类设计说明

功能性类

argParser

- 设计目的：为了实现对命令行传参的解析，并通过 `get` 方法来获取对应的命令行参数细节，如对 `--interactive true` 的命令行参数，可以通过 `ap.get("interactive ")` 来获得字符串 `true`
- 为了实现上述目标，需要明确可以支持的参数类型，目前仅支持
  - int
  - double
  - string
  - boolean

  对这些支持的，采用了枚举对象 `argType` 来方便管理
- 对于每个命令行参数，使用了一个类 `Args` 来做封装，具体的，类 `Args` 可以支持
  - 设置不同的类种类，即包含了一个 `argType` 的枚举变量
  - 对应的触发词，例如 `--interactive true` 的触发词为 `interactive`
  - 一个描述语句，便于在必要的时候进行打印
  - 一个可能的默认值，便于在没有命令行直接提供的时候能有一个默认值返回
  - 一个标识符，表示当前是否提供默认值
  - 具体的命令行变量，例如上述例子，则应该是 `"true"`
  - 对具体的变量检查有效性，在源代码中，采用了正则表达式来做检验
  - 获取具体的变量，并作相应的检查；例如如果支持默认变量且未从命令行获取，则返回默认值
- 具体的来说，使用这个类的方法为
  - 在 `main` 函数中新建一个 `argParser` 对象
  - 通过 `argParser.addArgs()` 方法添加命令行参数，这个方法是做了重载的，可以支持是否填写描述字符串以及是否设定默认值，但是必须给定的是触发词和类型
  - 通过 `argParser.parseArgs(args)` 的方法，将命令行参数进行解析，其中 `args` 为 `main` 函数从命令行接受的参数列(`String[] args`)
  - 通过 `argParser.get()` 方法来获取命令行值，其会直接调用对应触发词的 `get` 方法，因而也能够在需要时获取默认值

range

`range` 类所作的是对一个用来表示范围的二元组的抽象，其为 `Comparable` 的一个实现，这样做的目的是实现排序，后面展示的时候可以以一个更加有序的方式进行展示

在 Java 中，类比较是通过重载一个函数 `compareTo` 来实现的，比较方法参考了二元组的比较方法，先比较第一个，相同再比较第二个

同时还需要一个判断两个范围是否有重合，使用了一个 `isOverlap` 的方法，传入的参数为另一个范围类，返回两者是否有重合

此外还重载了一个方法 `toString()` 来将范围用一个更加简便的方式来完成

tax

`tax` 类是继承自 `range` 类的，用来描述某个特定的规则，由于类的继承，自然的，它也有了可比较的属性，同时还添加了一个用于表示范围的浮点数，用于表示具体的在这个范围的税率

基于这样的设计思路，这个类需要实现

- 构造时通过基类的构造，将 `range` 设置正确，同时设置当前的税率
- 根据传入的薪资，返回在当前条目下应该支付的税额(见 `getTax` 方法)

shell

`shell` 类所做的为整个实验的核心功能，包括计算逻辑的核心，如何处理用户输入，检查输入的合法性，根据用户输入给出相对应的提示，计算输出等

因而所需要做的有：

- 检索用户输入的命令，并根据用户语义来完成不同的功能
- 打印命令帮助信息，方便用户了解如何进行交互
- 实现具体的计算模块，考虑到 `tax` 类是针对每个规则做的，从而这个 `shell` 类需要维护一个 `tax` 的动态列表，之后进行遍历计算即可
- 针对不同需求进行打印计算过程及结果

TaxCalculator

该类主要作用是创建 `main` 函数，并未有特别的设计

错误类

错误类主要根据不同的场景，设计的不同的错误类，其设计大都是如下的方式

```
class someTypeException extends Exception {
    public someTypeException(String info) {
        super(info);
    }
}
```

这样设计的好处在于，可以较为自由的将错误信息用格式化字符串的方式发送至上层，上层可以通过调用 `e.getMessage()` 的方式来获取所设置的错误信息，可以将其直接打印到控制台，方便用户了解到具体的错误细节；同时封装了不同的错误类，方便 `try catch` 的时候针对性的处理

具体设计了如下的错误类

- `ArgException` : 用于处理在解析命令行参数时发生的错误，如类型匹配错误，缺少参数错误
- `RangeException` : 用于处理将两个整数解析为一个范围时发生的错误，例如 high > low 这样的错误

- `RuleException`：用于处理解析规则时出现的错误，例如文件出错等