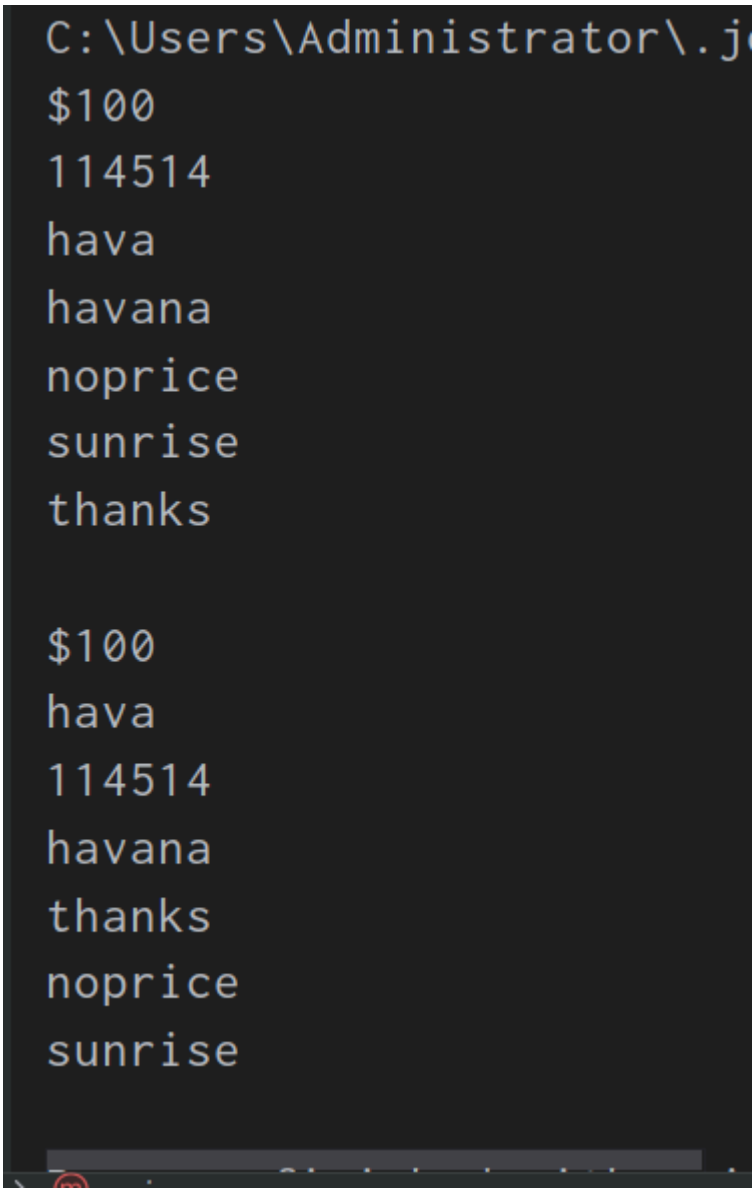
2024090905014-林仪-Java-08

1. 比较器

```
import java.util.ArrayList;
import java.util.Comparator;

public class Main {
    public static void main(String[] args) {
        ArrayList<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("hava");
        songs.add("114514");
        songs.sort(null);
        songs.forEach(System.out::println);
        songs.sort((s1, s2) -> s1.length() - s2.length());
        songs.forEach(System.out::println);



    }
}
```



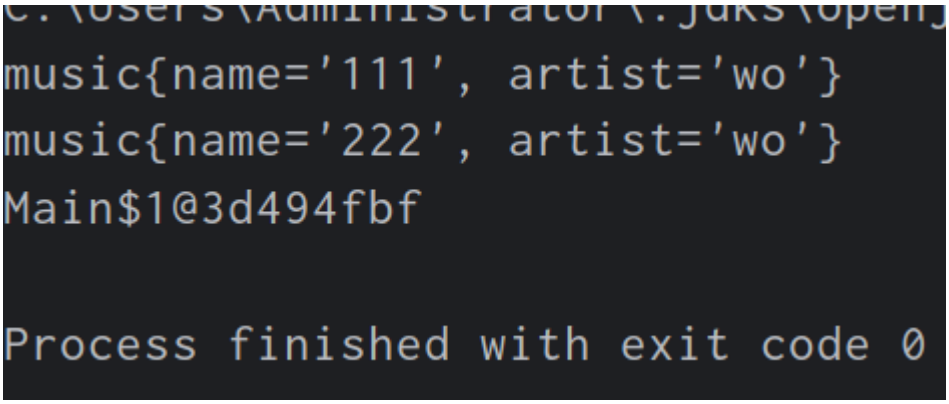
比较器返回两个字符串的差值会根据这个值对元素排序

2. 泛型

泛型可以确定集合里面放什么元素，用ArrayList<T.>的方式，T可为任意类或接口

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<music> musics = new ArrayList<>();
        musics.add(new song("111","wo"));
        musics.add(new song("222","wo"));
        musics.forEach(System.out::println);
        ArrayList<game> games = new ArrayList<>();
```

```
        games.add(new game(){
            @Override
            public void play() {}
        });
        games.forEach(System.out::println);
    }
}
class music{
    String name;
    String artist;

    @Override
    public String toString() {
        return "music{" +
                "name='" + name + '\'' +
                ", artist='" + artist + '\'' +
                '}';
    }

    public music(String name, String artist) {
        this.name = name;
        this.artist = artist;

    }
}
class song extends music{
    public song(String name, String artist) {
        super(name, artist);
    }
}
interface game{
    void play();
}
```



<T.>中的T.指定后可以放入其子类或者实现类
<.? extend A>可以限制类型为A及其子类
<.? super A>可以限制类型为A及其父类

自定义泛型的应用

在类或方法中希望传入的参数可有不同形式时可以自定义其传入形式

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
       ArrayList<song> songs = new ArrayList<>();
       songs.add(new song<Integer>(111));
       songs.add(new song<String>("haha"));
       songs.forEach(System.out::println);

    }
}
class song<T>{
    T name;

    public song(T name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "song{" +
                "name=" + name +
                '}';
```

```
    }
}
```

```
C:\Users\Administrator\.jdks\open
song{name=111}
song{name=haha}

Process finished with exit code 0
```

但是由于类的静态方法和静态变量在类加载时就要确定，此时还没传入泛型的类型，
所以不能使用在静态方法和静态变量中使用自定义泛型