button, lineChart);

        AnchorPane root = *new* AnchorPane();

        root.getChildren().addAll(vBox);

        Scene scene = *new* Scene(root);

        stage.setScene(scene);

        stage.setWidth(1600);

        stage.setHeight(1055);

        stage.setAlwaysOnTop(true);

        stage.setTitle("磁盘调度-算法比较");

        stage.show();

        button.setOnAction(*new* EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent actionEvent) {

                LineChart<Number, Number> new\_lineChart = getView();

                new\_lineChart.setPrefWidth(1500);

                new\_lineChart.setPrefHeight(1000);

                vBox.getChildren().remove(1);

                vBox.getChildren().add(new\_lineChart);

            }

        });

    }

    public LineChart<Number, Number> getView(){

*// 生成数据*

        Random random = *new* Random();

        Postion initPostion = *new* Postion(random.nextInt(1500));

        ArrayList<Postion> postionArrayList = *new* ArrayList<>();

        int n = Out.n;

*// 50% -> 0~499*

*for*(int i=0;i<n/2;i++){

            Postion temp = *new* Postion(random.nextInt(500));

            postionArrayList.add(temp);

        }

*// 25% -> 500,999*

*for*(int i=0;i<n/4;i++){

            Postion temp = *new* Postion(random.nextInt(500)+500);

            postionArrayList.add(temp);

        }

*// 25% -> 1000,1500*

*for*(int i=0;i<n/4;i++){

            Postion temp = *new* Postion(random.nextInt(500)+1000);

            postionArrayList.add(temp);

        }

*// 洗牌算法*

        int length = postionArrayList.size();

*for*(int i=0;i<length;i++){

            int iRandNum = (int)(Math.random() \* length);

            Postion temp = postionArrayList.get(iRandNum);

            postionArrayList.set(iRandNum,postionArrayList.get(i));

            postionArrayList.set(i,temp);

        }

        System.out.println("初始点: " + initPostion);

        System.out.println("创建的序列: " + postionArrayList);

        fcfs = outFCFS(initPostion, postionArrayList);

        sstf = outSSTF(initPostion, postionArrayList);

        scan1 = outSCAN1(initPostion, postionArrayList);

        scan0 = outSCAN0(initPostion, postionArrayList);

        c\_scan1 = outC\_SCAN1(initPostion, postionArrayList);

        c\_scan0 = outC\_SCAN0(initPostion, postionArrayList);

*// 绘图*

        NumberAxis x = *new* NumberAxis("index",0, n,1); *// 间距为20*

        NumberAxis y = *new* NumberAxis("道数",0,1500,10);

        LineChart<Number, Number> lineChart = *new* LineChart<>(x,y);

        XYChart.Series<Number, Number> fcfs\_line = *new* XYChart.Series<>();

        fcfs\_line.setName("FCFS");

        XYChart.Series<Number, Number> scan1\_line = *new* XYChart.Series<>();

        scan1\_line.setName("SCAN[增大]");

        XYChart.Series<Number, Number> scan0\_line = *new* XYChart.Series<>();

        scan0\_line.setName("SCAN[减小]");

        XYChart.Series<Number, Number> c\_scan1\_line = *new* XYChart.Series<>();

        c\_scan1\_line.setName("C-SCAN[增大]");

        XYChart.Series<Number, Number> c\_scan0\_line = *new* XYChart.Series<>();

        c\_scan0\_line.setName("C-SCAN[减小]");

        XYChart.Series<Number, Number> sstf\_line = *new* XYChart.Series<>();

        sstf\_line.setName("SSTF");

*for*(int i=0;i<n; i++){

            XYChart.Data<Number, Number> temp0 = *new* XYChart.Data<>(i,fcfs.drawFlow.get(i).getLocation());

            fcfs\_line.getData().add(temp0);

            XYChart.Data<Number, Number> temp2 = *new* XYChart.Data<>(i,scan1.drawFlow.get(i).getLocation());

            scan1\_line.getData().add(temp2);

            XYChart.Data<Number, Number> temp3 = *new* XYChart.Data<>(i,scan0.drawFlow.get(i).getLocation());

            scan0\_line.getData().add(temp3);

            XYChart.Data<Number, Number> temp4 = *new* XYChart.Data<>(i,c\_scan1.drawFlow.get(i).getLocation());

            c\_scan1\_line.getData().add(temp4);

            XYChart.Data<Number, Number> temp5 = *new* XYChart.Data<>(i,c\_scan0.drawFlow.get(i).getLocation());

            c\_scan0\_line.getData().add(temp5);

            XYChart.Data<Number, Number> temp1 = *new* XYChart.Data<>(i,sstf.drawFlow.get(i).getLocation());

            sstf\_line.getData().add(temp1);

        }

        lineChart.getData().add(fcfs\_line);

        lineChart.getData().add(scan1\_line);

        lineChart.getData().add(scan0\_line);

        lineChart.getData().add(c\_scan1\_line);

        lineChart.getData().add(c\_scan0\_line);

        lineChart.getData().add(sstf\_line);

*return* lineChart;

    }

}