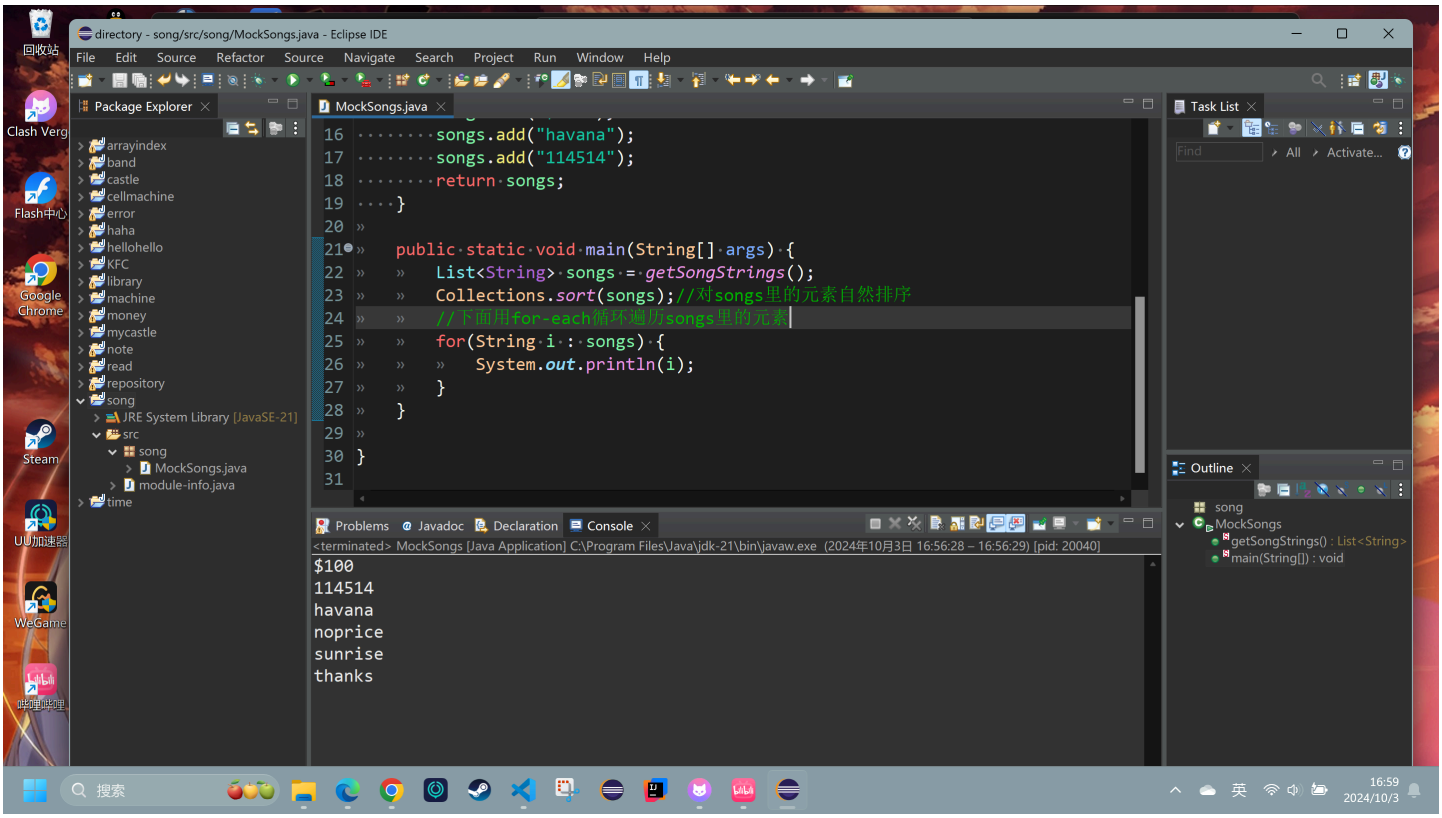
Java08

1. 简单挑战_排序问题

```
package song;

import java.util.ArrayList;
import java.util.List;
import java.util.Collections;

public class MockSongs {

        public static List<String> getSongStrings(){
        List<String> songs = new ArrayList<>();
        //模拟将要处理的列表
        songs.add("sunrise");
        songs.add("noprice");
        songs.add("thanks");
        songs.add("$100");
        songs.add("havana");
        songs.add("114514");
        return songs;
    }

        public static void main(String[] args) {
                List<String> songs = getSongStrings();
                Collections.sort(songs);//对songs里的元素自然排序
                //下面用for-each循环遍历songs里的元素
                for(String i : songs) {
                        System.out.println(i);
                }
        }

}
```

输出结果如下:



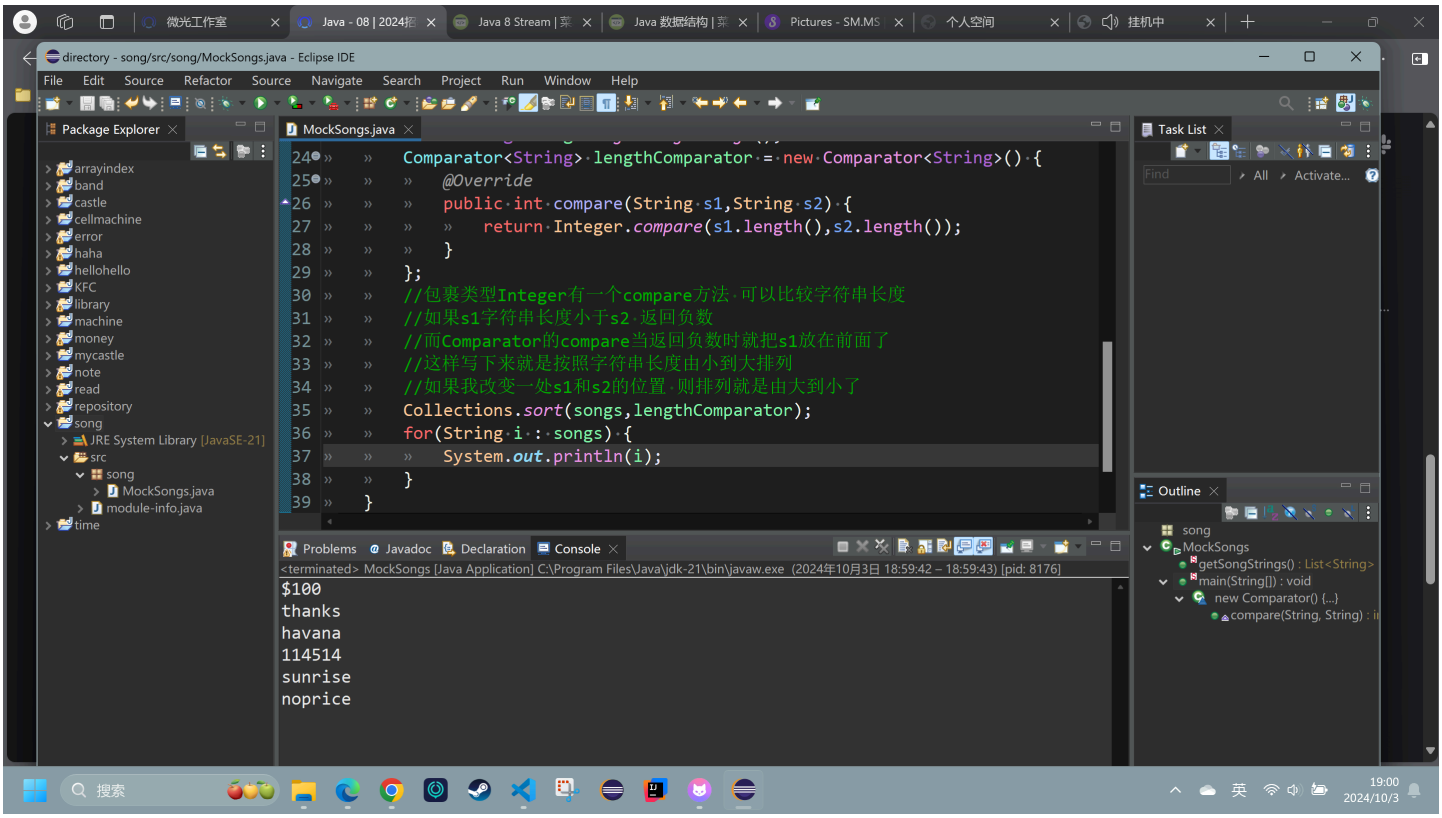
- 之后简单了解了一下Comparator的用法
  Comparator是一个接口 它内部有一个返回int类型的compare方法 参数表是两个对象 我用s1 s2表示 如果返回负整数就把s1放在s2前面 返回0表示在你的比较方法中s1和s2地位相同 保持他们在原列表中的先后顺序 返回正数就把s1放在s2后面 而sort(List,Comparator) 会自动连续多次调用Comparator方法 两两比较 最后得到你想要的排序 前面说过Comparator是一个接口 它没有具体compare方法的实现 所以当你新建一个Comparator对象时要对compare进行重写 按照我上面介绍的规则写出你想要的排序方式

> 我学c语言时写过一个题 把a b c 按照从大到小顺序排列 假设原本排列为a b c 先比较a b大小 a大则不动 b大则互换 再拿之后的第二个位置上的数和c比较 同样的互换规则 最后再把前两个数比较一次 这里连续调用compare两两比较可能类似这个算法 emmm这只是我的猜测 只是从它的工作原理上来讲有点像 我还没接触过其他比较复杂的算法

下面举个按照字符串长度排序的例子:

```
package song;

import java.util.ArrayList;
import java.util.List;
import java.util.Collections;
import java.util.Comparator;

public class MockSongs {

        public static List<String> getSongStrings(){
        List<String> songs = new ArrayList<>();
        //模拟将要处理的列表
        songs.add("sunrise");
        songs.add("noprice");
        songs.add("thanks");
        songs.add("$100");
        songs.add("havana");
        songs.add("114514");
        return songs;
    }

        public static void main(String[] args) {
                List<String> songs = getSongStrings();
                Comparator<String> lengthComparator = new Comparator<String>() {
                        @Override
                        public int compare(String s1,String s2) {
                                return Integer.compare(s1.length(),s2.length());
                        }
                };
                //包裹类型Integer有一个compare方法 可以比较字符串长度
                //如果s1字符串长度小于s2 返回负数
                //而Comparator的compare当返回负数时就把s1放在前面了
                //这样写下来就是按照字符串长度由小到大排列
                //如果我改变一处s1和s2的位置 则排列就是由大到小了
                Collections.sort(songs,lengthComparator);
                for(String i : songs) {
                        System.out.println(i);
                }
        }

}
```

运行成功截图如下:



2. 进阶挑战_加入对象

- 简单了解了注释 最早接触的注释就是@Override 用来覆盖父类的方法 这里@Data放在类的上面可以自动生成get set 重写toString等方法 而@AllArgsConstructor可以自动生成一个构造器 感觉用起来挺方便的 代码没有那么冗长了
- 言归正传 接下来我创建了一个新的List容器 它里面的对象不再是String 而是真正的Song 如果对象是String 在排序时会自动调用toString方法遍历整个列表 根据返回的字符串进行排序 那我在想 如果我为我的Songs类重写一个toString 假设返回title 那系统是否会根据返回的title的字符串来排序呢???

```java
package song;

public class Songs {
        private String title;
    private String artist;
    private int bpm;

    @Override
    public String toString() {
        return title;
    }

    public Songs(String title,String artist,int bpm) {
        this.title = title;
        this.artist = artist;
        this.bpm = bpm;
    }
}
```

```java
package song;

import java.util.List;
import java.util.ArrayList;
import java.util.Collections;

public class MockSongs{

        public static List<Songs> getSongs(){
                List<Songs> songs = new ArrayList<Songs>();
                songs.add(new Songs("sunrise","aaa",90));
                songs.add(new Songs("114514","bbb",60));
                return songs;
        }

        public static void main(String[] args) {
                List<Songs> songs = getSongs();
                Collections.sort(songs);
        }

}
```

居然报错了???虽然sort方法里写着参数表是List 但是它并不认可这里的songs 接着我查看了sort的API

```java
public static <T extends Comparable<? super T>> void sort(List<T> list)
```

这是一个我没见过的方法 简单了解了泛型之后 我大致能明白这个方法是什么意思了 sort的参数表List中的元素为T类型 T类型必须extends Comparable类 但显然Songs是我们自己创建出来的类 肯定不会继承Comparable 但还有一个细节 Comparable类是一个接口啊 这里为什么要用extends呢 后来我明白了泛型里面这里的extends就指"继承"或"实现" 貌似是一种更为广泛的定义 那接下来问题就简单了 我们只需要让Songs类实现Comparable接口就行了
查阅Comparable的API 里面只有一个compareTo函数 那就更简单了 这个函数返回整数 规则类似前面的compare

```
package song;

public class Songs implements Comparable<Songs> {
        private String title;
    private String artist;
    private int bpm;

    @Override
    public String toString() {
        return title;
    }

    public Songs(String title,String artist,int bpm) {
        this.title = title;
        this.artist = artist;
        this.bpm = bpm;
    }

    public int compareTo(Songs s) {
        return title.compareTo(s.title);
    }
    //这里突然想到前面的String类能直接比较
    //不就说明String类实现了Comparable接口吗??????
    //那我直接调用String类的compareTo方法就行辣!!!!!
}
```

```java
package song;

import java.util.List;
import java.util.ArrayList;
import java.util.Collections;

public class MockSongs{

	public static List<Songs> getSongs(){
		List<Songs> songs = new ArrayList<Songs>();
		songs.add(new Songs("sunrise","aaa",90));
		songs.add(new Songs("114514","bbb",60));
		return songs;
	}

	public static void main(String[] args) {
		List<Songs> songs = getSongs();
		Collections.sort(songs);
		//果然那边敲完这里报错就消失了
		for(Songs s : songs) {
			System.out.println(s);
		}
	}

}
```

运行成功 好耶