

# Тестовое задание для java-разработчика

Результат: выложить на GitHub

### Задание 1

Упрощенный Match-making

Необходимо промоделировать работу матчмейкера онлайн-игры.

В процессе работы матчмейкер получает команды регистрации игрока в очереди, вида (user rank), где

user - уникальный идентификатор пользователя

rank - уровень пользователя в игре, от 1 до 30

enter\_time(user) - время получения команды регистрации

Матчмейкер должен создавать матчи для 8 игроков так, чтобы игроки находились в очереди как можно меньше.

Для каждой пары игроков (A,B) в созданном матче должны выполняться условия

abs(A.rank - B.rank) <= waiting\_time(A) / 5000 + waiting\_time(B) / 5000 , где

waiting\_time(A) = (текущее время) - enter\_time(A)

Результат

Вывести в консоль результат работы матчмейкера.

Набор строк вида (time user[0] user[1] ... user[7]), где

time - время создания матча

user[0] user[1] ... user[7] - список игроков, которые отобраны для данного матча

### Задание 2

Отрефакторить следующий код

import java.io.File;  
  
  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.IOException;  
  
interface DataConnection {  
  
 int loadDatas(int sum) throws Exception;  
  
 void saveData(int year, int qq) throws IOException;  
}  
  
public class MyApp implements DataConnection {  
  
  
  
 public static class MyAppFactory {  
  
 public static MyApp create(String y) {  
 return new MyApp(y);  
 }  
 }  
  
 public MyApp(String y) {  
 this.y = y;  
 }  
 private String y;  
 private static int COUNT = 0;  
 private static int COUNT1 = 0;  
 protected static int startYear = 1990;  
 protected static int endYear = 2020;  
 /\*\*  
 \* @param args  
 \*/  
 public static void main(String[] args) {  
 try {  
 System.out.println("app v.1.13");  
 for (int i = startYear; i < endYear; i++) {  
 int sum = 0;  
 COUNT = 0;  
 COUNT1 = 0;  
 String y = i + "";  
 sum = new MyAppFactory().create(y).loadDatas(sum);  
 double qq = sum > 0 ? (double) sum / (double) COUNT : 0;  
 if (qq > 0) {  
 System.out.println(i + " " + qq);  
 }  
 new /\*new\*/ MyAppFactory().create(y).saveData(i, (int) qq);  
 }  
 System.out.println("gotovo");  
 } catch (Exception e) {  
 System.out.println("oshibka!");  
 }  
 }  
  
 public int loadDatas(int sum) throws Exception {  
 File file = new File("1.txt");  
 //try {  
 //FileOutputStream fis = new FileOutputStream(file);  
 FileInputStream fis = new FileInputStream(file);  
 String s = "";  
 int i = fis.read();  
 do {  
 s = s + new String(new byte[]{(byte) i});  
 i = fis.read();  
 } while (i != -1);  
 int begin = 0;  
 while (true) {  
 int e = s.indexOf("\n", begin + 1);  
 if (e == -1) {  
 break;  
 }  
 String ss = s.substring(begin, e);  
 //System.out.println(ss);  
 String[] sss = ss.split(" ");  
 for (String string : sss) {  
 //System.out.println(string);  
 }  
 if (sss[2].contains(this.y) || sss[2].contains(y)) {  
 sum = sum + Integer.parseInt(sss[3]);  
 }  
 COUNT++;  
 begin = e;  
 }  
 return sum;  
 }  
  
 public void saveData(int year, int qq) throws IOException {  
 FileOutputStream fis = new FileOutputStream(new File("statistika.txt"), true);  
 String s = new String();  
 s = COUNT1 + " " + year + " " + qq + "\n";  
 fis.write(s.getBytes());  
 COUNT1++;  
 }  
}  
