

实验 5 泛型与集合框架

实验目的

1. 掌握泛型中参数类型的声明和使用
2. 掌握 List 实现类的主要操作
3. 掌握 Map 实现类的主要操作

实验步骤

- 步骤 1:** 创建工程，工程名以学号姓名方式命名，“2020010505-薛飞宇”。
- 步骤 2:** 为每道实习题目创建对应的包，如“work1”、“work2”……；
- 步骤 3:** 按照要求创建源代码，进行编写，注意编码格式，如缩进、命名规范等；
- 步骤 4:** 规范书写实习报告；
- 步骤 5:** 实现与测试。附上代码或以附件的形式提交，同时贴上必要的代码运行截图；
- 步骤 6:** 及时总结心得体会与备忘。

实验内容

实验题 1 计算柱体的体积

根据题目要求，先设计抽象类 Shape，其中仅有一个抽象方法 **abstract double getarea()**。然后在 Circle 类和 Rectangle 类中分别实现抽象类中的抽象方法，如下图。

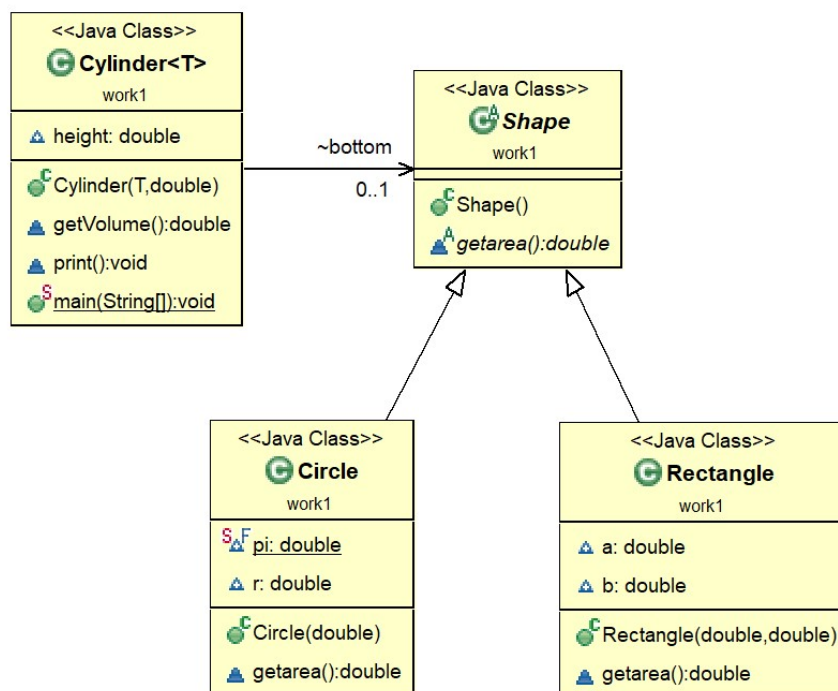
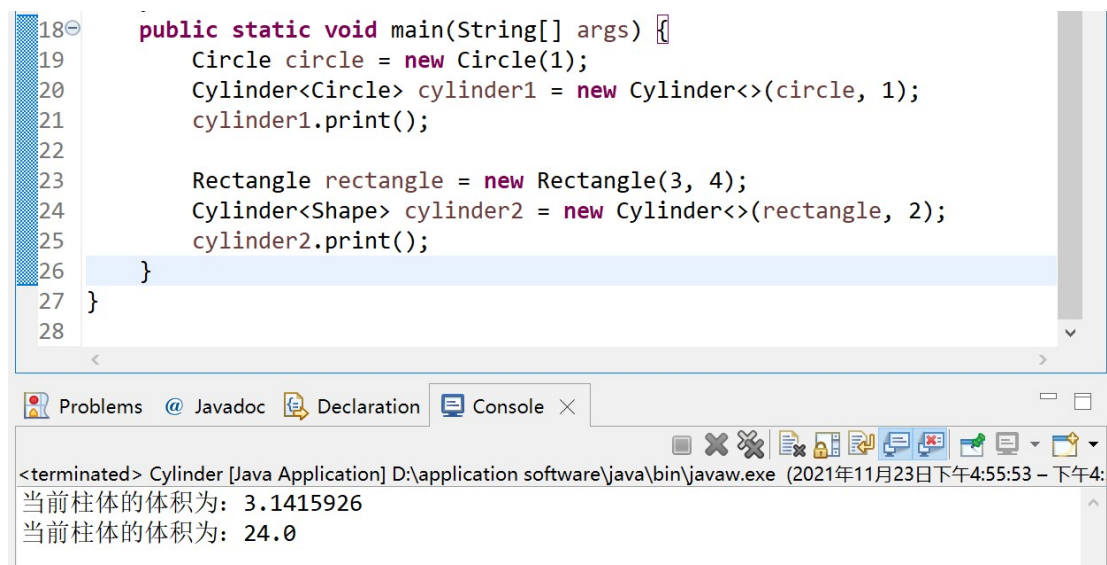


图 1.1 实验 1UML 类图

最后，在 `Cylinder` 类中设计相关的泛型即可。本题十分简单，看实验结果就好。



```
18 public static void main(String[] args) {
19     Circle circle = new Circle(1);
20     Cylinder<Circle> cylinder1 = new Cylinder<>(circle, 1);
21     cylinder1.print();
22
23     Rectangle rectangle = new Rectangle(3, 4);
24     Cylinder<Shape> cylinder2 = new Cylinder<>(rectangle, 2);
25     cylinder2.print();
26 }
27 }
28
```

Console Output:

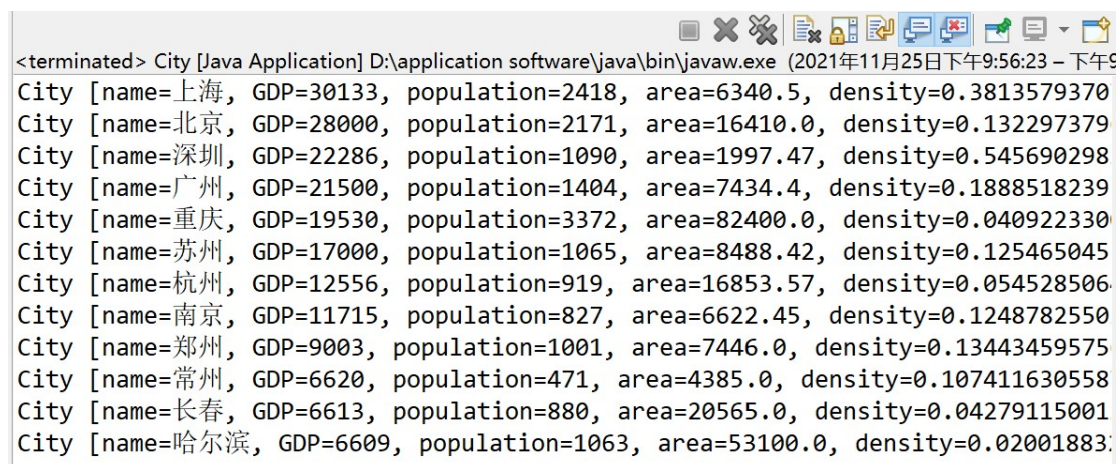
```
<terminated> Cylinder [Java Application] D:\application software\java\bin\javaw.exe (2021年11月23日下午4:55:53 - 下午4:
当前柱体的体积为: 3.1415926
当前柱体的体积为: 24.0
```

图 1.2 实验 1 结果

可以看到程序在 `main` 方法中正确输出了柱体的相关信息，本实验成功。

实验题 2 测试 `List<E>` 的相关方法

本实验依然十分简单，先看测试结果。



```
<terminated> City [Java Application] D:\application software\java\bin\javaw.exe (2021年11月25日下午9:56:23 - 下午9:
City [name=上海, GDP=30133, population=2418, area=6340.5, density=0.3813579370
City [name=北京, GDP=28000, population=2171, area=16410.0, density=0.132297379
City [name=深圳, GDP=22286, population=1090, area=1997.47, density=0.545690298
City [name=广州, GDP=21500, population=1404, area=7434.4, density=0.1888518239
City [name=重庆, GDP=19530, population=3372, area=82400.0, density=0.040922330
City [name=苏州, GDP=17000, population=1065, area=8488.42, density=0.125465045
City [name=杭州, GDP=12556, population=919, area=16853.57, density=0.054528506
City [name=南京, GDP=11715, population=827, area=6622.45, density=0.1248782550
City [name=郑州, GDP=9003, population=1001, area=7446.0, density=0.13443459575
City [name=常州, GDP=6620, population=471, area=4385.0, density=0.107411630558
City [name=长春, GDP=6613, population=880, area=20565.0, density=0.04279115001
City [name=哈尔滨, GDP=6609, population=1063, area=53100.0, density=0.02001883:
```

图 2.1 实验 2 结果

根据题目要求设计 `City` 类，并且实现 `Comparable<City>` 接口。这里也解答我前期的一个疑惑——在实验 2 中实现 `Comparable` 接口而不提供相关的泛型参数，编译器会警告。然而，加入 `City` 参数，编译器将不再报错。最后在 `List` 中直接调用 `Collections.sort(cityList)` 这个方法，再逐一输出，整个实验就完成了。

然后，我又学习了 `Comparator` 接口，同样十分简单。就相当于 C++ 中的 `sort` 函数中最后一个仿函数指针参数一样。在 `java` 中只要使用匿名类实现 `Comparator` 接口即可。

```

Collections.sort(cityList, new Comparator<City>() {

    @Override
    public int compare(City o1, City o2) {
        return o2.population - o1.population;
    }

});
System.out.println("按人口排序: ");
for(City i:cityList) {
    System.out.println(i);
}

```

图 2.2 匿名类实现 Comparator 接口

但是，这里会出现一个问题，如下图。

```

Collections.sort(cityList, new Comparator<City>() {
    //将double的差值强制类型转换为int会出问题
    @Override
    public int compare(City o1, City o2) {
        if(o2.averageGDP > o1.averageGDP) {
            return 1;
        }
        return -1;
    }

});

System.out.println("按人均GDP排序");
for(City i:cityList) {
    System.out.println(i);
}

```

图 2.3 浮点数减法出错

Compare()方法的返回值是 int 类型。如果直接返回两个浮点数的减法，编译器直接报错。然而，对两个浮点数相减的结果强制转换为 int 类型在比较的过程中还会出错，所以需要自己手动判断返回值。

最终，实验成功。


```

<terminated> ComparatorTest [Java Application] D:\application software\java\bin\javaw.exe (2021年11月25日下午9:56:51 - 下午9:56:51)
按人口排序:
City [name=重庆, GDP=19530, population=3372, area=82400.0, density=0.04092233009708738, averageGDP=5.791814946619217]
City [name=上海, GDP=30133, population=2418, area=6340.5, density=0.3813579370712089, averageGDP=12.461952026468156]
City [name=北京, GDP=28000, population=2171, area=16410.0, density=0.13229737964655697, averageGDP=12.897282358360203]
City [name=广州, GDP=21500, population=1404, area=7434.4, density=0.1888518239535134, averageGDP=15.313390313390313]
City [name=深圳, GDP=22286, population=1090, area=1997.47, density=0.5456902982272575, averageGDP=20.445871559633026]
City [name=苏州, GDP=17000, population=1065, area=8488.42, density=0.12546504532056613, averageGDP=15.96244131455399]
City [name=哈尔滨, GDP=6609, population=1063, area=53100.0, density=0.02001883239171375, averageGDP=6.2173095014111]
City [name=郑州, GDP=9003, population=1001, area=7446.0, density=0.13443459575611066, averageGDP=8.994005994005994]
City [name=杭州, GDP=12556, population=919, area=16853.57, density=0.05452850642326819, averageGDP=13.662676822633298]
City [name=长春, GDP=6613, population=880, area=20565.0, density=0.042791150012156574, averageGDP=7.514772727272727]
City [name=南京, GDP=11715, population=827, area=6622.45, density=0.12487825502646302, averageGDP=14.16565900846433]
City [name=常州, GDP=6620, population=471, area=4385.0, density=0.10741163055872292, averageGDP=14.0552016985138]
按人均GDP排序
City [name=深圳, GDP=22286, population=1090, area=1997.47, density=0.5456902982272575, averageGDP=20.445871559633026]
City [name=苏州, GDP=17000, population=1065, area=8488.42, density=0.12546504532056613, averageGDP=15.96244131455399]
City [name=广州, GDP=21500, population=1404, area=7434.4, density=0.1888518239535134, averageGDP=15.313390313390313]
City [name=南京, GDP=11715, population=827, area=6622.45, density=0.12487825502646302, averageGDP=14.16565900846433]
City [name=常州, GDP=6620, population=471, area=4385.0, density=0.10741163055872292, averageGDP=14.0552016985138]
City [name=杭州, GDP=12556, population=919, area=16853.57, density=0.05452850642326819, averageGDP=13.662676822633298]
City [name=北京, GDP=28000, population=2171, area=16410.0, density=0.13229737964655697, averageGDP=12.897282358360203]
City [name=上海, GDP=30133, population=2418, area=6340.5, density=0.3813579370712089, averageGDP=12.461952026468156]
City [name=郑州, GDP=9003, population=1001, area=7446.0, density=0.13443459575611066, averageGDP=8.994005994005994]
City [name=长春, GDP=6613, population=880, area=20565.0, density=0.042791150012156574, averageGDP=7.514772727272727]
City [name=哈尔滨, GDP=6609, population=1063, area=53100.0, density=0.02001883239171375, averageGDP=6.2173095014111]
City [name=重庆, GDP=19530, population=3372, area=82400.0, density=0.04092233009708738, averageGDP=5.791814946619217]
按人口密度排序:
City [name=深圳, GDP=22286, population=1090, area=1997.47, density=0.5456902982272575, averageGDP=20.445871559633026]
City [name=上海, GDP=30133, population=2418, area=6340.5, density=0.3813579370712089, averageGDP=12.461952026468156]
City [name=广州, GDP=21500, population=1404, area=7434.4, density=0.1888518239535134, averageGDP=15.313390313390313]
City [name=郑州, GDP=9003, population=1001, area=7446.0, density=0.13443459575611066, averageGDP=8.994005994005994]
City [name=北京, GDP=28000, population=2171, area=16410.0, density=0.13229737964655697, averageGDP=12.897282358360203]
City [name=苏州, GDP=17000, population=1065, area=8488.42, density=0.12546504532056613, averageGDP=15.96244131455399]
City [name=南京, GDP=11715, population=827, area=6622.45, density=0.12487825502646302, averageGDP=14.16565900846433]
City [name=常州, GDP=6620, population=471, area=4385.0, density=0.10741163055872292, averageGDP=14.0552016985138]
City [name=杭州, GDP=12556, population=919, area=16853.57, density=0.05452850642326819, averageGDP=13.662676822633298]

```

图 2.4 定制排序

下面是本实验的 UML 类图

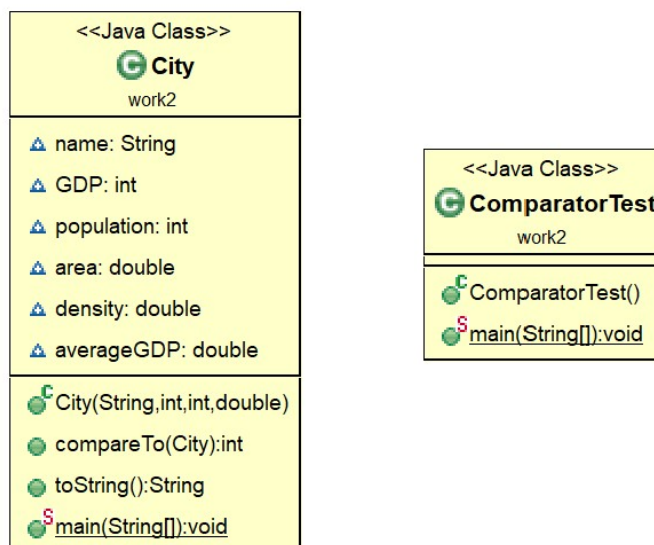


图 2.5 实验 2 UML 类图

实验题 3 测试 TreeMap<K,V>的相关用法

首先根据题目要求设计 Student 类。

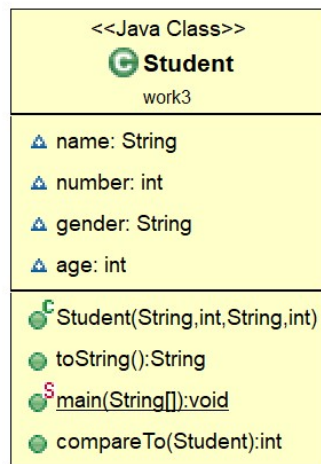


图 3.1 实验 3UML 类图

学生 stuMap 的遍历，我使用了两种方法。第一种使用 `entrySet`，相对简单。

```
//遍历所有学生的信息，输出学号、姓名、城市
Set<Map.Entry<Student, String>>entrySet = stuMap.entrySet();
for(Map.Entry<Student, String>entry : entrySet) {
    Student key = entry.getKey();
    String value = entry.getValue();
    System.out.println("学号: "+key.number+"\t姓名: "+key.name+"\t城市: "+value);
}
```

图 3.2 `entrySet` 遍历方法

第二种使用迭代器，我认为还是比较麻烦的。

```
//用迭代器遍历
boolean flag1 = false;
Iterator<Entry<Student, String>>iterator = entrySet.iterator();
while(iterator.hasNext()) {
    Entry<Student, String>entry = iterator.next();
    if(number==entry.getKey().number) {
        flag1 = true;
        System.out.println(entry.getKey()+entry.getValue());
        break;
    }
}
```

图 3.3 迭代器遍历方法

下面是遍历结果。

```
Student [Java Application] D:\application software\java\bin\javaw.exe (2021年11月25日下午10:17:53)
学号: 1001      姓名: 李明      城市: 上海
学号: 1002      姓名: 张帆      城市: 广州
学号: 1003      姓名: 胡青      城市: 上海
学号: 1004      姓名: 王利军    城市: 成都
学号: 1005      姓名: 陈小兵    城市: 西安
学号: 1006      姓名: 姚华芝    城市: 西安
学号: 1007      姓名: 唐秀      城市: 青岛
学号: 1008      姓名: 吴洪      城市: 成都
```

图 3.4 stuMap 遍历结果

这里需要注意，如果使用 TreeMap 增删元素，元素的类必须实现 Comparable 接口，不然就会出错。

然后测试搜索功能、增加和删除功能。

```
Student [Java Application] D:\application software\java\bin\javaw.exe (2021年11月25日下午10:17:53)
学号: 1001      姓名: 李明      城市: 上海
学号: 1002      姓名: 张帆      城市: 广州
学号: 1003      姓名: 胡青      城市: 上海
学号: 1004      姓名: 王利军    城市: 成都
学号: 1005      姓名: 陈小兵    城市: 西安
学号: 1006      姓名: 姚华芝    城市: 西安
学号: 1007      姓名: 唐秀      城市: 青岛
学号: 1008      姓名: 吴洪      城市: 成都
请输入要搜索的学号:
1005
Student [name=陈小兵, number=1005, gender=男, age=20] 西安
```

图 3.5 搜索功能

请输入要添加学生的(姓名 学号 性别 年龄 城市):

邦子 1010 男 18 西安

```
学号: 1001      姓名: 李明      城市: 上海
学号: 1002      姓名: 张帆      城市: 广州
学号: 1003      姓名: 胡青      城市: 上海
学号: 1004      姓名: 王利军    城市: 成都
学号: 1005      姓名: 陈小兵    城市: 西安
学号: 1006      姓名: 姚华芝    城市: 西安
学号: 1007      姓名: 唐秀      城市: 青岛
学号: 1008      姓名: 吴洪      城市: 成都
学号: 1010      姓名: 邦子      城市: 西安
```

图 3.6 添加功能

请输入要删除学生的学号:

1003

胡青删除成功

学号: 1001	姓名: 李明	城市: 上海
学号: 1002	姓名: 张帆	城市: 广州
学号: 1004	姓名: 王利军	城市: 成都
学号: 1005	姓名: 陈小兵	城市: 西安
学号: 1006	姓名: 姚华芝	城市: 西安
学号: 1007	姓名: 唐秀	城市: 青岛
学号: 1008	姓名: 吴洪	城市: 成都
学号: 1010	姓名: 邦子	城市: 西安

图 3.7 删除功能

在 C++ 的 STL 中, 我使用 `map` 相关的功能已经非常熟练, java 中的集合框架也是大同小异, 我不再赘述。

下面一个要求, 输出某个城市的所有学生信息。说一说具体的算法。首先要新建一个键为城市、值为 `Student` 的 `ArrayList` 的 `hashMap`, 然后遍历所有的 `stuMap`, 根据对应的城市做出相应的操作。如果在 `hashMap` 中没有整个城市, 那么加入这个城市, 在值的列表中加入学生信息, 如果 `hashMap` 中存在这个城市的信息, 直接在值的列表中加入这个学生的信息, 这个算法还是比较简单的。

```
成都:[Student [name=王利军, number=1004, gender=男, age=19], Student [name=吴洪, number=1008, gender=男, age=19]]
上海:[Student [name=李明, number=1001, gender=男, age=19], Student [name=胡青, number=1003, gender=女, age=20]]
广州:[Student [name=张帆, number=1002, gender=女, age=19]]
西安:[Student [name=陈小兵, number=1005, gender=男, age=20], Student [name=姚华芝, number=1006, gender=女, age=18]]
青岛:[Student [name=唐秀, number=1007, gender=女, age=19]]
```

图 3.8 城市列表

本次实验尚未解决的疑难:

本次实验相对简单, 疑难问题已经全部解决。

心得体会:

本次实验还是相对简单, 很多知识与 C++ 中的泛型以及 STL 中类似, 在 java 中上手十分迅速。在实验一中我学会了泛型的基本使用方法, 实验 2 中使用 `List<E>` 实现了一些方法, 再次使用 `Comparable` 接口, 此外还学习了 `Comparator` 接口。实验 3 中使用了 `hashMap` 和 `treeMap`, 他们封装的接口都很好用, 但是内部的具体细节还需要在数据结构课程中深究。