# 核心代码

```
package com.nightwingky;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.*;

/**
 * Created by nightwingky on 17-3-10.
 */
public class Process {

    private Random random;

    //轴承寿命(初始)
    private int[] life_axletree = {
            1400 * 60,
            1500 * 60,
            1100 * 60
    };

    //轴承寿命概率
    private double life_probability;

    //总成本
    private int total_cost;

    //时间轴
    private int time;

    //维修人员到达概率
    private double reach_probability;

    //轴承寿命map
    private Map<Double, Integer> axletree_life_map;

    //等待时间map
    private Map<Double, Integer> reach_time_map;

    //当前等待时间
    private int reach_time;

    //坏一换一数据，坏一换三数据
    private List<List<String>> data_process1_1, data_process1_3;
    private List<String> mList;

    public Process() {

        //初始成本为0
        this.total_cost = 0;
        //初始时间为0
        this.time = 0;
        //获取轴承寿命和等待时间
        this.axletree_life_map = MyConst.getAxletree_working_life();
        this.reach_time_map = MyConst.getReach_time();

        //数据列表初始化
        this.data_process1_1 = new ArrayList<List<String>>();
        this.data_process1_3 = new ArrayList<List<String>>();
        this.mList = new ArrayList<String>();
        mList.add("时间(小时)");
        mList.add("等待工人时间(分钟)");
        mList.add("停工时间(分钟)");
        mList.add("累计成本");
        mList.add("新寿命");

        data_process1_1.add(mList);
        data_process1_3.add(mList);

        this.random = new Random();
    }

    //坏1换1流程
```

```
public int process1_1() throws IOException {

    time += life_axletree[2];

    while (time <= MyConst.getTotal_time()) {
        //确定三者中最小寿命
        int min = life_axletree[0];
        int num = 0;
        for (int i = 0; i < life_axletree.length; i++) {
            if (life_axletree[i] < min) {
                min = life_axletree[i];
                num = i;
            }
        }

        //三个轴承减去最小寿命
        for (int i = 0; i < life_axletree.length; i++) {
            life_axletree[i] = life_axletree[i] - min;
        }

        this.mList = new ArrayList<String>();
        this.mList.add(String.valueOf(time / 60.00));
        reach_probability = random.nextDouble();

        //获取reach_time中的key值
        Set<Double> reach_time_key_set = this.reach_time_map.keySet();
        //Set转List
        List<Double> reach_time_key = new ArrayList<Double>(reach_time_key_set);
        Collections.sort(reach_time_key);
        //确定等待时间
        for (int i = 0; i < reach_time_key.size(); i++) {
            if (reach_probability != 1) {
                if (reach_probability >= reach_time_key.get(i) &&
                        reach_probability < reach_time_key.get(i + 1)) {
                    reach_probability = reach_time_key.get(i + 1);
                    reach_time = reach_time_map.get(reach_probability);
                    break;
                }
            } else {
                reach_time = reach_time_map.get(
                        reach_time_key.get(
                                reach_time_key.size()
                        )
                );
            }
        }
        //添加等待时间
        time += reach_time;
        this.mList.add(String.valueOf(reach_time));
        //添加等待成本和机器停工损失
        this.total_cost = this.total_cost + MyConst.getStop_loss() * (
                reach_time + MyConst.getChange_time().get(1));
        //添加换轴承成本
        this.total_cost = this.total_cost + MyConst.getAxletree_price();
        //添加工作人员工资
        this.total_cost = this.total_cost + MyConst.getWage()
                * MyConst.getChange_time().get(1);

        //添加停工时间
        time += MyConst.getChange_time().get(1);
        this.mList.add(String.valueOf(MyConst.getChange_time().get(1)));

        //添加累计成本
        this.mList.add(String.valueOf(total_cost));

        //模拟概率
        life_probability = random.nextDouble();

        //获取key值
        Set<Double> axletree_life_key_set = this.axletree_life_map.keySet();
        List<Double> axletree_life_key = new ArrayList<Double>(axletree_life_key_set);
        Collections.sort(axletree_life_key);
        //确定新轴承寿命
        while (true) {
            for (int j = 0; j < axletree_life_key.size(); j++) {
                if (life_probability != 1) {
                    if (life_probability >= axletree_life_key.get(j)
                            && life_probability < axletree_life_key.get(j + 1)) {
```

```
                        life_probability = axletree_life_key.get(j + 1);
                        life_axletree[num] = axletree_life_map.get(life_probability);
                        break;
                    }
                } else {
                    life_axletree[num] = axletree_life_map.get(
                            axletree_life_key.get(
                                    axletree_life_key.size()
                            )
                    );
                }
            }
            //保证只有一个轴承坏，即数组中元素不重复
            Set<Integer> set = new HashSet<Integer>();
            for (int a : life_axletree) {
                set.add(a);
            }
            if (set.size() == life_axletree.length) {
                break;
            }
        }

        mList.add(life_axletree[0] / 60 + "/" + life_axletree[1] / 60 + "/" +
                life_axletree[2] / 60);
        //确定三者中最小寿命
        min = life_axletree[0];
        for (int x : life_axletree) {
            if (x < min) {
                min = x;
            }
        }
        //加入总时间
        time += min;
        data_process1_1.add(mList);
    }

//        System.out.println("坏1换1方案：");
    printList(data_process1_1, "plan1_1.html");

    return total_cost;
}

//坏1换3流程
public int process1_3() throws IOException {
    time += life_axletree[2];

    while (time <= MyConst.getTotal_time()) {
        this.mList = new ArrayList<String>();
        this.mList.add(String.valueOf(time / 60.00));
        reach_probability = random.nextDouble();
        //获取reach_time中的key值
        Set<Double> reach_time_key_set = this.reach_time_map.keySet();
        //Set转List
        List<Double> reach_time_key = new ArrayList<Double>(reach_time_key_set);
        Collections.sort(reach_time_key);
        //确定等待时间
        for (int i = 0; i < reach_time_key.size(); i++) {
            if(reach_probability != 1) {
                if(reach_probability >= reach_time_key.get(i) &&
                        reach_probability < reach_time_key.get(i + 1)) {
                    reach_probability = reach_time_key.get(i + 1);
                    reach_time = reach_time_map.get(reach_probability);
                    break;
                }
            } else {
                reach_time = reach_time_map.get(
                        reach_time_key.get(
                                reach_time_key.size()
                        )
                );
            }
        }
        //添加等待时间
        time += reach_time;
        this.mList.add(String.valueOf(reach_time));
        //添加等待成本和机器停工损失
        this.total_cost = this.total_cost + MyConst.getStop_loss() * (
                reach_time + MyConst.getChange_time().get(3));
```

```
            //添加换轴承成本
            this.total_cost = this.total_cost + MyConst.getAxletree_price() * 3;
            //添加工作人员工资
            this.total_cost = this.total_cost + MyConst.getWage()
                    * MyConst.getChange_time().get(3);


            //添加停工时间
            time += MyConst.getChange_time().get(3);
            this.mList.add(String.valueOf(MyConst.getChange_time().get(3)));

            //添加累计成本
            this.mList.add(String.valueOf(total_cost));

            //确定新轴承寿命
            for (int i = 0; i < life_axletree.length; i++) {
                //模拟概率
                life_probability = random.nextDouble();
                //获取key值
                Set<Double> axletree_life_key_set = this.axletree_life_map.keySet();
                List<Double> axletree_life_key = new ArrayList<Double>(axletree_life_key_set);
                Collections.sort(axletree_life_key);
                //确定寿命
                for (int j = 0; j < axletree_life_key.size(); j++) {
                    if (life_probability != 1) {
                        if (life_probability >= axletree_life_key.get(j)
                                && life_probability < axletree_life_key.get(j + 1)) {
                            life_probability = axletree_life_key.get(j + 1);
                            life_axletree[i] = axletree_life_map.get(life_probability);
                            break;
                        }
                    } else {
                        life_axletree[i] = axletree_life_map.get(
                                axletree_life_key.get(
                                        axletree_life_key.size()
                                )
                        );
                    }
                }
            }

            mList.add(
                    life_axletree[0] / 60 + "/" +
                    life_axletree[1] / 60 + "/" +
                    life_axletree[2] / 60);
            //确定三者中最小寿命
            int min = life_axletree[0];
            for (int x : life_axletree) {
                if (x < min) {
                    min = x;
                }
            }
            //加入总时间
            time += min;
            data_process1_3.add(mList);
        }

//        System.out.println("坏1换3方案：");
        printList(data_process1_3, "plan1_3.html");

        return total_cost;
    }

    //打印结果
    private void printList(List<List<String>> dataList, String title) throws IOException {
        //控制台输出
        for (List<String> l : dataList) {
            System.out.println(l);
        }

        //文件输出
        String path = title;

        File file = new File(path);
        if (!file.exists()) {
            file.createNewFile();
        }
```

```
        FileOutputStream fileOutputStream = new FileOutputStream(file, true);

        StringBuilder sb = new StringBuilder();
        sb.append("<html>\n<h2>信管14-2  140614406  
            阙琨洋</h2>\n<h3>轴承修理模拟" + title + "</h3>");
        sb.append("<table border=\"2\">\n");
        fileOutputStream.write(sb.toString().getBytes("utf-8"));

        for (List<String> m : dataList) {
            sb = new StringBuilder();
            sb.append("<tr>");
            for (String s : m) {
                sb.append("<td>" + s + "</td>");
            }
            sb.append("</tr>\n");
            fileOutputStream.write(sb.toString().getBytes("utf-8"));
        }

        sb = new StringBuilder();
        sb.append("</table>\n</html>\n");
        fileOutputStream.write(sb.toString().getBytes("utf-8"));

        fileOutputStream.close();

    }

    //两个过程分别跑100次得到成本输出至文件
    public void getResult() throws IOException {
        List<Integer> data1_1 = new ArrayList<Integer>();
        for (int i = 0; i < 100; i++) {
            data1_1.add(new Process().process1_1());
        }

        List<Integer> data1_3 = new ArrayList<Integer>();
        for (int i = 0; i < 100; i++) {
            data1_3.add(new Process().process1_3());
        }

        System.out.println(data1_1);
        System.out.println(data1_3);

        //输出至文件
        File file = new File("cost.html");
        if (!file.exists()) {
            file.createNewFile();
        }
        FileOutputStream fileOutputStream = new FileOutputStream(file, false);
        StringBuilder sb = new StringBuilder();
        sb.append("<html>\n<h2>信管14-2  140614406  阙琨洋</h2>\n<h3>100次模拟后结果</h3>");

        sb.append("<table border=\"2\">\n<tr><th>坏1换1</th><th>坏1换3</th></tr>\n");
        fileOutputStream.write(sb.toString().getBytes("utf-8"));

        for (int i = 0; i < data1_1.size(); i++) {
            sb = new StringBuilder();
            sb.append("<tr><td>" + data1_1.get(i) + "</td>\n");
            sb.append("<td>" + data1_3.get(i) + "</td></tr>\n");
            fileOutputStream.write(sb.toString().getBytes("utf-8"));
        }

        sb = new StringBuilder();
        sb.append("</table>\n</html>\n");
        fileOutputStream.write(sb.toString().getBytes("utf-8"));

        fileOutputStream.close();
    }
}
```