哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2021 年春季学期

计算学部《软件构造》课程

课程报告

姓名	卢兑玩
学号	L170300901
班号	1803501
电子邮件	nty0725@naver.com
手机号码	+82 1059395270

已知如下业务背景：

1. 疫情期间需要记录核酸检验的结果和日期，为此设计了护士类（Nurse）和检验记录类（Record）。
2. 每次检验会生成检验记录（Record），每名护士（Nurse）维护被其检验的人员集合（Set<String>），检验记录集合（List<Record>），所有的检验记录需按日期先后顺序存储。
3. 每名护士具有姓名（String）属性，且不同护士之间的姓名不能相同。
4. 检验记录（Record）中记录了被检验的人员姓名（String）、检验时间（Date）和检验结果（boolean）。
5. 护士之间可以相互代班，每名护士拥有一个列表（Set<Nurse>），该列表中存储了能够代替当前护士上班的其他护士，护士之间的代班是相互的。
6. 护士姓名不能为空，且为英文字符，姓在前名在后，姓和名的首字母大写，如：LiXiaoming 或 LiMing，被检验人姓名的规则同护士姓名。

针对此业务背景的部分代码如下：

```
1   public class Nurse {
      //rep
2   private final String name;
3   private final Set<Nurse> switches;
4   private final Set<String> persons;
5   private final List<Record> records;

// methods
//构造函数
6   public Nurse (string name) {
7       this.name=name;
```

```
8       this.switches=new HashSet< >( );
9       this.persons=new HashSet< >( );
10      this.records=new ArrayList<>( );
11    }

//增加一条检验记录，同时将被检验人增加到 persons 集合中
12    public void insertRecord (Record one) {
13       this.records.add (one);
14       this.persons.add (one.getName( ));
         //利用 Collections 类提供的 sort 方法对 records 进行排序
15       …….
16    }

//根据被检验人姓名和检验时间删除相应的检验记录
17    public void removeRecord (String name, Date time) {
18       Iterator iter=this.records.iterator ( );
19       while (iter.hasNext ( )){
20          Record r=iter.next ( );
21          int compareTo = r.getDate.compareTo(time)
22          if (r.getName( ).equals (name) && compareTo){
23                iter.remove ( );
24                break;
25            }
26        }
27    }
```

```
//按输入的被检验人姓名和日期获取检验记录
28  public Record getOneRecord (String name, Date time) {
29      Iterator iter=this.records.iterator ( );
30      while (iter.hasNext ( )){
31          Record r=iter.next ( );
32          int compareTo = r.getDate.compareTo(time)
33          if (r.getName( ).equals (name) && compareTo)
34              return r;
35       }
36  }

//按输入的条形码获取检验记录
37  public Record getOneRecordbyLabel (String label) {
38      Iterator iter=this.records.iterator ( );
39      while (iter.hasNext ( )){
40          Record r=iter.next ( );
41          if (r.getLabel.equals (label)
42              return r;
43      }
44  }

//将一名护士加入到当前护士的代班列表中
45  public void addSet (Nurse other){}
46      this.switches.add(other);
```

```
47        other.addSet(this);
48    }

//将一名护士从当前护士的代班列表中删除
49    public void deleteSet (Nurse other){
50        this.switches.remove(other);
51        other.deleteSet(this);
52    }

//通过姓名判断两名护士是否是同一人
53    public boolean equals (Object other){
54        Nurse a=(Nurse) other;
55        if (this.nurse.name.toLowerCase( ).equals (a.name))
56            return true;
57        else
58            return false;
59    }

//haseCode 方法
60    public int hashCode( ){
61        return this.name.length( );
62    }

//  利用正则表达式判断字符串是否符合一条标准的检验记录。
63    public boolean read (String content){
64        String regEx = "*****";
65        Pattern pattern= Pattern.compile(regEx);
```

```
66      Matcher m = pattern.matcher(content);
67      if (m.matches( ))
68          return true;
69      else
70          return false;
71  }
72 }
```

//检验记录类，记录了被检验人，检验日期，检验结果和检验记录条形码，其中检验//结果为布尔值，“Y”代表结果阳性，“N”代表结果阴性。

```
73 public class Record{
//rep
74  public final String person;
75  public final Date time;
76  public final boolean result;
77  public final String label;

//methods
78  public Record(String name, Date time, boolean result){
79      this.person=name;
80      this.time=time;
81      this.result=result
82      this.label=generateLabel( );
83  }
84  public String getName ( ){
85      return this.person;
```

```
86    }
87    public Date getDate ( ){
88          return this.time;
89    }
90    public String generateLabel ( ){
                //省略了生成标签的方法
91                  ……
92    }
93 }
```

1、 请根据需求以注释的形式给出 Nurse 类的 RI（表示不变性）

```
private final String name;
// rep invariant
// name is a string that can be divided into two sub strings and both o
f the strings must contain a capital letter followed by several lowerca
se letters
private final Set<Nurse> switches;
// rep invariant
// switches is a set and each element in this set should be an object b
elongs to the self difined class Nurse
// it could be empty
private final Set<String> persons;
// rep invariant
// persons is a set of String and each elements have the same requireme
nts with name
// it could be empty
private final List<Record> records;
// rep invariant
// switches is a list and each element in this set should be an object
belongs to the self difined class Record
// it could be empty
// for each element s   s.person is a string and should meet the requir
ments of name   s.date is a format of time     s.result is a boolean wh
ich should be either true or false
```

2、 请分析上述 Record 类中的代码中是否存在表示暴露（Rep Exposure）缺陷，如果存在，请指出具体位置（代码行数），并予以修改。

存在，因为 Date 是 mutable 类的，因此对其返回值的更改会导致其对象的值发生变化。

更改方法：

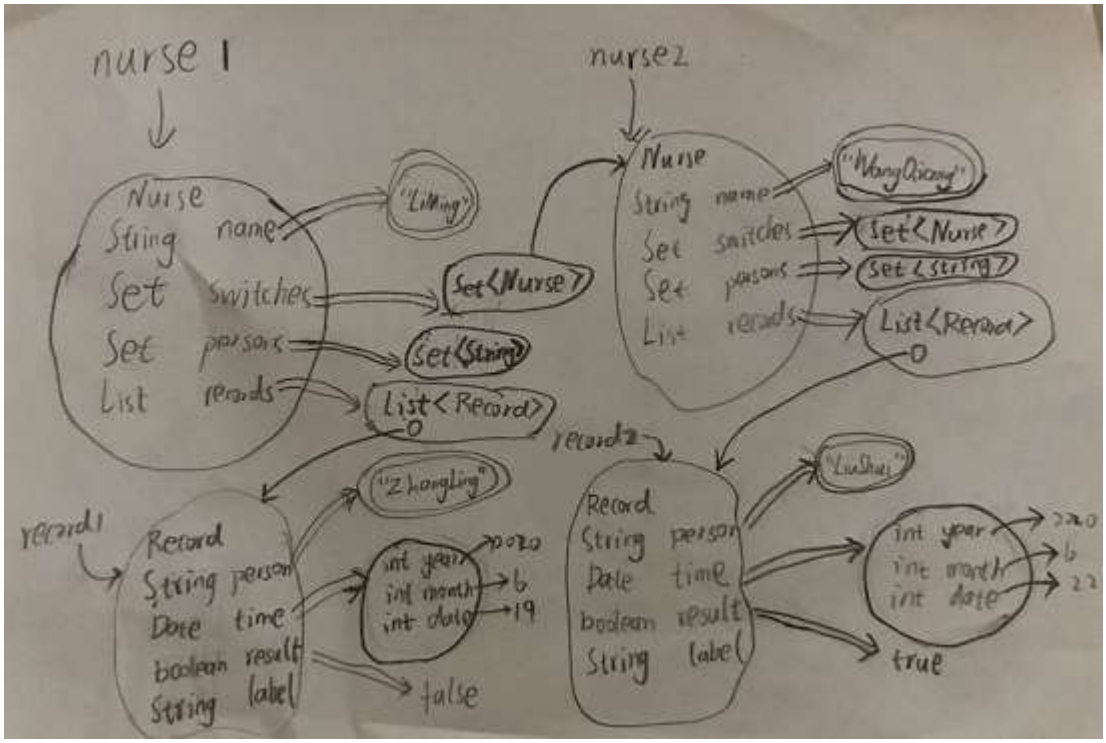
方法一：使用 LocalDate 类。日期为 `LocalDate.of(year, month, date);`。

方法二：使用防御性克隆：

```
public Date getDate(){
return new Date(time.getYear(), time.getMonth(), time.getDate());
}
```

3、 客户端按照下面的代码运行后，请给出运行后的 snapshot diagram 图示。

nurse1=new Nurse (“LiMing”);
nurse2=new Nurse (“WangQiang”);
record1=new Record (“ZhangLing”, new Date(“2020-06-19”), false);
record2=new Record (“LiuShui”, new Date(“2020-06-22”), true);
nurse1. insertRecord (record1);
nurse2. insertRecord (record2);
nurse1. addSet(nurse2);



4、 请判断代码中实现的判相等方法（53-59）是否合理，并给出理由，如不合理，请给出你认为合理的判相等方法（给出理由和修改后的代码）

不合理，因为两个人的名字可能读音相同，因此只判断内容的情况下判断的时候可能会把两个人判断为同一个人。所以应该同时判断内容和地址。代码如下：

```
public boolean equals (Object other){
    Nurse a=(Nurse) other;
    if (this.nurse.name==a.name)
      return true;
    else
      return false;
}
```

5、 为对检验记录（Record）进行更好的保管，系统会自动为每个检测记录生成条形码（代码行 90-92），条形码的组成规则如下：

- 条形码由两部分组成，分别对应护士姓名和随机生成的编码，中间用“_”分割；
- 护士姓名为英文字符，姓在前名在后，姓和名的首字母大写，如：“LiXiaoming”或“LiMing”；
- 随机生成的编码由英文小写字符和数字混合而成，共 8 位，首位必须为英文字符，其它位为英文或数字；

请跟据上述规则为 Nurse 类中的方法 getOneRecordbyLabel 撰写测试策略 testing strategy ，并给出测试用例。

判断输入的标签，返回值是否是想得到的 record

```
public class RecordTest {


  Record s1 = new Student("LiMing", new Date(2020, 2, 3), false);
  Record s2 = new Student("LiNing", new Date(2020, 3, 3), false);
  Record s3 = new Student("LiSen", new Date(2020, 4, 6), false);
  Record s4 = new Student("XuXiu", new Date(2020, 5, 7), false);
  Record s5 = new Student("KimDon", new Date(2020, 1, 4), false);
  Record s6 = new Student("KimGe", new Date(2020, 4, 23), false);

  Nurse n1 = new Nurse("KimTai")
  Nurse n2 = new Nurse("LiDong")
  Nurse n3 = new Nurse("LiXi")

  String l1 = s1.getLabel
  String l2 = s2.getLabel
  String l3 = s3.getLabel
  String l4 = s4.getLabel
  String l5 = s5.getLabel
  String l6 = s6.getLabel


  n1.insertRecord(s1)
  n1.insertRecord(s2)
  n1.insertRecord(s3)
  n2.insertRecord(s4)
  n2.insertRecord(s5)
```

```
  n3.insertRecord(s6)

  @Test
  public void RecTest(){
    assertEquals(n1.getOneRecordbyLabel(l1), s1)
    assertEquals(n1.getOneRecordbyLabel(l2), s2)
    assertEquals(n1.getOneRecordbyLabel(l3), s3)
    assertEquals(n2.getOneRecordbyLabel(l4), s4)
    assertEquals(n2.getOneRecordbyLabel(l5), s5)
    assertEquals(n3.getOneRecordbyLabel(l6), s6)
}
```

6、 Nurse 类中的 read 方法（63-72 行）可利用正则表达式判断输入的字符串（content）是否是一条标准的检验记录。检验记录的形式如题目 5 所示。请在下面给出正确的正则表达式 regEx。

```
String regEx = "[A-Z]{1}[a-z]{1,}[A-Z]{1}[a-z]{1,}_[a-z][a-z0-9]{7}";
```

7、 检验记录集合（List<Record>）中所有的检验记录需按日期先后顺序存储，故此需要实现检验记录排序方法。考虑到职责分配原则，Nurse 类中不希望实现对 Record 对象判断先后次序的方法，而是直接调用 Collections 类提供的 sort 函数对 List<Record>进行排序（15 行）。故此，请给出使 Record 对象能够比较大小的设计方案。

```
public void insertRecord (Record one) {
    this.records.add (one);
    this.persons.add (one.getName( ));
      //利用 Collections 类提供的 sort 方法对 records 进行排序
    Collections.sort(this.records,new Comparator<Record>(){
        @Override
        public int compare(Record o1, Record o2) {
          /*sort by time*/
          return o1.after(o2.getDate());
        }
    }
}
```

8、 代码中的 read 方法（63-72 行）利用正则表达式判断输入的字符串（content）是否是一条标准的检验记录。此种方式读取效率有限，故而可实现 read 方法的一个重载方法（overload），用于从文件中读取连续字符串，考虑从文件中读取字符串时会出现某些异常情况，为可能的异常编写异常类，用于提示用

户异常发生的原因，例如护士姓名首字母非大写，请至少给出两种异常情况下的异常类。

请根据上述需求：

1） 首先设计重载方法；

将记录存储在文件中，没行存储 1 条记录。通过从文件中按行读取。再判断每一行的记录是否符合要求。

```
public boolean read (String filename){
  boolean res[];
  String regEx = "[A-Z]{1}[a-z]{1,}[A-Z]{1}[a-z]{1,}_[a-z][a-z0-9]{7}";

  Pattern pattern= Pattern.compile(regEx);
  File file = new File(fileName);
  BufferedReader reader = null;
  try {
    reader = new BufferedReader(new FileReader(file));
    String tempString = null;
    int line = 1;
    while ((tempString = reader.readLine()) != null) {
      Matcher m = pattern.matcher(content);
      if (m.matches( ))
        res.add(true);
      else
        res.add(false);
      }
      line++;
    }
    reader.close();
  } catch (IOException e) {
    e.printStackTrace();
  }

  return res
}
```

2）给出可能发生的异常类的定义和实现。

在按行读取的过程中，设计了两个异常类，分别是姓名首字母未大写和_之后的

数据不是 8 位。实现如下：

```
public boolean read (String filename){
  boolean res[];
  String regEx = "[A-Z]{1}[a-z]{1,}[A-Z]{1}[a-z]{1,}_[a-z][a-z0-9]{7}";
  String regEx2 = "[a-z]{1}[a-z]{1,}[a-z]{1}[a-z]{1,}_[a-z][a-z0-9]{7}"
;
  String regEx3 = "[A-Z]{1}[a-z]{1,}[A-Z]{1}[a-z]{1,}_[a-z][a-z0-9]{^7}
";
  Pattern pattern= Pattern.compile(regEx);
  Pattern pattern2= Pattern.compile(regEx2);
  Pattern pattern3= Pattern.compile(regEx3);
  File file = new File(fileName);
  BufferedReader reader = null;
  try {
    reader = new BufferedReader(new FileReader(file));
    String tempString = null;
    int line = 1;
    while ((content = reader.readLine()) != null) {
      try{
        Matcher m = pattern.matcher(content);
        if (m.matches( ))
        {
          res.add(true);
        }
        else
        {
          res.add(false);
          Matcher m2 = pattern.matcher(content);
          if(m2.matches( )){
            throw new WrongInputException1("No capital letters in name"
);
          }
          Matcher m3 = pattern.matcher(content);
          if(m3.matches( )){
            throw new WrongInputException2("the message after the symbo
l _ is either too much or too little");
          }
        }
      }
```

```
      catch(WrongInputException1 wie){
        System.out.println(wie.getMessage());
      }
      catch(WrongInputException2 wie){
        System.out.println(wie.getMessage());
      }

      }
      line++;
    }
    reader.close();
  } catch (IOException e) {
    e.printStackTrace();
  }

  return res
}
```

简答题：

1、 请对比 SVN 和 GIT 在存储文件方面的区别。

GIT 把内容按元数据方式存储，而 SVN 是按文件所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn,.cvs 等的文件夹里。git 对于一个文件的修改存储的是一个快照，就是说针对文件 1,修改之后，生成文件 2,文件 2 中包含文件的 1 的内容，如果当文件 1 不存在，版本回退也就不管用了。SVN 存储的是对文件的差异对比，即是，针对文件 1 进行修改之后，生成一个文件 2 的差异文件，只记录一个差异，不包含之前文件的 1 内容，这个看起来比 Git 高级了一点，最后看到的文件可以理解为是文件 1 和文件 2 merge 之后的结果。对于存储容量，git 相比 svn 占用的容量大，但是 git 一般用来管理代码，现在的磁盘容量也够用，针对不是 IT 人员的版本控制使用，SVN 一般用来存储 word excel pdf 之类的，一定要保证自己的磁盘容量够用。

2、 请分别对比 exception 和 error 的区别，以及 checked exception 和 unchecked exception 的区别。

Exception 和 Error 都是继承了 Throwable 类，在 java 中只有 Throwable 类型的实例才可以被抛出（throw）或者捕获（catch），他是异常处理机制的基本组成类型。Exception 和 Error 体现了 java 平台设计者对不同异常情况的分类，Exception 是程序正常运行中，可以预料的意外情况，可能并且应该被捕获，进行相应的处理。Error 是指正常情况下，不大可能出现的情况，绝大部分的 Error 都会导致程序处于非正常状态，不可恢复状态。既然是非正常情况，所以不便于也不需要捕获，常见的比如 OutOfMemoryError 之类，都是 Error 的子类。

Exception 又分为可检查（checked）异常和不检查（unchecked）异常，可检查异常在源码里必须显示的进行捕获处理，这里是编译期检查的一部分。前面我们介绍的不可查的 Error，是 Throwable 不是 Exception。不检查异常就是所谓的运行时异常，类似 NullPointerException, ArrayIndexOutOfBoundsExceptin 之类，通常是可以编码避免的逻辑错误，具体根据需要来判断是否需要捕获，并不会在编译器强制要求。

3、 判断以下代码是否符合 LSP 原理，如不符合请给出原因。

```
class Rectangle {·
    // invariant h>0 && w>0;
    int h, w;
    Rectangle(int h, int w) {
```

```
this.h=h;
this.w=w;
    }
    // requires factor > 0;
    void scale(int factor) {
        w=w*factor;
        h=h*factor;
    }
}
class Square extends Rectangle {
    // invariant h>0 && w>0;
    //invariant h==w;
    Square(int w) {
        super(w, w);
    }
}
```

符合 LSP 原理，scale 函数不会破坏 RI。

4、 请指出下面给出的类是否是 immutable 的，如果不是请给出具体原因和修改策略。

```
public myString{
public final char a[ ];
myString(char[ ] input){
        a=input;
}
public void length ( ){
     return a.length;
```

}

public myString void getstring ( ) {

return a;

}

public char void getchar (int i)

{

If (i<this.length)

return a[i];

}

}

不是，immutable 类的方法不可以被重写，所以需要在类的前面加上 final。

直接返回会导致表示泄露，所以要进行深拷贝之后再返回

```
public final class myString{
  private final char a[];
  myString(char[] input){
      a=input;
  }
  public void length(){
    return a.length;
  }
  public myString void getstring() {
    char b[];
    System.copyof(a,0,b,0,a.length);
    return b;
  }
  public char void getchar (int i)
  {
    if(i<this.length)
        return a[i];
  }
}
```

5、 观察 Animal、Duck 和 Penguin 类，之后请分别给出后续代码的运行结果。

public Animal{

```
    void fly( ){
    System.out.println(“fly in 100 meters”);
}
void run (int p){
    System.out.println(“run in 100 meters”);
}
}
public Duck extends Animal{
    void fly( ){
    System.out.println(“fly in 200 meters”);
}
void run (string p){
    System.out.println(“run in 200 meters”);
}
}
public Penguin extends Duck{
void run (int p){
    System.out.println(“run in 300 meters”);
}
}
```

```
Animal a = new Duck( );
Duck b=new Duck( );
Animal c=new Penguin( );
a.run( );
b.fly( );
```

c.fly( );

c.run(100);

c.run(“100”);

运行结果如下：

run in 200 meters

fly in 200 meters

fly in 100 meters

run in 300 meters

run in 200 meters