KHOA CÔNG NGHỆ THÔNG TIN - ĐH NÔNG LÂM TP. HCM

Bộ môn: Mạng máy tính và truyền thông

Môn học: Lập trình mạng

Giảng viên: ThS. Nguyễn Xuân Vinh

Đề bài: Viết CT lưu/Đọc danh sách sinh viên xuống file nhị phân (Lưu từng thuộc tính). Hỗ trợ đọc thông tin sinh viên ở vị trí bất kỳ.

CÁCH 1: TAB

Lưu bằng file text, mỗi sinh viên là một dòng văn bản, mỗi thuộc tính cách nhau bằng "tab" character.

Ví dụ:

SID	Name	GPA
05130113	Nguyễn Xuân Vinh	7.2
05130113	Nguyễn Thanh An	7.4
05130113	Châu Quốc Tuấn	7.8

Ưu điểm: nhanh gọn, dễ hiện thực.

Nhược điểm: trong Name không được có ký tự tab (\t)

Student.java

```
}
    @Override
    public String toString() {
        return "Student [sid=" + sid + ", name=" + name + ", gpa=" + gpa + "]";
    public String getSid() {
        return sid;
    public void setSid(String sid) {
        this.sid = sid;
    public String getName() {
        return name;
    public void setName(String name) {
        this.name = name;
    public float getGpa() {
        return gpa;
    public void setGpa(float gpa) {
        this.gpa = gpa;
}
```

StudentUtil.java

```
package solution1_tab;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;
public class StudentUtil {
    public static void write(String fileName, Student... students) throws IOException {
        PrintWriter pw = new PrintWriter(new File(fileName));
        for (Student's : students) {
   pw.println(s.getSid() + "\t" + s.getName() + "\t" + s.getGpa());
        pw.close();
    }
    public static List<Student> read(String fileName) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader(new File(fileName)));
        List<Student> students = new ArrayList<Student>();
        String line = "";
        StringTokenizer tokenizer;
        Student student;
        while ((line = br.readLine()) != null) {
            tokenizer = new StringTokenizer(line, "\t");
            student = new Student();
            student.setSid(tokenizer.nextToken().trim());
            student.setName(tokenizer.nextToken().trim());
```

```
student.setGpa(Float.parseFloat(tokenizer.nextToken().trim()));
               students.add(student);
         br.close();
         return students;
     }
     public static Student read(int index) {
         throw new UnsupportedOperationException();
     }
     public static boolean update(int index, Student student) {
         throw new UnsupportedOperationException();
     public static void main(String[] args) throws IOException {
         Student s1 = new Student("05130001", "Nguyễn Xuân Vinh", 9.2f);
Student s2 = new Student("05130002", "Nguyen Thanh An", 7.5f);
Student s3 = new Student("05130003", "Châu Quốc Tuấn", 8.5f);
         write("students1.txt", s1, s2, s3);
         List<Student> students = read("students1.txt");
         for (Student s : students) {
               System.out.println(s);
     }
}
```

CÁCH 2: FIXED SIZE

Quy định kích thước cố định cho từng trường:

- SID: 8 byte
- Name: 20 byte maximum (2 byte Name Length (NL) + 18 byte name text)
- GPA: 4 byte floating point number
- ➡ Mỗi sinh viên có độ dài cố định là 32 bytes

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29			32
SID NL NAME																				GF	Α										
0	5	1	3	0	0	0	1	0	16	Ν	G	U	Υ	Ε	Ν		Χ	U	Α	Ν		V	ı	Ν	Н			7.5 float			t
0	5	1	3	0	0	0	2	0	14	С	Н	Α	U		Q	U	0	С		Т	U	Α	Ν					8.5 float			t

Student.java

```
package solution2;
/** 32 Bytes per student */
```

```
public class Student {
        public static final int SID_LENGTH = 8;
        public static final int NAME_MAX_LENGTH = 20;
        // 8 bytes
        private String sid;
        // 20 bytes (2 bytes Name Length [NL] + 18 bytes Name Text [NT])
        private String name;
        // 4 bytes
        private float gpa;
        public Student() {
                // no arg
        }
        @Override
        public String toString() {
                return "Student [sid=" + sid + ", name=" + name + ", gpa=" + gpa + "]";
        public Student(String sid, String name, float gpa) {
                this.sid = sid;
                this.name = name;
                this.gpa = gpa;
        }
        public String getSid() {
                return sid;
        }
        public void setSid(String sid) {
                this.sid = sid;
        }
        public String getName() {
                return name;
        public void setName(String name) {
                this.name = name;
        }
        public float getGpa() {
                return gpa;
        }
        public void setGpa(float gpa) {
                this.gpa = gpa;
}
```

StudentUtil.java

```
package solution2;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ByteBuffer;

public class StudentUtil {
    public static void write(String fileName, Student... students) throws IOException {
```

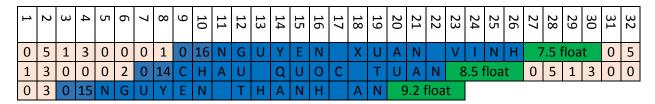
```
DataOutputStream dos = new DataOutputStream(new FileOutputStream(fileName));
        dos.writeInt(students.length);
        for (Student s : students) {
                 dos.write(s.getSid().getBytes());
                 byte[] name = s.getName().getBytes("UTF-8");
                 byte[] nameLength = ByteBuffer.allocate(2).putShort((short) name.length).array();
                 byte[] blank = new byte[Student.NAME_MAX_LENGTH - (name.length + 2)];
                 dos.write(nameLength);
                 dos.write(name);
                 dos.write(blank);
                 dos.writeFloat(s.getGpa());
        dos.close();
}
public static Student[] read(String fileName) throws IOException {
        DataInputStream dis = new DataInputStream(new FileInputStream(fileName));
        int size = dis.readInt();
        Student[] students = new Student[size];
        Student s;
        byte[] sid = new byte[Student.SID_LENGTH];
        byte[] nameLength = new byte[2];
        byte[] nameBytes;
        for (int i = 0; i < size; i++) {</pre>
                 s = new Student();
                 // read sid
                 dis.read(sid);
                 s.setSid(new String(sid, "UTF-8"));
                 // read name
                 dis.read(nameLength);
                 short length = ByteBuffer.wrap(nameLength).getShort();
                 nameBytes = new byte[length];
                 dis.read(nameBytes);
                 s.setName(new String(nameBytes, "UTF-8"));
                 int blankLength = Student.NAME_MAX_LENGTH - (length + 2);
                 dis.skip(blankLength);
                 // read GPA
    s.setGpa(dis.readFloat());
                 students[i] = s;
        dis.close();
        return students;
}
public static void main(String[] args) throws IOException {
        Student s1 = new Student("05130001", "Nguyen Xuan Vinh", 9.2f);
Student s2 = new Student("05130002", "Nguyen Van A", 7.5f);
Student s3 = new Student("05130003", "Phan Thi B", 8.5f);
        write("students2.txt", s1, s2, s3);
        Student[] students = read("students2.txt");
        for (Student s : students) {
                 System.out.println(s);
}
```

}

CÁCH 3: VARIABLE NAME LENGTH

Quy định kích thước cố định cho SID và GPA, kích thước tùy ý cho Name:

- SID: 8 fixed bytes
- Name: variable bytes maximum (2 bytes Name Length (NL) + variable bytes NAME text)
- GPA: 4 fixed bytes floating point number
- ➡ Mỗi sinh viên có độ dài KHÔNG CỐ ĐỊNH (VARIABLE)



Ưu điểm:

- Mỗi sinh viên có độ dài bất kỳ.
- Tiết kiệm bộ nhớ.
- Có thể dùng hàm readUTF() của DataInputStream hoặc RandomAccessFile để đọc trực tiếp Name mà không cần đọc 2 lần (mảng byte chứa độ dài + mảng byte chứa tên).

Nhươc điểm:

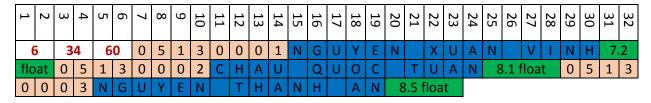
- Để đọc sinh viên ở vị trí thứ N cần phải "lướt" qua các sinh viên trước đó. ("lướt" qua vì chỉ cần đọc NL: Name Length là có thể skip qua).

CÁCH 4: VARIABLE NAME LENGTH OPTIMIZED

Cũng như cách trên, bên cạnh đó thêm header chứa thông tin offset của từng user:

Offset: 2 bytes, vi trí bắt đầu của một student (0 < offset < 65535)

Giả sử, mỗi sinh viên có độ dài khoảng 32 bytes → có thể chứa được khoảng tối đa 2047 students



Ưu điểm:

- Có thể truy vấn (đọc/ghi) sinh viên ở vị trí bất kỳ.
- Tiết kiệm bộ nhớ.

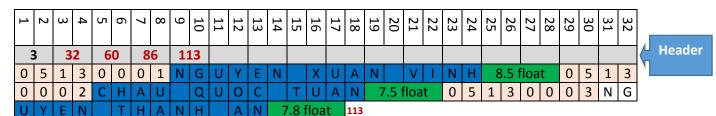
Nhươc điểm:

- Khi thêm sinh viên cần phải điều chỉnh lại tất cả offset.

CÁCH 5: VARIABLE NAME LENGTH OPTIMIZED + FIXED HEADER

Tương tự cách trên, tuy nhiên cần quy định header có thể chứa cố định số lượng sinh viên.

Ví dụ: quy định file có thể chứa tối đa 16 sinh viên, mỗi OFFSET có độ dài 2 bytes → tổng độ dài header = 32 bytes



3: Number of records

32, 60, 86: offset of each student

113: offset of new next student

Ưu điểm:

- Quản lý tối ưu bộ nhớ.
- Có thể thêm sinh viên trong giới hạn cho phép của header mà không ảnh hưởng dữ liệu trước
 đó.
- Có thể truy vấn một record bất kỳ nhanh chóng.

Nhươc điểm:

- Cách hiện thực phức tạp.

Student.java

package solution5;

/** variable bytes per student */

```
public class Student {
        public static final int SID_LENGTH = 8;
        public static final int GPA_LENGTH = 4;
        // 8 fixed bytes
        private String sid;
        // variable bytes (2 bytes Name Length [NL] + variable bytes Name Text [NT])
        private String name;
        // 4 fixed bytes
        private float gpa;
        public Student() {
                // no arg
        }
        @Override
        public String toString() {
                return "Student [sid=" + sid + ", name=" + name + ", gpa=" + gpa + "]";
        public Student(String sid, String name, float gpa) {
                this.sid = sid;
                this.name = name;
                this.gpa = gpa;
        }
        public String getSid() {
                return sid;
        }
        public void setSid(String sid) {
                this.sid = sid;
        }
        public String getName() {
                return name;
        public void setName(String name) {
                this.name = name;
        }
        public float getGpa() {
                return gpa;
        }
        public void setGpa(float gpa) {
                this.gpa = gpa;
}
```

StudentUtil.java

```
package solution5;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.Arrays;

/**
    * Support 15 students maximum
    * @author vinhnx
    *
```

```
public class StudentUtil {
    public static final int MAX_RECORD = 15;
    public static final int HEADER_LENGTH = (MAX_RECORD * 2) + 2;
        public static void write(String fileName, Student... students) throws IOException {
            RandomAccessFile raf = new RandomAccessFile(fileName, "rw");
            short[] offsets = new short[students.length + 1]; // number of offset = number of student + 1
            short offset = HEADER_LENGTH;
            int i = 0;
            offsets[i] = HEADER_LENGTH;
            // write number of records
            raf.writeShort(students.length);
            // write empty header
        raf.write(new byte[HEADER_LENGTH - 2]);
            for ( ; i< students.length; i++) {</pre>
                    Student s = students[i];
                    // write SID
                    byte[] sid = s.getSid().getBytes();
                    raf.write(sid, 0, Student.SID_LENGTH); // need to make sure just 8byte are written
                    // write name
                         byte[] name = s.getName().getBytes("UTF-8");
                         raf.write(name);
                         // write GPA (number of bytes = Student.GPA LENGTH)
                        raf.writeFloat(s.getGpa());
                         offset += (Student.SID_LENGTH + name.length + Student.GPA_LENGTH);
                         offsets[i+1] = offset;
                // write headers
                raf.seek(2); // move to beginning of offset part (after number-of-record part)
                for (short o : offsets)
                    raf.writeShort(o);
                System.out.println("offsets:" + Arrays.toString(offsets));
                raf.close();
        }
        public static Student[] read(String fileName) throws IOException {
                DataInputStream dis = new DataInputStream(new FileInputStream(fileName));
                // read number of records
                short count = dis.readShort();
                System.out.println("count:" + count);
                short[] offsets = new short[count + 1]; // number of offset = number of student + 1
                for (int i = 0; i <= count; i++)</pre>
                    offsets[i] = dis.readShort();
                System.out.println("offsets:" + Arrays.toString(offsets));
                // skip empty header
                dis.skip(HEADER_LENGTH - 4 - (count * 2));
                Student[] students = new Student[count];
                Student student;
                byte[] sid = new byte[Student.SID_LENGTH];
                byte[] name;
                for (int i = 0; i < count; i++) {</pre>
                    short off1 = offsets[i];
                    short off2 = offsets[i+1];
                    short length = (short) (off2 - off1);
```

```
short nameLength = (short) (length - (Student.SID_LENGTH + Student.GPA_LENGTH));
                 name = new byte[nameLength];
                 dis.read(sid);
                 dis.read(name);
                 student = new Student();
                 student.setSid(new String(sid));
                 student.setName(new String(name, "UTF-8"));
                 student.setGpa(dis.readFloat());
                 students[i] = student;
             dis.close();
             return students;
    }
     * @param fileName
     * @param index: from 0
     * @return
     * @throws IOException
    public static Student readAt(String fileName, int index) throws IOException {
    DataInputStream dis = new DataInputStream(new FileInputStream(fileName));
    // read number of records
    short count = dis.readShort();
    if (index >= count) {
        dis.close();
        return null;
    }
    short skipByte = (short) (index * 2);
    dis.skip(skipByte);
    short offset = dis.readShort();
    short offsetNext = dis.readShort();
    dis.skip(offset - (skipByte + 6));
    short length = (short) (offsetNext - offset);
    short nameLength = (short) (length - (Student.SID_LENGTH + Student.GPA_LENGTH));
    Student student = new Student();
    byte[] sid = new byte[Student.SID_LENGTH];
    byte[] name = new byte[nameLength];
    dis.read(sid);
    dis.read(name);
    student.setSid(new String(sid));
    student.setName(new String(name, "UTF-8"));
    student.setGpa(dis.readFloat());
    dis.close();
    return student;
}
    public static void main(String[] args) throws IOException {
        String fileName = "students5.txt";
             Student s1 = new Student("05130001", "Nguyen Xuan Vinh", 9.2f);
Student s2 = new Student("05130002", "Chau Quoc Tuan", 7.5f);
Student s3 = new Student("05130003", "Nguyen Thanh An", 8.5f);
             write(fileName, s1, s2, s3);
             System.out.println("----");
             Student[] students = read(fileName);
             for (Student s : students) {
                      System.out.println(s);
             }
```

```
System.out.println("----");
Student s = readAt(fileName, 2);
System.out.println(s);
}
```