第4次Java实验

姓名：李哲彦
学号：37220222203675
学院：信息学院
专业：计算机科学与技术
日期：2024年5月19日

实验一 体会基本异常的捕捉

解题思路

按照题目要求，创建一个实验类，在其中创建两个方法，尝试捕捉其异常。

核心代码

对于输入两个整数做除法的方法，主要考虑除数为 0 以及输入的不是整数的问题。所以在 $b == 0$ 时抛出运算异常。同时直接捕捉输入不匹配的异常即可。

```
try{
    a = input.nextInt();
    b = input.nextInt();
    if(b == 0)
        throw new ArithmeticException("除数为0，请重新输入！ ");   // 直接抛出异常
    isValid = true;
}
catch (ArithmeticException e){
    System.out.println(e.getMessage());
    input.next();
}
catch (InputMismatchException e){
    System.out.println("输入不是整数，请重新输入!");
    input.next();
}
```

对于第二个方法，则每次判断数组下标是否越界。如果越界，则立即抛出异常，并发出信息。类似的，如果遇到输入下标不是整数的情况，则同样抛出错误。

```
try{
    index = input.nextInt();
    if(index < 0 || index ≥ 100)
```

```
            throw new ArrayIndexOutOfBoundsException("数组下标越界,请重新输入");
        isValid = true;
    }
    catch (ArrayIndexOutOfBoundsException e){
        System.out.println(e.getMessage());

    }
    catch (InputMismatchException e){
        System.out.println("请输入整数! ");
        input.next();
    }
```

代码调试过程与心得

本项目代码较为简单，无较长调试过程。但是，通过这次实验，我掌握了对异常信息的检测的基本方法。同时，这也让我体会到了异常检测的重要性，为之后几个项目的异常检测打下了基础。

实验二 自定义异常

解题思路

按照题目要求，定义创建一个MyException异常类，继承自RuntimeException。然后再创建两个子异常类，分别用于不同的异常的抛出。

核心代码解析

MyException 直接继承自RuntimeException,作为本项目两个异常的超类。

```
public class MyException extends RuntimeException {
    MyException(String Message){
        super(Message);
    }
}
```

以 MalformedException 为例，继承自 MyException 类，用于检测工资的形式错误。

```
MalformedSalary(String Message) {
    super(Message);
    System.out.println("This is the constructor of MalformedSalary");
}
```

对于邮件的检测，本次使用了正则表达式进行格式检测，快速判断其格式的正确性。
然后，在设置邮箱地址的方法中，调用检测方法判断格式是否正确。

```
public void setEmailAddress(String emailAddress){
    boolean valid = isValidEmail(emailAddress);
    if(!valid)
        throw new MalformedEmailAddress("The format of the address is
invalid!");
    this.emailAddress = emailAddress;
}
```

对于工资的设置，本次主要考虑了如下几个方面

1. 工资不能为负
2. 工资中不能含有多个小数点
3. 开头不能直接是小数点
4. 含有除小数点、数字外的其他字符

在遇到相应错误时，将相应的错误信息写入
综上，该方法写为:

```
public void setSalary(String salary1){
    int num = 0;
    if (salary1.charAt(0) == '.')
            throw new MalformedSalary("The first char cannot be \".\" !");
        for (int i = 0; i < salary1.length(); i++) {
            if (salary1.charAt(i) ≠ '.' &&
!Character.isDigit(salary1.charAt(i)) && salary1.charAt(i) ≠ '-')
                throw new MalformedSalary("The salary format is wrong! Catched
" + salary1.charAt(i));
            if(salary1.charAt(i) == '-')
                throw new MalformedSalary ("The salary cannot be smaller than
zero or the format is wrong as \"-\" is cathed!");
            else if (salary1.charAt(i) == '.') num++;
            if (num ≥ 2) {
                throw new MalformedSalary("There are more than 2 \".\" in the
String! ");
            }
        }
        this.salary = salary1;
    return ;
}
```

对于第一个测试样例，直接传入正确形式，第二个测试样例，输入不同的错误进行检测。

```
User information:
Name: 张三, Age: 25, Salary: 1000.0, EmailAddress: 12345@qq.com
请输入工资:
-1
This is the constructor of MalformedSalary
The salary cannot be smaller than zero or the format is wrong as "-" is found!
请重新输入!
请输入工资:
1..2
This is the constructor of MalformedSalary
There are more than 2 "." in the String!
请重新输入!
请输入工资:
.2
This is the constructor of MalformedSalary
The first char cannot be "." !
请重新输入!
请输入工资:
-1
This is the constructor of MalformedSalary
The salary cannot be smaller than zero or the format is wrong as "-" is found!
请重新输入!
请输入工资:
123
请输入邮箱!
2@@1
This is the constructor of MalformedEmailAddress
The format of the address is invalid!
请重新输入!
请输入邮箱!
123@qq.com
User information:
Name: 小红, Age: 22, Salary: 123.0, EmailAddress: 123@qq.com
```

代码调试过程与心得

通过本次实验我对自定义异常类有了更加清晰的认知，同时也让我意识到了Java中各个异常类的方便之处。灵活的自定义各种异常类，可以让我们非常方便的找到代码错误的地方，及时排查各

类错误。同时，通过抛出错误，也可以及时处理各类特殊病情况，防止程序意外中止，提高程序的安全与稳定性。

实验三 Java版的Grep

实验思路

在文件夹内放两个文本文件 in1.txt 与 in2.txt 进行测试。为防止特殊情况发生，本项目只搜索.txt结尾的文本文件。对于目标文件，使用 BufferedReader 逐行读入，然后使用 PrintWriter 向out.txt中写入结果。

核心代码解析

首先是 AddFile 方法，根据命令行的输入，如果未指定文件名或路径，则在目录下寻找所有对应的txt文件，如果符合要求则加入待搜索队列。相应地，如果找不到任何txt文件，则抛出相应异常。

```
File currentdir = new File("src/main/resources");
boolean txtFound = false;
for(File file : currentdir.listFiles()){ // 直接遍历所有的当前文件夹文件
    if(file.getName().equals("out.txt")){ // 注意跳过 out.txt
System.out.println("Will not search in out.txt");
        continue;
    }
    if(file.getName().endsWith(".txt")){ // 判断是否为 txt 文件
        FiletoSearch.add(file);
        txtFound = true;
    }
}
try{
    if(txtFound == false) // 找不到任何 txt 文件
        throw new FileTypeError("Cannot find a txt file!");
}
catch (FileTypeError e){
    System.out.println(e.getMessage());
}
```

相应地，如果指定了某个文件，则传入的是一个文件名。则不用进行遍历，直接对该文件进行判断即可。

```
File file = new File(Filepath);
try{
    if(!file.getName().endsWith(".txt")){ // 判断是否为 txt 文件
        throw new FileTypeError("The specific file is not a" +
```

```
                " .txt file!");
    }
    else FiletoSearch.add(file);

}
catch (FileTypeError e){ // 不是txt文件
    System.err.println(e.getMessage());
}
```

在逐行读取文件的时候，每行判断其是否包含有目标字符串。如果包含，则输入到相应out.txt文件中。

```
if(str.contains(strToSearch)){
    output.write(file.getName() + ": " + lineNum + str + "\n");    // 输出到
out.txt
    isFound = true;
}
```

同时，在尝试写入 out.txt 时，也要注意判断 out.txt 是否可以正常打开、该文件是否可以正常写入等异常。

```
catch (FileNotFoundException e) {  // 未找到文件，说明 out.txt 可能创建失败
    System.err.println("Cannot find file " + file.getName());
    output.println("Cannot find file " + file.getName());
} catch (IOException e) {   // 无法打开文件
    System.err.println("Cannot open file output.txt");
}
```

在主函数中，我们直接获取命令行传入的参数 args, 然后通过判断其长度来决定是直接遍历整个文件夹还是搜索指定文件即可。

```
try{
    if(args.length == 0){
        throw new ArgsError("Lack enough arguments!");
    }
}
catch (ArgsError e){ // 没有传入参数
    System.out.println(e.getMessage());
    return ;
}
String strToSearch = args[0]; // 要搜索的字符串
if(args.length == 1){ // 没有传入指定文件，搜索所有目录下的文件。
    AddFile(".");
```

```
}
else{
    for(int i = 1; i < args.length; i++)
        AddFile("src/main/resources/" + args[i]);   // 传入所有的
}
```

代码调试过程与心得

最开始，在搜索“默认文件夹”时，我发现其自动定位到了项目的根文件夹，即 "Grep" 文件夹。为了方便，我将默认路径全部改为了项目的 resources 文件夹。
在调试过程中，非常值得注意的是，最开始我的程序正常结束，但是 out.txt 中并没有出现任何文字。经过检查，发现是我在结束写入之后，没有关闭Writer。后经过查阅，只有 output.close()之后才会刷新输出流，否则可能就会出现output.txt 中没有任何内容的情况。今后，在涉及到文件操作时，一定要注意到这一点，避免出现错误。

实验4 网站外链管理系统

解题思路

由于涉及到多线程，本题目的难点就主要在于多线程运行时的线程安全问题。由于每个线程之间共享的是同一个识别用的数组或set, 在判断某个链接是否被重复使用时，不同线程对同一个set的访问就成了应该被注意的问题。本人的解决方法时，所有的线程在搜索完之后，都将新搜索到的链接放入一个线程安全的队列，再由额外的一个线程对其进行判断重复性等，决定是否加入待搜索队列。由于其他线程的访问网页以及解析等操作时间开销较大，故该方法对总时间的影响不大。

核心代码

对于爬虫类，首先我们需要保存一个 $Website$ 类，用于存储主网站的所有信息，如域名、主页等，方便后续判断。在获取网页文件时，使用 jsoup 中的 connect 来 get 其 document（即所有信息），然后解析出其中的所有链接，并将所有内容保存为 html 文本。在访问网站时，注意设计用户画像，防止被网站反爬。

```
Document web = Jsoup.connect(url).userAgent("Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.3")
        .timeout(20000).get(); // 20s内不回复就跳过
Elements allUrls = web.select("a[href]");
ArrayList<Link> urlsList = ExtractallUrls(allUrls);
return new WebPage(url, web.html(), urlsList, web.title());
```

对于解析链接的具体操作：使用 jsoup 的 absUrl 方法来获取所有的 href 类，因为 href 在 html 中代表的就是外链。同时通过判断其中是否含有学校域名来决定是否为校内网站。

```
private ArrayList<Link> ExtractallUrls(Elements allUrls){
    ArrayList <Link> list = new ArrayList<>();
    for(Element element : allUrls){
        String urlStr = element.absUrl("href"); // 提取所有 href        boolean isInternal = urlStr.contains(website.getDomain());// 含有域名则认为是校内网站
        list.add(new Link(urlStr, isInternal));
    }
    return list;
}
```

对于 CrawlScheduler 类，我主要声明了如下几个对象，作用如注释：

```
public CrawlerScheduler(Website website, int numThreads, int pageCapacity) {
    this.website = website;      // 学校网站
    this.numThreads = numThreads;    // 线程数量
    this.latch = new CountDownLatch(numThreads);  // 统计还有多少个线程在运行
    this.visitedUrls = Collections.synchronizedSet(new HashSet<>());  // 线程安全的set
    this.urlQueue = new LinkedBlockingQueue<>();  // 线程安全的队列
    this.executorService = Executors.newFixedThreadPool(numThreads + 1); // 创建线程池,额外加上一个线程  用于更新待搜索链接
    this.urltoUpdate = new LinkedBlockingQueue<>();  // 待判断的链接
    this.running = true;    // 运行标志
    this.isInternalnum = 0;  // 校内网站数量
    this.isNotIntenalnum = 0;    // 校外网站数量
    this.Allpage = new LinkedBlockingQueue<>(pageCapacity + 10);  //  结果存储
    this.totNum = pageCapacity;  // 最大搜索多少个网站
}
```

对于搜索线程的具体内容：首先，为了随时能够及时终止线程，且防止网站被搜索完成后，该线程被阻塞，以及由于访问速度较慢导致线程阻塞或终止，在从搜索队列中获取线程时，使用非阻塞的 poll 方法，当连续多次未读取到新链接时，才可以视为没有新链接可以访问，从而退出循环，终止线程。

```
nowUrl = urlQueue.poll();
if (nowUrl == null) {
    if (!running) break;
    failTimes++;
    Thread.sleep(100);
    if (failTimes > 100) break; // 超过10秒都没有新网页，判断为搜索完成
    continue;
}
```

之后，对于每个线程，创建一个 Crawler 类，爬取对应的链接。

```
Crawler crawler = new Crawler(website);
WebPage page = crawler.startCrawl(nowUrl);
```

对于搜索线程的更新操作，每次将搜索到的线程都塞入到唯一的线程安全的队列中，防止出现错误。

```
private void UpdateUrls(ArrayList<Link> nowList) throws InterruptedException {
    for (Link link : nowList) {
        urltoUpdate.add(link);
    }
    return ;
}
```

对于更新线程，每次都尝试从待更新队列中读取待判断的链接。类似的，如果长时间没有读取到新的待处理链接，则认为已经爬取完成，退出该线程。

```
public void run() {
    int failTimes = 0;
    while (running) { // 只要没有被打断
        try {
            nowUrl = urlQueue.poll();  // 尝试取出一个搜索链接
            if (nowUrl == null) {
                if (!running) break;
                failTimes++;
                Thread.sleep(100);
                if (failTimes > 100) break; // 超过10秒都没有新网页，判断为搜索完成
                continue;
            }
            failTimes = 0;
            if (!nowUrl.contains("http")) continue; // 不是http协议的链接不进去
            Crawler crawler = new Crawler(website);
            WebPage page = crawler.startCrawl(nowUrl);
            if (page == null) continue;
            if (getAllpage().size() ≥ totNum){ // 丢入新网页时进行判断
                running = false;
                break;
            }
            Allpage.add(page); // 尝试塞入
            ArrayList<Link> nowList = page.getLinks();
            UpdateUrls(nowList); // 注意更新链接被访问情况
        } catch (InterruptedException e) {
            System.out.println("Thread is interrupted!");
```

```
            } catch (IOException e) {
                System.out.println(e.toString());
            }
        }
        latch.countDown(); // 一个线程结束
    }
```

对于更新带搜索链接的线程，每次塞入时进行统计，查看搜索到了多少个内链和外链。由于该线程只有一个，不需要使用线程锁等操作来保证统计结果的线程安全性。

```
if(!visitedUrls.contains(nowUrl.getUrl()) && nowUrl.isInternal()){  // 如果没有访问过，且是校内网站，则添加到待搜索队列中
    visitedUrls.add(nowUrl.getUrl());
    urlQueue.add(nowUrl.getUrl());
    isInternalnum++;
}
else if(!visitedUrls.contains(nowUrl.getUrl()) && !nowUrl.isInternal())  // 否则仅进行统计
    isNotIntenalnum++;
```

对于用户界面，题目要求我们展示所有网页的信息，这显然是非常不雅观且混乱的。因为根据爬取结果来看，学校网页至少多达几万个，全部展示出来显然会导致界面混乱。故我将用户界面设置为，只有用户输入1，才会展示下一个页面。此处为了节省内存，只存储了前20个页面。可以根据需要随意调整。

```
while(true){
    int x = input.nextInt();
    if(x == 1){
        WebPage page = allPage.poll();
        System.out.println(page.showCompleteInfo());
        countNUm++;
        if(countNUm ≥ pageCapacity){
            System.out.println("All the websites have been showed!");
            break;
        }
    }
    else if(x == 2){
        break;
    }
}
```

在搜索过程中，程序可以显示有哪些链接，虽然是本校域名，但访问错误，例如：

```
Unknown host: https://zchqc2022.xmu.edu.cn
HTTP error:
http://alumni.xmu.edu.cn/xyzh2023/__local/7/5A/95/5CAB7787FE1E2EABF323206D9D5_
55AE2871_644E58.pdf?e=.pdf - 404
Unsupported type for URL:
http://alumni.xmu.edu.cn/xyzh2023/shadaxiaoyoutongxun63qi.pdf -
application/pdf
Unsupported type for URL: http://alumni.xmu.edu.cn/xyzh2023/61.pdf -
application/pdf
```

等 404 错误，或者为 pdf 文件等不支持的格式。

最终得到的输出与统计结果如下（部分）

```
Input the number of Website to be showed:
20
Please wait.......
IsInternal Link num: 238, NotInternal Link num: 62
Press 1 to show the next Webpage, 2 to exit
1
Title: 厦门大学信息学院 url: https://informatics.xmu.edu.cn/
isInternal Num: 7 isNotinternal Num: 7
suburl 0 : https://mp.weixin.qq.com/s/SPNGQW3A5uY8Ha5hlDhZHA is Internal:
false
suburl 1 : https://informatics.xmu.edu.cn/article_list.jsp?
urltype=tree.TreeTempUrl&wbtreeid=1061 is Internal: true
suburl 2 : http://172.27.65.113/ is Internal: false
suburl 3 : https://cmsadmin.xmu.edu.cn/ is Internal: true
suburl 4 : https://informatics.xmu.edu.cn/index.htm is Internal: true
suburl 5 : https://informatics.xmu.edu.cn/introduction/xyjj.htm is Internal:
true
suburl 6 : https://informatics.xmu.edu.cn/info/1036/35599.htm is Internal:
true
suburl 7 : https://informatics.xmu.edu.cn/introduction/xyjj.htm is Internal:
true
suburl 8 : https://informatics.xmu.edu.cn/introduction/xyld.htm is Internal:
true
suburl 9 : https://informatics.xmu.edu.cn/introduction/zzjg.htm is Internal:
true
suburl 10 : https://informatics.xmu.edu.cn/introduction/lxwm.htm is Internal:
```

```
true
suburl 11 : https://informatics.xmu.edu.cn/djsz1.htm is Internal: true
suburl 12 : https://informatics.xmu.edu.cn/djsz1/djxxjy.htm is Internal: true
suburl 13 : https://informatics.xmu.edu.cn/djsz1/llxx.htm is Internal: true
suburl 14 : https://informatics.xmu.edu.cn/djsz1/ztjy.htm is Internal: true
Only shows the first 15 suburls!
HTML CONTENT
<!doctype html>
<html lang="zh-cn">
 <head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>厦门大学信息学院</title>
  <meta name="pageType" content="1">
  <meta name="pageTitle" content="厦门大学信息学院">
  <meta name="keywords" content="信息学院，智能系，计算机系，
.............

1
Title: 组织机构-厦门大学信息学院 url:
https://informatics.xmu.edu.cn/introduction/zzjg.htm
isInternal Num: 4 isNotinternal Num: 4
suburl 0 : https://informatics.xmu.edu.cn/education/gjjlxm.htm is Internal:
true
suburl 1 : http://172.27.65.113/ is Internal: false
suburl 2 : https://informatics.xmu.edu.cn/article_list.jsp?
urltype=tree.TreeTempUrl&wbtreeid=1061 is Internal: true
suburl 3 : https://cmsadmin.xmu.edu.cn/ is Internal: true
suburl 4 : https://weibo.com/u/3176355500 is Internal: false
suburl 5 : https://informatics.xmu.edu.cn/introduction/zzjg.htm# is Internal:
true
suburl 6 : https://informatics.xmu.edu.cn/introduction/zzjg.htm# is Internal:
true
suburl 7 : https://informatics.xmu.edu.cn/index.htm is Internal: true
suburl 8 : https://informatics.xmu.edu.cn/introduction/xyjj.htm is Internal:
true
suburl 9 : https://informatics.xmu.edu.cn/info/1036/35599.htm is Internal:
true
suburl 10 : https://informatics.xmu.edu.cn/introduction/xyjj.htm is Internal:
true
```

```
suburl 11 : https://informatics.xmu.edu.cn/introduction/xyld.htm is Internal:
true
suburl 12 : https://informatics.xmu.edu.cn/introduction/zzjg.htm is Internal:
true
suburl 13 : https://informatics.xmu.edu.cn/introduction/lxwm.htm is Internal:
true
suburl 14 : https://informatics.xmu.edu.cn/djsz1.htm is Internal: true
Only shows the first 15 suburls!
HTML CONTENT
<!doctype html>
<html lang="zh-cn">
 <head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>组织机构-厦门大学信息学院</title>
  <meta name="pageType" content="2">
  <meta name="pageTitle" content="组织机构">
  <meta name="keywords" content="信息学院, 智能系, 计算机系,
............

1
Title:  url: https://cmsadmin.xmu.edu.cn/
isInternal Num: 0 isNotinternal Num: 0
No suburl!
HTML CONTENT
<html>
 <head>
  <script>
    location.href="/system/login.jsp"
</script>
 </head>
 <body></body>
</html>
1
Title: 联系我们-厦门大学信息学院 url:
https://informatics.xmu.edu.cn/introduction/lxwm.htm
isInternal Num: 4 isNotinternal Num: 4
suburl 0 : https://informatics.xmu.edu.cn/education/gjjlxm.htm is Internal:
true
suburl 1 : http://172.27.65.113/ is Internal: false
```

```
suburl 2 : https://informatics.xmu.edu.cn/article_list.jsp?
urltype=tree.TreeTempUrl&wbtreeid=1061 is Internal: true
suburl 3 : https://cmsadmin.xmu.edu.cn/ is Internal: true
suburl 4 : https://weibo.com/u/3176355500 is Internal: false
suburl 5 : https://informatics.xmu.edu.cn/introduction/lxwm.htm# is Internal:
true
suburl 6 : https://informatics.xmu.edu.cn/introduction/lxwm.htm# is Internal:
true
suburl 7 : https://informatics.xmu.edu.cn/index.htm is Internal: true
suburl 8 : https://informatics.xmu.edu.cn/introduction/xyjj.htm is Internal:
true
suburl 9 : https://informatics.xmu.edu.cn/info/1036/35599.htm is Internal:
true
suburl 10 : https://informatics.xmu.edu.cn/introduction/xyjj.htm is Internal:
true
suburl 11 : https://informatics.xmu.edu.cn/introduction/xyld.htm is Internal:
true
suburl 12 : https://informatics.xmu.edu.cn/introduction/zzjg.htm is Internal:
true
suburl 13 : https://informatics.xmu.edu.cn/introduction/lxwm.htm is Internal:
true
suburl 14 : https://informatics.xmu.edu.cn/djsz1.htm is Internal: true
Only shows the first 15 suburls!
HTML CONTENT
<!doctype html>
<html lang="zh-cn">
 <head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>联系我们-厦门大学信息学院</title>
  <meta name="pageType" content="2">
  <meta name="pageTitle" content="联系我们">
  <meta name="keywords" content="信息学院, 智能系, 计算机系,
.............
```

调试过程与心得

最开始，我尝试使用自带线程锁的set类来存储所有已访问的网页链接，但是发现，这依然会引起线程冲突。后进过查阅，发现可能的原因是其保证了写入的安全，但是不能保证每个线程读取到

的set是最新的。最后，我选择了使用一个额外的线程，使用队列统一处理所有的待判断链接，以此来保护线程的安全性。

实验总体心得

通过这次实验，我一方面了解了Java抛出异常的处理，理解了异常处理的方便与重要性，利用好异常，其将会是一个非常强大的处理错误的方法。对于多线程操作，通过本次实验，我对多线程的熟练度上升到了一个新的高度。多线程可以极大的提升软件的性能和效率，日后我还有继续学习更多有关多线程的知识。