1. 问题描述

Given an array of citations (each citation is a non-negative integer) of a researcher, write a function to compute the researcher's h-index.

According to the definition of h-index on Wikipedia:

> A scientist has index h if h of his/her N papers have at least h citations each, and the other N – h papers have no more than h citations each.

Example:

```
Input: citations = [3,0,6,1,5]
Output: 3
```

Explanation:

- `[3,0,6,1,5]` means the researcher has `5` papers in total and each of them had received `3, 0, 6, 1, 5` citations respectively.
- Since the researcher has `3` papers with at least `3` citations each and the remaining two with no more than `3` citations each, her h-index is `3`.
- Note: If there are several possible values for `h`, the maximum one is taken as the h-index.

2. 解决思路

先把数组从大到小排序，再从头遍历，找到符合要求的h-index值。遍历时，如果数组中该位置的数值大于等于其序号+1，则h-index至少为序号+1。

例如给定输入为`[3,0,6,1,5]`, 经过排序后可以得到:

```
[3,0,6,1,5] -> [6,5,3,1,0]
```

- 第0位为`6`， `6>=1`，则`h-index=1`；
- 第1位为`5`， `5>=2`，则`h-index=2`；
- 第2位为`3`， `3>=3`，则`h-index=3`；
- 第3位为`1`， `1<4`，则`h-index`不再变化，算法终止。

3. Java 解决方案

1. 在IDE中构建Java Project，命名为HIT_SC_Excise1;
    i. Eclipse 创建项目教程
    ii. IDEA项目创建教程
    iii. VS Code项目创建教程
2. 在 `src` 目录下创建 `HIndex.java`

3.1 基础方案

(1) 从控制台读入用户输入，按照3,0,6,1,5的格式，存储于数组

```
Scanner scanner = new Scanner(System.in);
int[] citations = new int[100];
String[] strs;
System.out.println("Please input the citation numbers:");
String line = scanner.nextLine();
strs = line.split(",");
for (int i = 0; i < strs.length; i++)
    citations[i] = Integer.parseInt(strs[i]);
```

（2）编写排序功能（冒泡排序）

```
int number = citations.length;
for (int i = 0; i < number - 1; i++) {
    for (int j = 0; j < number - i - 1; j++) {
        if (citations[j] < citations[j + 1]) {
            int temp = citations[j + 1];
            citations[j + 1] = citations[j];
            citations[j] = temp;
        }
    }
}
```

(3) 计算 h-index

```
int hindex = 0;
for (int j = 0; j < number; j++) {
    if (citations[j] >= j + 1) {
        hindex = j + 1;
     } else {
        break;
     }
}
System.out.println("The h-index is: " + hindex);
```

(4) 输入不同数组，进行手工测试

(5) 完成编写，git commit到v0.1 All code in main()

3.2 分离函数

(1) 提取出来形成单独的排序函数void sort(int[] array)

```
private static void sort(int[] array) {
        int number = array.length;
        for (int i = 0; i < number - 1; i++) {
            for (int j = 0; j < number - i - 1; j++) {
                if (array[j] < array[j+1] ) {
                    int temp = array[j+1];
                    array[j+1] = array[j];
                    array[j] = temp;
                }
            }
        }
    }
```

(2) 提取出单独的int hindex(int[] citations)函数

```
public static int hindex(int[] citations) {
        // bubble sorting
        sort(citations);

        // calculate h-index
        int hindex = 0;
        for (int i = 0; i < citations.length; i++) {
            if (citations[i] >= i + 1) {
                hindex = i + 1;
            } else {
                break;
            }
        }
        return hindex;
    }
```

(3) git commit到v0.2 separate functions

3.3 从文本文件读取，存储于数组

> Try it by Yourself.

4. 健壮性处理

4.1 空值测试

(1) 输入空值(不进行输入，直接敲击回车) -- 产生异常

```
Please input the citation numbers:

Exception in thread "main" java.lang.NumberFormatException: For input string: ""
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatEx
        at java.base/java.lang.Integer.parseInt(Integer.java:678)
        at java.base/java.lang.Integer.parseInt(Integer.java:786)
        at HIndex.main(HIndex.java:12)

Process finished with exit code 1
```

(2) 编写修复代码

```
String line = new String();
line = scanner.nextLine();
while(line.isEmpty()) {
    System.out.println("Input empty, please re-input:");
    line = scanner.nextLine();
}
```

(3) git commit到v0.3 avoid empty input string

4.2 负值处理

> Try it by Yourself.

> 💡: split之后，调用Integer.parseInt(strs[i])得到整数，检查其是否<0;

4.3 特殊值处理(非整数、非法字符等)

策略1：比较笨的方法：逐个字符位置检查是否为数字（*学生自己实现*）
策略2：与负值一起处理，用正则表达式检查分割后字符串是否匹配"[0-9]+"
策略3：直接调用Integer.parseInt()或者Integer.valueOf(). intValue()，若抛出异常，说明不合法，需要在捕获异常之后让用户重新输入（*学生自己实现*）

(1) 输入特殊字符测试

```
Please input the citation numbers:
3,4,5e3,7
Exception in thread "main" java.lang.NumberFormatException: For input string: "5e3
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatEx
        at java.base/java.lang.Integer.parseInt(Integer.java:668)
        at java.base/java.lang.Integer.parseInt(Integer.java:786)
        at HIndex.main(HIndex.java:18)

Process finished with exit code 1
```

(2) 基于策略2进行修复

```
Scanner scanner = new Scanner(System.in);
int[] citations = new int[100];
String[] strs;
System.out.println("Please input the citation numbers:");

// loop, until user inputs legal string
while (true) {
    String line = scanner.nextLine();
    // if the input is empty.
    while (line.isEmpty()) {
        System.out.println("Input is empty, please re-input:");
        line = scanner.nextLine();
    }

    // check if each part is non-negative integer
    boolean legalNumbers = true;
    strs = line.split(",");
    for (int i = 0; i < strs.length; i++) {
        // if not, stop checking others and let user re-input.
        if (! strs[i].matches("[0-9]+")) {
            System.out.println(strs[i] + " is illegal, please re-input:");
            legalNumbers = false;
            break;
        }
        citations[i] = Integer.parseInt(strs[i]);
    }
    if (legalNumbers) {
        break;
    }
}

int hindex = hindex(citations);

System.out.println("The h-index is: " + hindex);
```

(3) git commit到v0.4 avoid input string containing illegal characters

4.4 更多的输入

(1) 测试超过100个输入 -- 异常抛出

现实中我们无法提前知道用户输入有多少

💡: 使用Java Collections, List, Set, Map, etc

(2) 基于List的实现

```
//int[] citations = new int[100];
List<Integer> citations = new ArrayList<>();

//citations[i] = value;
citations.add(value);

int hindex = hindex(citations);

public static int hindex(List<Integer> citations) {

    citations.sort(Collections.reverseOrder());

    // calculate h-index
    int hindex = 0;
    for (int i = 0; i < citations.size(); i++) {
        if (citations.get(i) >= i + 1) {
            hindex = i + 1;
        } else {
            break;
        }
    }
    return hindex;
}
```

(3) git commit到v0.5 use java collections instead of arrays.

5. 计算功能与用户输入分离

从main中分离出来计算，main只处理调用（作为客户端程序）

5.1 实现方案

(1) 基本实现

```
public class HIndex {

    private List<Integer> citations = new ArrayList<>();

    public HIndex(String input) {
        if(input == null || input.length() == 0)
            throw new IllegalArgumentException("Empty");

        dealInput(input);
    }

    private void dealInput(String input) {

        String[] strs = input.split(",");

        for (int i = 0; i < strs.length; i++) {
            if(! strs[i].matches("[0-9]+"))
                throw new IllegalArgumentException
(strs[i] + " is illegal");

            citations.add(Integer.parseInt(strs[i]));
        }
    }

    public static void main(String[] args) {
        String[] inputs = new String[] {"1,0", "3,-2,4,8"};
        for(int i=0;i<inputs.length;i++) {
            HIndex h = new HIndex(inputs[i]);
            System.out.println(h.calcHIndex());
        }
    }
    public int calcHIndex() {...}
    ...
}
```

(2) git commit到v0.6 separate input with calculation

5.2 静态函数调用实现

(1) 关键片段实现

```
public static int calcHIndex(String input) {
    List<Integer> citations = HIndex.dealInput(input);
    return hindex(citations);
}

public static void main(String[] args) {
    String[] inputs = new String[] {"1,0", "3,-2,4,8"};

    for (String input: inputs) {
        // HIndex h = new HIndex(input);
        System.out.println(HIndex.calcHIndex(input));
    }
}
```

(2) git commit到v0.7 use static methods of a class

6. 直接编写测试用例

打开Junit，演示测试用例如何书写、如何运行、正确和错误的执行结果. (习题课上演示所用版本为Junit 5)

(1) 创建一个新的测试类`HIndexTest.java`, 放在`test/`目录下
(2) `assertEquals`的使用

```
@Test
void testValidInput() {
    assertEquals(1, HIndex.calcHIndex("0,0,1"));
    assertEquals(4, HIndex.calcHIndex("1,4,8,10,20"));
}
```

(3) 抛出异常的类型是否正确测试

```
@Test
void testEmptyInput() {
    assertThrows(IllegalArgumentException.class, () -> {
        HIndex.calcHIndex("");
    });
}
```

(4) 抛出异常的类型及信息是否正确测试

```
@Test
void testEmptyInput2() {
    Exception exception = assertThrows(IllegalArgumentException.class, () -> {
        HIndex.calcHIndex("");
    });

    // Check the exception message
    assertEquals("Empty input", exception.getMessage());
}
```

(5) git commit到v0.8 design test cases manually

7. 根据等价类和边界值思想，设计和编写测试用例

设计更完备的测试用例。

Testing Strategy:

- 按输入的长度进行划分：0,1,2,n(>2)
- 按值划分：正整数或0、负值、小数、包含其他字符
- 按排序后数值大小：第1个<1，第2个<2，第n个<n
- 按相对值大小：数字都一样、一个大其他都小、一个小其他都大

按“笛卡尔积全覆盖”策略设计测试用例或按“每个取值至少覆盖一次”策略

(1) 编写测试用例
(2) git commit到v0.9 design test cases by equivalence partitioning

> 请自行补充更完备的测试用例

8. 改造为OOP

(1) 构造`Paper`类，存储论文的题目、年份、期刊、引用数，封装起来;对`Paper`类进行功能扩展，包括增加引用数、减少引用数

```
public class Paper {
    private String title;
    private int year;
    private String journal;
    private int citation;

    public Paper(String title, int year, String journal, int citation) {
        // Robust?
        this.title = title;
        this.year = year;
        this.journal = journal;
        this.citation = citation;
    }

    public int increaseCitation(int n) {
        if (n <= 0)
            throw new IllegalArgumentException("Please give a positive number!");

        this.citation += n;
        return this.citation;
    }

    public int decreaseCitation(int n) {
        if (this.citation < n)
            throw new IllegalArgumentException("Too large decreasing number!");

        if (n <= 0)
            throw new IllegalArgumentException("Please give a positive number!");

        this.citation -= n;
        return this.citation;
    }

    public String getTitle() {
        return title;
```

```
    }

    public int getCitation() {
        return citation;
    }
}
```

(2) 构建`Author`类

- 基于Paper类改造之前的HIndex类，将其改造为Author类，其中管理作者名字、发表论文清单，使用集合类表达一组Paper而不再用数组。
- 在Author类中增加功能：新增论文、修改某个论文的引用数、计算HIndex、toString（第一行打印出作者，后面每行是一篇论文[题目、引用数]，最后是该作者的HIndex）。
- 写一个客户端（Author类的main函数，模拟使用Paper和Author类）

```
public class Author {
    private String name;
    private List<Paper> papers = new ArrayList<>();

    public Author(String name) {
        this.name = name;
    }

    public void addPaper(Paper p) {
        papers.add(p);
    }

    public void modifyCitation(Paper p, int number, boolean increase) {
        if (increase)
            p.increaseCitation(number);
        else
            p.decreaseCitation(number);
    }

    public int calcHIndex() {
        String input = "";
        for (Paper p : papers) {
            input += p.getCitation() + ",";
        }
```

```
        return HIndex.calcHIndex(input);
    }

    @Override
    public String toString() {
        String content = "Author:\t" + this.name + "\n";
        for (Paper p : papers) {
            content += p.getTitle() + ":\t" + p.getCitation() + "\n";
        }
        content += "HIndex:\t" + calcHIndex();

        return content;
    }

    public static void main(String[] args) {
        ...
    }
}
```

(3) git commit到v1.0 oop

(4) 让Paper类具备在Collections中的排序功能

方案0：自行编写排序功能 (*自行实现*)
方案1：使用Comparator (*自行实现*)
方案2：使用Comparable

```
public class CitationComparator implements Comparator<Paper> {

    @Override
    public int compare(Paper o1, Paper o2) {
        if (o1.getCitation() > o2.getCitation())
            return -1;
        if (o1.getCitation() < o2.getCitation())
            return 1;
        return 0;
    }
}
```

(5) git commit到v1.1 use comparator for sorting papers

9. 重新编写和执行JUnit测试用例

- 对Paper类的各函数进行测试(*自行实现*)
- 对Author类的各函数进行测试(*自行实现*)

10. 构造GUI

为用户开发一个GUI，让用户输入一组论文题目和引用数，计算出HIndex. 使用`JFrame`(或者其他)，创建新GUI类(*自行实现*)

11. 读取文本文件

在GUI上选择一个文件，从文本文件里读取数据，写入Paper对象和Collections。

代码仓库

https://github.com/icecity96/HIT_SC_Excise_1