需求分析报告

# 引言

* 编写目的（阐明编写需求分析报告的目的）
* 完成 CS 1B Programming Assignment #7
* 名词解释（列出文档中所用到的专门术语的定义和缩写词的原文。）
* TestCountry: 测试类
* CSVReader:阅读类
* CellularData: 操作类
* SubscriptionYear: 存储年份及特定指令
* Country: 存储国家及国家所有数据
* Programming Style: 在sysllbus 内规定程序规则

# 任务概述

* 目标
* 转换CountryNode 为泛型Node
* 转换CountryList为泛型LinkedList.容器LinkedList存储泛型对象Node Object
* 在LikedList 内实现Iterable接口
* 联系使用iterator和for-each循环遍历list
* 联系组织java文件为包
* 使用Git的特性上传与保存文件库

Node类

变量

* 一个泛型变量data
* 一个参数变量next
* 方法
  + 一个构造方法接入一个泛型对象参数
  + 一个构造方法接入一个泛型对象参数以及参数化Node对象(!instanceof, 测试)
  + 所有参数的Getter.
  + Next的setter
  + 一个toString 返回data对象的String表示(调用data的toString)

LinkedList 类

重命名CountryList

接入Iterable接口

1. 方法与参数
   1. 一个Node对象的集合(用linkedlist)
   2. 一个构造函数用于创建空LinkedList对象(什么都别写)
   3. add(泛型对象参数) 接入泛型对象参数并放到队伍的末尾
   4. contains(泛型对象参数)用于检查是否有同样的对象在linkedlist里
   5. toString()方法返回一个String包含所有Country的所有信息
   6. iterator()返回listiterator对象(要返回正确的对象)
      1. 例如,对于subscriptions来说,
         1. Iterator <SomeType>iterator = subscriptions.iterator();
      * 提示
        + 重写Country与subscriptionYear里的方法
          - 取得当前Node里的信息
        + 检查list是否包含该对象(用equal方法)

ListIterator类

一个在LinkedList里的内部类

* 遍历list里的所有对象
  + 方法与参数
    - 一个泛型current用于追踪当前位置
    - 一个构造方法初始化current使其位于list的开头
    - 一个next()返回下一个list里的对象

Country类

* 在Linkedlist通过测试后修改
* 包含
  + 一个叫Subscriptions的SubscriptionYear的linkedlist(之前为array,同时不允许创建新的linkedlist)
  + 两个int: minYear, maxYear
  + 一个构造方法以”name”为唯一参数
  + 在**getNumSubscriptionsForPeriod()里使用iterator()(Iterator<SubscriptionYear>)去遍历subsriptionYear**
    - **用minYear 与maxYear检查是否超限**
  + ToString()使用for-each遍历linked list
  + Add方法: 检查新的subscription是否需要更新minYear与maxYear

TestGenericList类

* 创建一个泛型linkedlist(Country),测试是否能加入不同长度的country
* 使用**getNumSupscriptionsForPeriod()(实现iterator)**
* Country class使用for-each

~~SubscriptionYear 类:~~

1. ~~变量~~
   1. ~~Int year: 存储数据的年份~~
   2. ~~Double subscription: 存储(国家在year指定)指定年份的数据~~
2. ~~方法~~
   1. ~~SubscriptionYear(int year, double subscription)~~
      1. ~~初始化类并调用方法导入year, subscription~~
   2. ~~Getter/Setter year, subscription~~ 
      1. ~~Year 和 subscription 的输入输出控件~~
   3. ~~Double toString()~~
      1. ~~返回 subscriptions~~

~~Country 类~~

1. ~~变量~~
   1. ~~String name: 国家名~~
   2. ~~SubscriptionYear[] subscriptions: 存储该国家的subscriptionyear 数据~~
2. ~~方法~~
   1. ~~Country(int sizeC)~~
      1. ~~初始化类,创建subscriptions数据并指定长度sizeC~~
   2. ~~addSubscriptionYear(int year, double subscription)~~
      1. ~~创建新的SubscriptionYear对象并存入subscription[]~~
   3. ~~getNumSubscriptionsForPeriod(int startYear, int endYear);~~
      1. ~~丢出startYear 与 endYear 之间的总数~~
         1. ~~丢出 public IllegalArgumentException(String s)~~
            1. ~~当startYear,endYear同时超限或输入顺序相反~~
            2. ~~返回范围内的数据~~
   4. ~~toString()~~
      1. ~~返回 名字 + 所有 数据~~

~~CSVReader类:~~

1. ~~局部变量(private):~~ 
   1. **~~String[] countryNames: 收录所有国家名(from CSV)~~**
   2. ~~int[] yearLabels: 收录所有年份(from CSV)~~
   3. ~~double[][] cellularDataTable: row: country; col: subscriptions for a specific year~~
2. ~~Constructor 方法(file name)~~
   1. ~~创建File fil~~
   2. ~~创建 Scanner sca~~
3. ~~Scanner sca 使用 nextLine()读取CSVfile~~
4. ~~Getter 方法~~
   1. ~~Country names~~
   2. ~~Yearlabels~~
   3. ~~parsedtable~~
5. ~~文件夹 resources~~
   1. ~~存储CSV files~~

~~TestCelluarData 类~~

1. ~~可以直接拷贝数据cellular.csv,初始化~~
   1. ~~final double[] name={ele,ele};~~

* 假定与约束

列出本软件开发工作的假定与约束，例如经费限制、开发期限等等。

1. 遵守大纲
   1. program guideline
   2. 在TestCellular的main()运行时得到正确的结果
   3. 每一个类单独java文件
   4. 会有不同的testfile,只能更改CellularData
   5. 提交时包括以下物品
      1. Sourse code
      2. Java doc(全部 comment, class& methods)
      3. RUN.txt(输出例子,至少两个)
      4. README.txt(简要描述程序)
2. 增加至少两个国家的数据(同时处理至少三个国家)
3. 更改警告提示(如输入年份无效时警告不是-1,而是更friendly的)
4. 不能固定table的行列数,动态创建

# 数数据描述

数据分为静态数据和动态数据。所谓静态数据，指在运行过程中主要作为参考的数据，它们在很长一段时间内不会变化，一般也不会随着运行而改变，所谓动态数据，包括所有在运行中要发生变化的数据，以及在运行中要输入、输出的数据。

* 静态数据（系统运行前已有的数据）

列出所有作为控制或参考用的静态数据，并给出名称。

国家[] , 包括国家名,数据,起始年份

国家数量

起始年份

* 动态数据（系统运行过程中需要的输入数据以及系统运行过程中产生的输出数据）

列出所有动态数据，并给出名称。

求和起始年份

求和终止年份

求和结果

a) String[] countryNames: 收录所有国家名(from CSV)

b) int[] yearLabels: 收录所有年份(from CSV)

c) double[][] cellularDataTable: row: country; col: subscriptions for a specific year

# 功能需求

* 流程图

画出系统的整体流程图。

* 功能划分

对于流程图中的各个功能用树状结构自顶向下进行细化。并对最底层的功能进行编码，给出功能标识符。

* 功能描述

对最底层的功能所要完成的功能进行详细描述，填入下表中：

|  |  |  |
| --- | --- | --- |
| 功能名称 | 功能标识符 | 功能详细描述 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* 数据与功能的对应关系

用一张矩阵图说明功能描述中的各个功能与数据描述中的静态数据、动态数据之间的对应关系，例如：

|  |  |  |
| --- | --- | --- |
| 功能标识符 | 输入 | 输出 |
| 功能标识符1 | 静态数据名称  动态数据名称（例如用户在运行过程中需要用键盘输入数据） | 动态数据名称（例如在运行过程中需要写日志或输出一个报表） |
| 功能标识符2 | 动态数据名称 | 动态数据名称 |
|  |  |  |

# 性能需求

* 时间要求

例如响应时间、更新处理时间、数据转换和传送时间等等。

* 适应性（在操作方式、运行环境、与其他软件的接口等发生变化时，所具有的适应能力。）

# 运行环境描述

* 硬件设备
* 支持软件（操作系统、数据库、其他软件系统如：Lotus Notes等）
* 接口（硬件接口、软件接口）
* 控制（说明控制该软件的运行的方法）
* 用户界面（反映业务流程的用户界面）

# 其他需求

* 如可用性、安全保密、可维护性、可跨平台性等。（分高、中、低定性详细描述）