

## 作业一

1. 声明四个变量用于记录大写字母，小写字母，数字和其他字符的个数。

```
int lowerCaseCounter = 0;
int upperCaseCounter = 0;
int specialCharacterCounter = 0;
int numberCounter = 0;
```

2. 利用 `for` 遍历字符串的每个字符。

- 利用ASCII的值来区分大写字母，小写字母，数字和其他字符

```
if(str.charAt(i) >= 65 && str.charAt(i) <= 90){
    upperCaseCounter++;
}else if(str.charAt(i) >= 97 && str.charAt(i) <= 122){
    lowerCaseCounter++;
}else if(str.charAt(i) >= 48 && str.charAt(i) <= 57){
    numberCounter++;
}else{
    specialCharacterCounter++;
}
```

3. 打印结果

```
System.out.println("大写字母的个数是" + upperCaseCounter);
System.out.println("小写字母的个数是" + lowerCaseCounter);
System.out.println("数字的个数是" + numberCounter);
System.out.println("其他字符的个数是" + specialCharacterCounter);
```

4. 运行结构截图

```
/Library/Java/JavaVirtualMachines/jdk-11.0.8.jdk/Contents/Home/
大写字母的个数是4
小写字母的个数是2
数字的个数是4
其他字符的个数是5

Process finished with exit code 0
```

## 作业二

1. 创建一个方法 `largestCommonString` 来获取最大相同子串

```
public static String largestCommonString(String str1, String str2){  
  
}
```

- 声明三个变量分别用于记录最大相同子串的左右下标和长度

```
int left = 0;  
int right = 0;  
int max = Integer.MIN_VALUE;
```

- 进行极端值的判断

```
if(str1.equals("") || str2.equals("")){  
    return "";  
}
```

- 运算时我们始终保持str1是较短的字符串

```
if(str1.length() - str2.length() >= 0){  
    String temp = str1;  
    str1 = str2;  
    str2 = temp;  
}
```

- 利用双重循环不断缩小搜索范围，并更新left, right, max的值。j的值从i+1开始，因为substring()方法的取值是左闭右开。

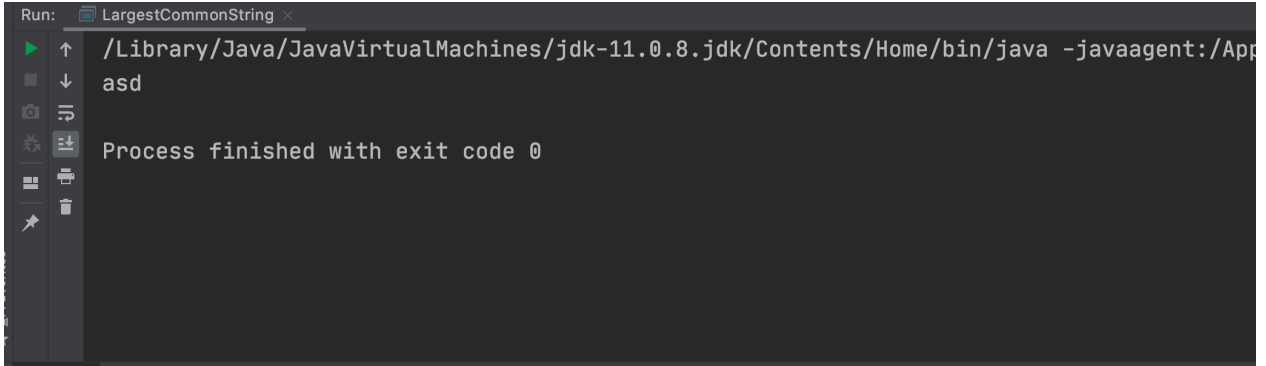
```
for(int i = 0; i < str1.length() - 1; i++){  
    for(int j = i+1; j < str1.length(); j++){  
        if(str2.contains(str1.substring(i,j))){  
            if(j - i >= max){  
                max = j - i;  
                left = i;  
                right = j;  
            }  
        }  
    }  
}
```

- 最后返回子串

2. 在 `main` 方法中调用 `largestCommonString` 方法，打印结果。

```
public static void main(String[] args){
    String largestCommonString = largestCommonString("asdafghjka","aaasdfg");
    System.out.println(largestCommonString);
}
```

3. 运行结构截图



## 作业三

1. 准备一个集合，并声明一个变量保存 "123,456,789,123,456"

```
Map<String, Integer> stringCounter = new HashMap<>();
String string = "123,456,789,123,456";
```

2. 利用 `split` 方法把字符串以 `,` 为分割线进行划分，并用数组保存

```
String[] arr = string.split(",");
```

3. 利用 `forEach` 来遍历数组，把得到的结果存在集合中

```
for (String s : arr) {
}
```

- 判断是否存在，如果存在，值加一

```
if (stringCounter.containsKey(s)) {
    stringCounter.put(s, stringCounter.get(s) + 1);
}
```

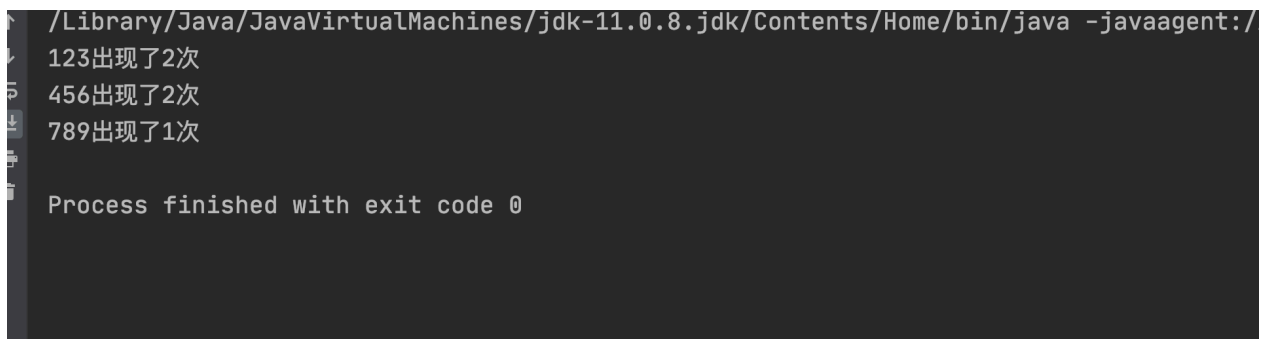
- 如果不存在，直接放入集合中

```
else {  
    stringCounter.put(s, 1);  
}
```

#### 4. 打印结果

```
Set<Map.Entry<String, Integer>> entries = stringCounter.entrySet();  
for(Map.Entry<String, Integer> entry: entries){  
    System.out.println(entry.getKey()+"出现了"+entry.getValue()+"次");  
}
```

#### 5. 运行结果截图



```
/Library/Java/JavaVirtualMachines/jdk-11.0.8.jdk/Contents/Home/bin/java -javaagent:/  
123出现了2次  
456出现了2次  
789出现了1次  
  
Process finished with exit code 0
```

## 作业四

#### 1. 创建一个学生类

- 创建私有变量，构造方法，get和set方法。
- 重写 equals 和 hashCode 方法，用于以后学生的比较。这里只比较学号，即两个同学的学号相等，这两个同学就是一个同学。

```
@Override  
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass()) return false;  
    Student student = (Student) o;  
    return id == student.id;  
}  
  
@Override  
public int hashCode() {  
    return Objects.hash(id);  
}
```

- 重写 `toString` 方法，便于打印。

## 2. 创建一个信息管理类，包含信息管理系统的主要方法

- 声明一个集合用于存放学生信息

```
List<Student> studentList = new ArrayList<>();
```

- 创建添加学生的方法，添加成功返回true，否则返回false

```
public boolean addStudent(Student student) {  
  
}
```

- 首先利用 `contains` 方法判断学生是否存在，如果学号一样，说明学生存在，直接返回false

```
if (studentList.contains(student)) {  
    return false;  
}
```

- 如果学生不存在，再进行添加

```
else {  
    studentList.add(student);  
    return true;  
}
```

- 创建删除学生的方法，删除成功返回true，否则返回false

```
public boolean removeStudent(Student student) {  
  
}
```

- 首先利用 `contains` 方法判断学生是否存在，如果存在，删除学生，并返回true

```
if (studentList.contains(student)) {  
    studentList.remove(student);  
    return true;  
}
```

- 否则直接返回false

```
else{
    return false;
}
```

- 创建查找学生的方法，这里我们根据学号进行查找，如果查找成功，返回学生信息，否则返回null

```
public Student findStudentById(int id){

}
```

- 利用 `foreach` 来查找

```
for(Student student : studentList){
    if(student.getId() == id){
        return student;
    }
}
```

- 否则返回null

- 创建修改学生的方法，修改成功返回true，否则返回false。

```
public boolean modifyStudent(int id, String newName, int newAge){

}
```

- 首先根据 `findStudentById` 方法查找学生
- 再查找结果进行判断，如果结果不等于null，说明学生存在。我们可以利用set方法来修改信息，并返回true

```
if(student != null){
    student.setName(newName);
    student.setAge(newAge);
    return true;
}
```

- 否则直接返回false

- 创建打印所有学生的方法

```
public void showAllStudents(){

}
```

- 先判断集合是否为空。如果是空，提示用户输入学生信息

```
if(studentList.isEmpty()){  
    System.out.println("学生档案为空，请添加学生信息！");  
}
```

- 如果集合不是空，利用 `foreach` 遍历打印

```
for(Student student : studentList){  
    System.out.println(student.toString());  
}
```

### 3. 创建一个测试类，主要的逻辑写在了 `workflow` 的方法里，这里主要对此方法进行讲解

- 声明一些变量和提示信息

```
System.out.println("-----");  
System.out.println("          学生信息管理系统          ");  
System.out.println("-----");  
ManageSystem ms = new ManageSystem();  
Scanner sc = new Scanner(System.in);  
boolean flag = true;
```

- 把主要逻辑写在 `while` 循环里

```
while(flag){  
  
}
```

- 打印操作信息，提示用户选择

```
System.out.println("1-增加学生信息");  
System.out.println("2-删除学生信息");  
System.out.println("3-修改学生信息");  
System.out.println("4-查找学生信息");  
System.out.println("5-打印所有学生信息");  
System.out.println("0-退出程序");  
System.out.println("请管理员输入操作代号0-5：");
```

```
/Library/Java/JavaVirtualMachines/jdk-11.0.8.jdk
```

## 学生信息管理系统

```
1-增加学生信息
2-删除学生信息
3-修改学生信息
4-查找学生信息
5-打印所有学生信息
0-退出程序
请管理员输入操作代号0-5:
```

- 通过 `switch` 来执行不同的功能

```
int order = sc.nextInt();
switch (order){

}
```

- 1-增加学生信息。提示用户输入学生的学号，姓名和年龄。调用 `addStudent` 方法进行添加。根据结果，显示不同的提示信息。这里注意学号不能重复添加。

请管理员输入操作代号0-5:

1

请分别输入学生的学号，姓名和年龄：

1

傅梓宸

28

学生添加成功！



请管理员输入操作代号0-5:

1

请分别输入学生的学号，姓名和年龄:

1

林旖曦

29

学生添加失败！学号不能重复，请重新添加！

- 2-删除学生信息。提示用户输入想要删除学生的学号。先调用 `findStudentById` 方法，再调用 `removeStudent` 方法。根据结果，显示不同的提示信息。

请管理员输入操作代号0-5:

2

请输入想要删除学生的学号:

1

学生删除成功！

请管理员输入操作代号0-5:

2

请输入想要删除学生的学号:

2

学生删除失败！学号不存在，请重新查看！

- 3-修改学生信息。先调用 `findStudentById` 方法，判断学生是否存在。如果存在，提示用户输入想要修改的信息，否则报错。

请管理员输入操作代号0-5:

3

请输入想要修改学生的学号:

1

修改后的名字:

林旖曦

修改后的年龄:

29

学生修改成功!

-----

请管理员输入操作代号0-5:

3|

请输入想要修改学生的学号:

2

学生修改失败, 请检查学号!

-----

- 4-查找学生信息。

请管理员输入操作代号0-5:

4

请输入想要查看学生的学号:

1

id=1, name='林旖曦', age=29

-----

请管理员输入操作代号0-5:

4

请输入想要查看学生的学号:

3

该学号不存在, 请重新查看!

- 5-打印所有学生信息。

请管理员输入操作代号0-5:

5

id=1, name='林旖曦', age=29

id=2, name='傅梓宸', age=28

请管理员输入操作代号0-5:

5

学生档案为空, 请添加学生信息!

- 0-退出程序。flag设为false。结束 while 循环

请管理员输入操作代号0-5:

0

程序已退出!

## 作业五

1. 创建一个卡类, 里面包含作业要求的功能。
  - 声明两个常量用于存放花色和大小

```
private static final String[] values = {"大王", "小王", "2", "A", "K", "Q",  
    "J", "10", "9", "8", "7", "6", "5", "4", "3"};  
private static final String[] suits = {"红桃", "黑桃", "梅花", "方块"};
```

- 声明一些变量用于存放扑克牌信息，顺序信息，三个玩家手牌信息和底牌信息。

```
private Map<Integer, String> cards = new HashMap<>();
private List<Integer> order = new ArrayList<>();
private List<Integer> player1 = new ArrayList<>();
private List<Integer> player2 = new ArrayList<>();
private List<Integer> player3 = new ArrayList<>();
private List<Integer> holeCards = new ArrayList<>();
```

- 创建 `generateCards` 方法用于生成扑克牌

```
private void generateCards(){

}
```

- 首先把大王和小王存入cards中，并在order中记录它们的键

```
cards.put(1, values[0]);
cards.put(2, values[1]);
order.add(1);
order.add(2);
```

- 利用双重循环把其他的卡牌信息分别存入cards和order中。这里注意一定要先遍历大小，再遍历花色。否则后期排序时会出现错误。

```
for(int i = 2; i < values.length; i++){
    for (String suit : suits) {
        cards.put(cardKey, suit + values[i]);
        order.add(cardKey);
        cardKey++;
    }
}
```

- 最后在将牌打乱

```
Collections.shuffle(order);
```

- 创建 `dealCards` 方法用于发牌

```
private void dealCards(){

}
```

- 利用 `for` 循环遍历 `order`

```
for(int i = 0; i < order.size(); i++){  
  
}
```

- 先发三张底牌

```
if(i < 3){  
    holeCards.add(cardKey);  
}
```

- 再依次给三个玩家发牌

```
else if(i % 3 == 1){  
    player1.add(cardKey);  
}else if(i % 3 == 2){  
    player2.add(cardKey);  
}else{  
    player3.add(cardKey);  
}
```

- 创建 `printCards` 方法打印玩家的牌

```
private void printCards(List<Integer> player, Map<Integer, String> cards){  
    for(Integer cardKey : player){  
        System.out.print(cards.get(cardKey)+" ");  
    }  
    System.out.println();  
}
```

- 创建 `showCardsInfo` 方法打印结果

```
public void showCardsInfo(){  
    generateCards();  
    dealCards();  
    Collections.sort(player1);  
    Collections.sort(player2);  
    Collections.sort(player3);  
    System.out.println("玩家一");  
    printCards(player1, cards);  
    System.out.println("玩家二");  
    printCards(player2, cards);  
}
```

```
System.out.println("玩家三");  
printCards(player3, cards);  
System.out.println("底牌");  
printCards(holeCards, cards);  
}
```

## 2. 创建一个测试类测试结果

```
public static void main(String[] args){  
    Cards cardsTest = new Cards();  
    cardsTest.showCardsInfo();  
}
```

## 3. 运行结果截图

```
玩家一  
红桃2 黑桃2 方块2 红桃A 黑桃K 梅花K 方块K 梅花J 黑桃9 方块8 红桃7 方块7 梅花6 方块6 红桃5 梅花5 红桃3  
玩家二  
小王 梅花2 黑桃A 黑桃Q 梅花Q 黑桃J 黑桃10 红桃9 方块9 红桃8 黑桃8 梅花7 黑桃6 红桃4 黑桃4 黑桃3 梅花3  
玩家三  
大王 方块A 红桃K 红桃Q 方块Q 红桃J 方块J 梅花10 方块10 梅花9 黑桃7 红桃6 黑桃5 方块5 梅花4 方块4 方块3  
底牌  
梅花8 红桃10 梅花A
```

**END**