day06【Map】

主要内容

- Collections类
- 排序算法
- Map集合
- 图书管理系统

教学目标

- [ ] 能够使用集合工具类
- [ ] 能够使用Comparator比较器进行排序
- [ ] 能够使用可变参数
- [ ] 能够理解冒泡排序的原理
- [ ] 能够说出Map集合特点
- [ ] 使用Map集合添加方法保存数据
- [ ] 使用"键找值"的方式遍历Map集合
- [ ] 使用"键值对"的方式遍历Map集合
- [ ] 能够使用HashMap存储自定义键值对的数据
- [ ] 能够理解图书管理系统案例

第一章 Collections类

1.1 Collections常用功能

- `java.utils.Collections`是集合工具类，用来对集合进行操作。

  常用方法如下：

- `public static void shuffle(List<?> list)`:打乱集合顺序。
- `public static <T> void sort(List<T> list)`:将集合中元素按照默认规则排序。
- `public static <T> void sort(List<T> list, Comparator<? super T> )`:将集合中元素按照指定规则排序。

代码演示：

```
public class CollectionsDemo {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<Integer>();

        list.add(100);
```

```
        list.add(300);
        list.add(200);
        list.add(50);
        //排序方法
        Collections.sort(list);
        System.out.println(list);
    }
}
```

结果:

```
[50,100, 200, 300]
```

我们的集合按照默认的自然顺序进行了排列，如果想要指定顺序那该怎么办呢?

1.2 Comparator比较器

创建一个学生类，存储到ArrayList集合中完成指定排序操作。

Student 类

```
public class Student{
    private String name;
    private int age;
    //构造方法
    //get/set
    //toString
}
```

测试类:

```
public class Demo {

    public static void main(String[] args) {
        // 创建四个学生对象 存储到集合中
        ArrayList<Student> list = new ArrayList<Student>();

        list.add(new Student("rose",18));
        list.add(new Student("jack",16));
        list.add(new Student("abc",20));
        Collections.sort(list, new Comparator<Student>() {
          @Override
            public int compare(Student o1, Student o2) {
            return o1.getAge()-o2.getAge();//以学生的年龄升序
         }
        });


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

```
Student{name='jack', age=16}
```

```
Student{name='rose', age=18}
Student{name='abc', age=20}
```

1.3 可变参数

什么是可变参数:一个方法需要接受多个参数，并且多个参数类型一致

格式:

```
修饰符 返回值类型 方法名(参数类型... 形参名){  }
```

代码演示:

```
  public class ChangeArgs {
    public static void main(String[] args) {
        int sum = getSum(6, 7, 2, 12, 2121);
        System.out.println(sum);
    }
    public static int getSum(int... arr) {
        int sum = 0;
         for (int a : arr) {
         sum += a;
        }
         return sum;
    }
}
```

注意:

1.一个方法只能有一个可变参数

2.如果方法中有多个参数，可变参数要放到最后。

应用场景: Collections

在Collections中也提供了添加一些元素方法： `public static <T> boolean addAll(Collection<T> c, T... elements)` :往集合中添加一些元素。

代码演示:

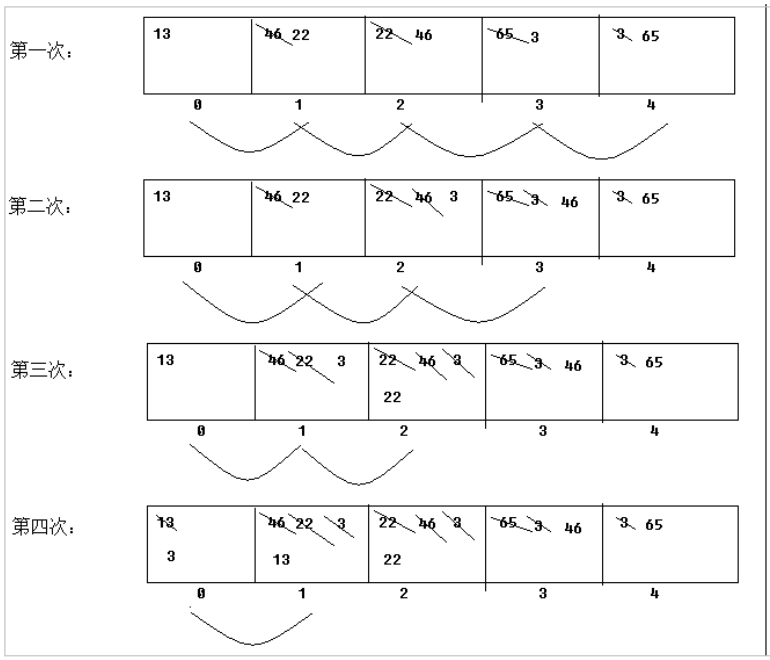
```
public class CollectionsDemo {
    public static void main(String[] args) {
      ArrayList<Integer> list = new ArrayList<Integer>();
      //原来写法
      //list.add(12);
      //list.add(14);
      //list.add(15);
      //list.add(1000);
      //采用工具类 完成 往集合中添加元素
      Collections.addAll(list, 5, 222, 1, 2);
      System.out.println(list);
}
```

第二章 排序算法

我们会使用数组来存储多个数据，有时候我们要将数组中的元素进行排序。比如原始数组为{5, 1, 3, 2}，排序后为{1, 2, 3, 5}。常见排序算法有：选择排序，冒泡排序，快速排序等。

2.1 冒泡排序

冒泡排序原理：相邻元素比较，大的往后放。




```
0  第一次：
        arr[0] 与 arr[1]      j < 5-1 -0
        arr[1] 与 arr[2]
        arr[2] 与 arr[3]
        arr[3] 与 arr[4]
1  第二次：                   j < 5-1 -1
        arr[0] 与 arr[1]
        arr[1] 与 arr[2]
        arr[2] 与 arr[3]
2  第三次：                   j < 5-1 -2
        arr[0] 与 arr[1]
        arr[1] 与 arr[2]
3  第四次：                   j < 5-1 -3
        arr[0] 与 arr[1]

                              j < arr.length-1 - i;
外层循环用来控制比较的次数
内层循环用来控制参与比较的元素
```

实现代码

```
public static void main(String[] args) {
    int[] arr = new int[] {55, 22, 99, 88};
    //功能
    //外层循环用来控制数组循环的圈数
    for (int i = 0; i < arr.length-1; i++) {
        //内层循环用来完成元素值比较，把大的元素值互换到后面
        for (int j = 0; j < arr.length-1-i; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
    System.out.println(Arrays.toString(arr));
}
```

2.2 小结

冒泡排序原理：相邻元素比较，大的往后放。

第三章 Map集合

3.1 概述

现实生活中，我们常会看到这样的一种集合：IP地址与主机名，身份证号与个人，系统用户名与系统用户对象等，这种一一对应的关系，就叫做映射。Java提供了专门的集合类用来存放这种对象关系的对象，即`java.util.Map`接口。

我们通过查看`Map`接口描述，发现`Map`接口下的集合与`Collection`接口下的集合，它们存储数据的形式不同，如下图。

- `Collection`中的集合，元素是孤立存在的（理解为单身），向集合中存储元素采用一个个元素的方式存储。
- `Map`中的集合，元素是成对存在的(理解为夫妻)。每个元素由键与值两部分组成，通过键可以找对所对应的值。
- `Collection`中的集合称为单列集合，`Map`中的集合称为双列集合。
- 需要注意的是，`Map`中的集合不能包含重复的键，值可以重复；每个键只能对应一个值。

3.2 Map的常用子类

通过查看Map接口描述，看到Map有多个子类，这里我们主要讲解常用的HashMap集合、LinkedHashMap集合。

- **HashMap<K,V>**：存储数据采用的哈希表结构，元素的存取顺序不能保证一致。由于要保证键的唯一、不重复，需要重写键的hashCode()方法、equals()方法。
- **LinkedHashMap<K,V>**：HashMap下有个子类LinkedHashMap，存储数据采用的哈希表结构+链表结构。通过链表结构可以保证元素的存取顺序一致；通过哈希表结构可以保证的键的唯一、不重复，需要重写键的hashCode()方法、equals()方法。

> tips：Map接口中的集合都有两个泛型变量<K,V>,在使用时，要为两个泛型变量赋予数据类型。两个泛型变量<K,V>的数据类型可以相同，也可以不同。

3.3 Map的常用方法

Map接口中定义了很多方法，常用的如下：

- `public V put(K key, V value)`：把指定的键与指定的值添加到Map集合中。
- `public V remove(Object key)`：把指定的键 所对应的键值对元素 在Map集合中删除，返回被删除元素的值。
- `public V get(Object key)` 根据指定的键，在Map集合中获取对应的值。
- `public Set<K> keySet()`：获取Map集合中所有的键，存储到Set集合中。
- `public Set<Map.Entry<K,V>> entrySet()`：获取到Map集合中所有的键值对对象的集合(Set集合)。
- `public boolean containKey(Object key)`:判断该集合中是否有此键。

Map接口的方法演示

```
public class MapDemo {
    public static void main(String[] args) {
        //创建 map对象
        HashMap<String, String>  map = new HashMap<String, String>();

        //添加元素到集合
        map.put("黄晓明", "杨颖");
        map.put("文章", "马伊琍");
        map.put("邓超", "孙俪");
        System.out.println(map);

        //String remove(String key)
        System.out.println(map.remove("邓超"));
        System.out.println(map);

        // 想要查看 黄晓明的媳妇 是谁
        System.out.println(map.get("黄晓明"));
        System.out.println(map.get("邓超"));
    }
}
```

> tips:
>
> 使用put方法时，若指定的键(key)在集合中没有，则没有这个键对应的值，返回null，并把指定的键值添加到集合中;
>
> 若指定的键(key)在集合中存在，则返回值为集合中键对应的值（该值为替换前的值），并把指定键所对应的值，替换成指定的新值。

3.4 Map的遍历

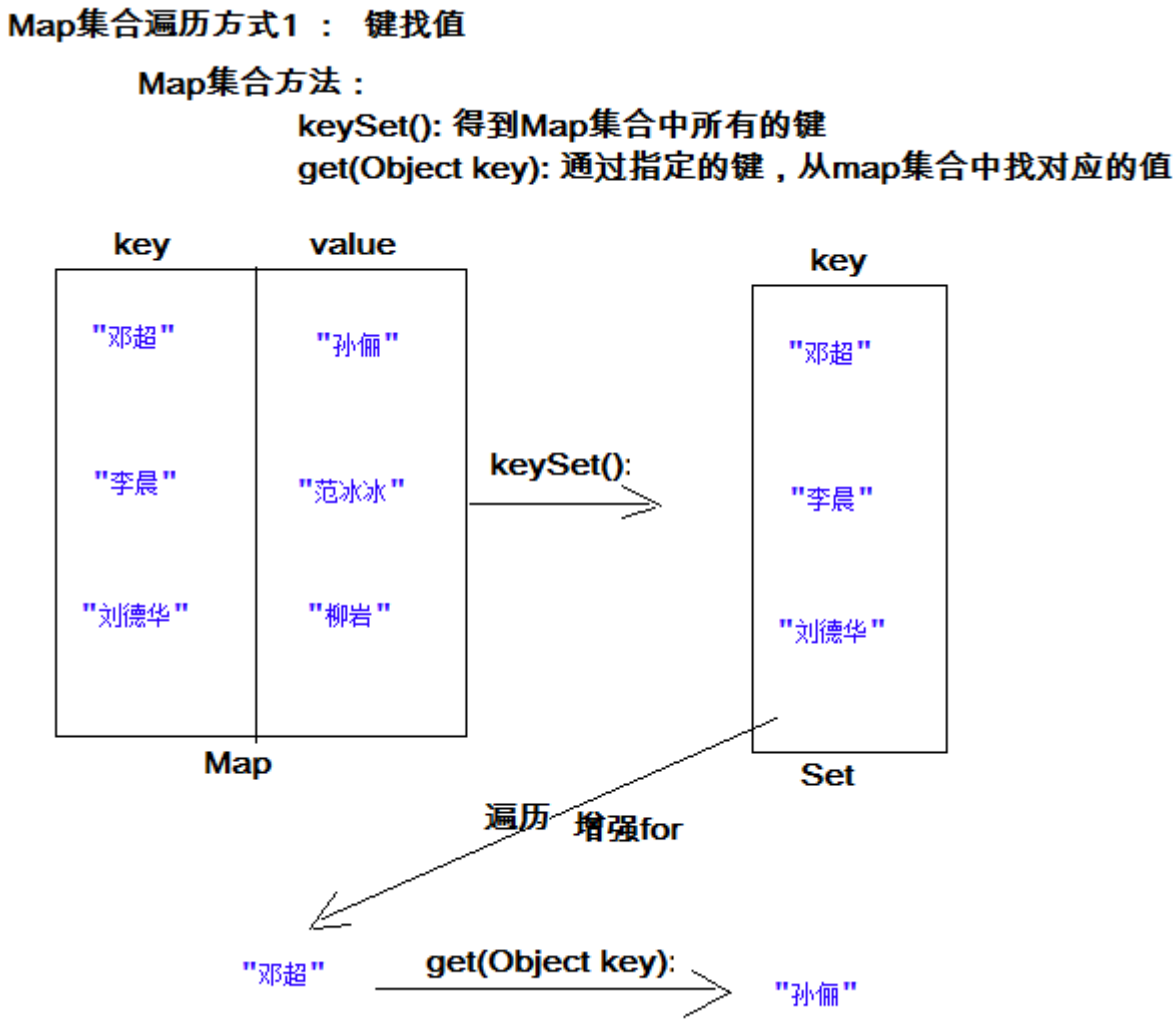
方式1:键找值方式

通过元素中的键，获取键所对应的值

分析步骤：

1. 获取Map中所有的键，由于键是唯一的，所以返回一个Set集合存储所有的键。方法提示:`keyset()`
2. 遍历键的Set集合，得到每一个键。
3. 根据键，获取键所对应的值。方法提示:`get(K key)`

遍历图解：



方式2:键值对方式

即通过集合中每个键值对(Entry)对象，获取键值对(Entry)对象中的键与值。

Entry键值对对象:

我们已经知道，`Map`中存放的是两种对象，一种称为**key**(键)，一种称为**value**(值)，它们在在`Map`中是一一对应关系，这一对对象又称做`Map`中的一个`Entry(项)`。`Entry`将键值对的对应关系封装成了对象。即键值对对象，这样我们在遍历`Map`集合时，就可以从每一个键值对（`Entry`）对象中获取对应的键与对应的值。

在Map集合中也提供了获取所有Entry对象的方法：

- `public Set<Map.Entry<K,V>> entrySet()`: 获取到Map集合中所有的键值对对象的集合(Set集合)。

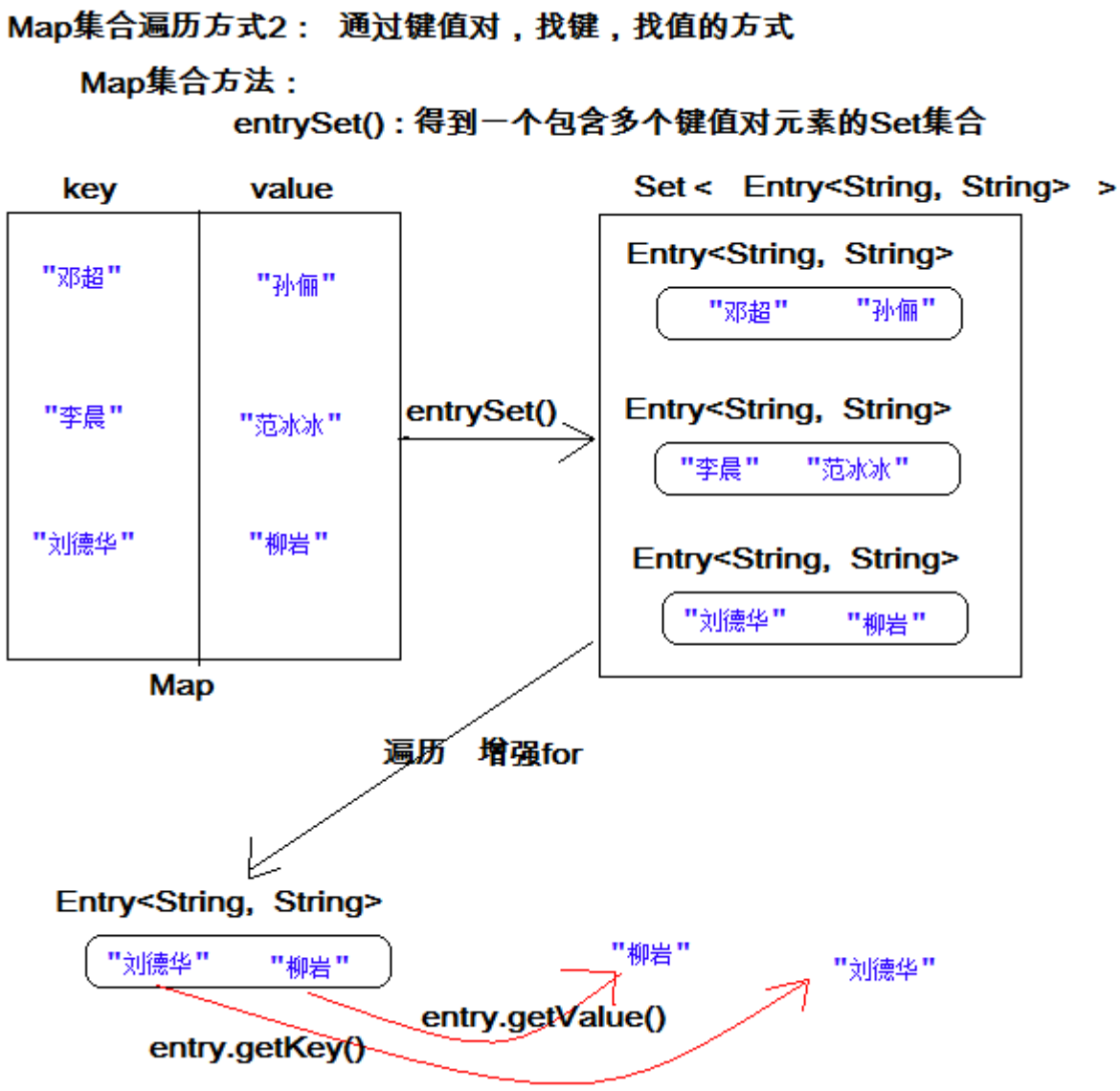
获取了Entry对象，表示获取了一对键和值，那么同样Entry中，分别提供了获取键和获取值的方法：

- `public K getKey()`：获取Entry对象中的键。
- `public V getValue()`：获取Entry对象中的值。

操作步骤与图解：

1. 获取Map集合中，所有的键值对(Entry)对象，以Set集合形式返回。方法提示:`entrySet()`。
2. 遍历包含键值对(Entry)对象的Set集合，得到每一个键值对(Entry)对象。
3. 通过键值对(Entry)对象，获取Entry对象中的键与值。 方法提示:`getkey() getValue()`

遍历图解：



> tips：Map集合不能直接使用迭代器或者foreach进行遍历。但是转成Set之后就可以使用了。

3.5 HashMap存储自定义类型

练习：每位学生（姓名，年龄）都有自己的家庭住址。那么，既然有对应关系，则将学生对象和家庭住址存储到map集合中。学生作为键, 家庭住址作为值。

> 注意，学生姓名相同并且年龄相同视为同一名学生。

编写学生类：

```
public class Student {
    private String name;
    private int age;

    //构造方法
    //get/set
    @Override
    public boolean equals(Object o) {
        if (this == o)
            return true;
        if (o == null || getClass() != o.getClass())
            return false;
        Student student = (Student) o;
        return age == student.age && Objects.equals(name, student.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, age);
    }
}
```

编写测试类：

```
public class HashMapTest {
    public static void main(String[] args) {
        //1,创建Hashmap集合对象。
        Map<Student,String> map = new HashMap<Student,String>();
        //2,添加元素。
        map.put(new Student("lisi",28), "上海");
        map.put(new Student("wangwu",22), "北京");
        map.put(new Student("wangwu",22), "南京");

        //3,取出元素。键找值方式
        Set<Student> keySet = map.keySet();
        for(Student key: keySet){
            String value = map.get(key);
            System.out.println(key.toString()+"....."+value);
        }
    }
}
```

- 当给HashMap中存放自定义对象时，如果自定义对象作为key存在，这时要保证对象唯一，必须复写对象的hashCode和equals方法(如果忘记，请回顾HashSet存放自定义对象)。
- 如果要保证map中存放的key和取出的顺序一致，可以使用`java.util.LinkedHashMap`集合来存放。

3.6 Map集合练习

需求:

计算一个字符串中每个字符出现次数。

分析：

1. 获取一个字符串对象
2. 创建一个Map集合，键代表字符，值代表次数。
3. 遍历字符串得到每个字符。
4. 判断Map中是否有该键。
5. 如果没有，第一次出现，存储次数为1；如果有，则说明已经出现过，获取到对应的值进行++，再次存储。
6. 打印最终结果

方法介绍

`public boolean containKey(Object key)`:判断该集合中是否有此键。

代码：

```
public class MapTest {
public static void main(String[] args) {
        //友情提示
        System.out.println("请录入一个字符串:");
        String line = new Scanner(System.in).nextLine();
        // 定义 每个字符出现次数的方法
        findChar(line);
    }
    private static void findChar(String line) {
        //1:创建一个集合 存储  字符 以及其出现的次数
        HashMap<Character, Integer> map = new HashMap<Character, Integer>();
        //2:遍历字符串
        for (int i = 0; i < line.length(); i++) {
            char c = line.charAt(i);
            //判断 该字符 是否在键集中
            if (!map.containsKey(c)) {//说明这个字符没有出现过
                //那就是第一次
                map.put(c, 1);
            } else {
                //先获取之前的次数
                Integer count = map.get(c);
                //count++;
                //再次存入  更新
                map.put(c, ++count);
            }
        }
        System.out.println(map);
    }
}
```

第四章 图书管理系统

4.1 图书管理系统项目演示

--------欢迎来到学生管理系统--------
1. 查看所有书籍　　主界面和选择
2. 添加书
3. 删除书
4. 修改书
5. 退出
请输入你的选择：

--------欢迎来到学生管理系统--------
1. 查看所有书籍
2. 添加书
3. 删除书
4. 修改书
5. 退出
请输入你的选择：
1　　查看书籍
类型　　书名　　价格
名著
　　西游记　19.0
　　水浒传　29.0
it书籍
　　Java入门到精通　99.0
　　PHP入门到精通　9.9

--------欢迎来到学生管理系统--------
1. 查看所有书籍
2. 添加书
3. 删除书
4. 修改书
5. 退出

添加书籍

请输入你的选择：
2
请输入要添加书籍的类型：
名著
请输入要添加的书名：
三国演义
请输入要添加书的价格：
19.9
添加三国演义成功

--------欢迎来到学生管理系统--------
1. 查看所有书籍
2. 添加书
3. 删除书
4. 修改书
5. 退出

删除书籍

请输入你的选择：
3
请输入要删除书籍的类型：
名著
请输入要删除的书名：
西游记
删除西游记书籍成功

--------欢迎来到学生管理系统--------
1. 查看所有书籍
2. 添加书
3. 删除书
4. 修改书
5. 退出
请输入你的选择：
4 修改书籍
请输入要修改书籍的类型：
名著
请输入要修改的书名：
三国演义
请输入新的书名：
三人行
请输入新的价格：
299
修改成功
--------欢迎来到学生管理系统--------
1. 查看所有书籍
2. 添加书
3. 删除书
4. 修改书
5. 退出
请输入你的选择：
5 退出
谢谢你的使用

图书管理系统分析: 1.定义Book类 2.完成主界面和选择 3.完成查询所有图书 4.完成添加图书 5.完成删除图书 6.完成修改图书 7.使用Debug追踪调试

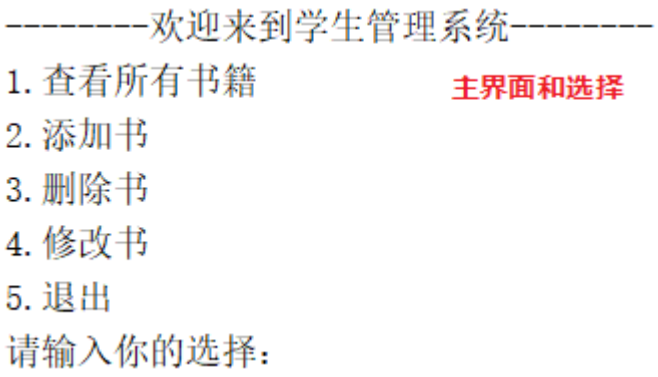
4.2 图书管理系统之标准Book类

```
1          查看书籍
类型          书名      价格
名著
          西游记    19.0
          水浒传    29.0
it书籍
          Java入门到精通    99.0
          PHP入门到精通     9.9
```

我们发现每一本书都有书名和价格,定义一个Book类表示书籍

```
public class Book {
    private String name;
    private double price;

    public Book() {
    }

    public Book(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}
```

4.3 图书管理系统之主界面和选择的实现



主界面的内容其实就是通过打印语句打印出来的.但是要注意因为每个操作过后都会重新回到主界面,所以使用`while(true)`死循环的方式.

```
public class BookManager {
    public static void main(String[] args) {
        while (true) {
            //这是学生管理系统的主界面
            System.out.println("--------欢迎来到学生管理系统--------");
            System.out.println("1.查看所有书籍");
            System.out.println("2.添加书");
            System.out.println("3.删除书");
            System.out.println("4.修改书");
            System.out.println("5.退出");
            System.out.println("请输入你的选择：");

            //创建键盘录入对象
            Scanner sc = new Scanner(System.in);
            int num = sc.nextInt();
            switch (num) {
                case 1:
                    // 查看所有书籍
                    break;
                case 2:
                    // 添加书籍
                    break;
                case 3:
                    // 删除书
                    break;
                case 4:
                    // 修改书
                    break;
                case 5:
                    // 退出
                    break;
                default:
                    System.out.println("输入错误,请重新输入");
                    break;
            }
        }
    }
}
```

4.4 图书管理系统之查询所有图书


```
public class BookManager {
    public static void main(String[] args) {
        Map<String, ArrayList<Book>> map = new HashMap<>();
        // 创建集合对象，用于存储学生数据
        ArrayList<Book> it = new ArrayList<Book>();
        it.add(new Book("Java入门到精通", 99));
        it.add(new Book("PHP入门到精通", 9.9));
        map.put("it书籍", it);
        ArrayList<Book> mz = new ArrayList<Book>();
        mz.add(new Book("西游记", 19));
        mz.add(new Book("水浒传", 29));
        map.put("名著", mz);

        while (true) {
            //这是学生管理系统的主界面
            System.out.println("--------欢迎来到学生管理系统--------");
            System.out.println("1.查看所有书籍");
            System.out.println("2.添加书");
            System.out.println("3.删除书");
            System.out.println("4.修改书");
            System.out.println("5.退出");
            System.out.println("请输入你的选择：");

            //创建键盘录入对象
            Scanner sc = new Scanner(System.in);
            int num = sc.nextInt();
            switch (num) {
                case 1:
                    // 查看所有书籍
                    findAllBook(map);
                    break;
                case 2:
                    // 添加书籍
                    break;
                case 3:
```

```
                    // 删除书
                    break;
                case 4:
                    // 修改书
                    break;
                case 5:
                    // 退出
                    System.out.println("谢谢你的使用");
                    System.exit(0); // JVM退出
                    break;
                default:
                    System.out.println("输入错误,请重新输入");
                    break;
            }
        }
    }

    private static void findAllBook(Map<String, ArrayList<Book>> map) {
        System.out.println("类型\t\t书名\t价格");
        Set<Map.Entry<String, ArrayList<Book>>> entries = map.entrySet();
        for (Map.Entry<String, ArrayList<Book>> entry : entries) {
            String key = entry.getKey();
            System.out.println(key);

            ArrayList<Book> value = entry.getValue();
            for (Book book : value) {
                System.out.println("\t\t" + book.getName() + "\t" + book.getPrice());
            }
        }
    }
}
```

4.5 图书管理系统之添加图书

--------欢迎来到学生管理系统--------
1. 查看所有书籍
2. 添加书
3. 删除书
4. 修改书
5. 退出
请输入你的选择：
2
请输入要添加书籍的类型：
名著
请输入要添加的书名：
三国演义
请输入要添加书的价格：
19.9
添加三国演义成功

添加书籍

```
private static void addBook(Map<String, ArrayList<Book>> map) {
    // 创建键盘录入对象
    Scanner sc = new Scanner(System.in);
    System.out.println("请输入要添加书籍的类型:");
    String type = sc.next();
    System.out.println("请输入要添加的书名:");
    String name = sc.next();
    System.out.println("请输入要添加书的价格:");
    double price = sc.nextDouble();
    Book book = new Book(name, price);

    // 拿到书籍列表
    ArrayList<Book> books = map.get(type);
    if (books == null) {
        // 如果书籍列表不存在创建一个书籍列表
        books = new ArrayList<>();
        map.put(type, books);
    }

    // 将书添加到集合中
    books.add(book);
    System.out.println("添加" + name + "成功");
}
```

4.6 图书管理系统之删除图书

--------欢迎来到学生管理系统--------
1. 查看所有书籍
2. 添加书
3. 删除书
4. 修改书
5. 退出
请输入你的选择：
3
请输入要删除书籍的类型：
名著
请输入要删除的书名：
西游记
删除西游记书籍成功

删除书籍

```
private static void deleteBook(Map<String, ArrayList<Book>> map) {
    // 创建键盘录入对象
    Scanner sc = new Scanner(System.in);
    System.out.println("请输入要删除书籍的类型:");
    String type = sc.next();
    System.out.println("请输入要删除的书名:");
    String name = sc.next();

    // 拿到书籍列表
    ArrayList<Book> books = map.get(type);
    if (books == null) {
        System.out.println("您删除的书籍类型不存在");
        return;
    }

    for (int i = 0; i < books.size(); i++) {
        Book book = books.get(i);
        if (book.getName().equals(name)) {
            books.remove(i); // 找到这本书,删除这本书
            System.out.println("删除" + name + "书籍成功");
            return; // 删除书籍后结束方法
        }
    }
    System.out.println("没有找到" + name + "书籍");
}
```

4.7 图书管理系统之修改图书

```
--------欢迎来到学生管理系统--------
1. 查看所有书籍
2. 添加书
3. 删除书
4. 修改书
5. 退出
请输入你的选择：
4                                    修改书籍
请输入要修改书籍的类型：
名著
请输入要修改的书名：
三国演义
请输入新的书名：
三人行
请输入新的价格：
299
修改成功
```

```java
private static void editBook(Map<String, ArrayList<Book>> map) {
    // 创建键盘录入对象
    Scanner sc = new Scanner(System.in);
    System.out.println("请输入要修改书籍的类型:");
    String type = sc.next();
    System.out.println("请输入要修改的书名:");
    String oldName = sc.next();

    System.out.println("请输入新的书名:");
    String newName = sc.next();
    System.out.println("请输入新的价格:");
    double price = sc.nextDouble();

    // 拿到书籍列表
    ArrayList<Book> books = map.get(type);
    if (books == null) {
        System.out.println("您修改的书籍类型不存在");
        return;
    }

    for (int i = 0; i < books.size(); i++) {
        Book book = books.get(i);
        if (book.getName().equals(oldName)) {
            // 找到这本书,修改这本书
            book.setName(newName);
            book.setPrice(price);
            System.out.println("修改成功");
```

```
            return; // 修改书籍后结束方法
        }
    }
    System.out.println("没有找到" + oldName + "书籍");
}
```

4.8 Debug追踪调试

之前我们看程序的执行流程都是通过`System.out.println();`但是有不能让程序执行到某条语句后**停下来**,也不能看到程序**具体的执行步骤**.而是执行完所有的语句程序结束了。

断点调试可以查看程序的执行流程和暂停程序.可以快速解决程序中的bug

Debug调试窗口介绍