按照字母顺序对歌曲排序

代码如下：

```
import java.util.*;

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();

//        //按字母顺序对songs重排
//        Collections.sort(songs);

        //冒泡排序
        BubbleSort(songs);

        //遍历输出
        for (String song : songs) {
            System.out.println(song);
        }
    }

    public static void BubbleSort(List<String> songs) {
        // 冒泡算法中的变量，用于判断是否排序完成
        boolean swapped = false;

        // 最多冒泡n-1次
        for (int i = 0; i < songs.size() - 1; i++) {
            swapped = false;
            //每次排序参与的元素依次减少一
            for (int j = 0; j < songs.size() - i - 1; j++) {
                //判断j与j+1号元素的字母顺序
                if (songs.get(j).compareTo(songs.get(j+1)) > 0) {
                    // 将j与j+1的元素互换
                    String temp = songs.get(j);
                    songs.set(j, songs.get(j + 1));
                    songs.set(j + 1, temp);
                    swapped = true; // 说明排序没有结束，防止退出循环
```

```
                }
            }

            //此时集合排序已经完成，退出循环
            if (!swapped) {break;}
        }
    }
}
```

笔记：因为由大佬的提示可知，Collection的sort方法可以自动实现排序，所以第一思路就是调用该方法然后for增强输出。然后我突然想到学过冒泡排序，而且我琢磨着这似乎也可以用，所以就写了个冒泡排序，虽然这种排序效率低的可怜，但算是我唯一会的了（悲）。在写if判断条件时，我本来准备用charAt获取字符串第一个字母然后比较，但马上我想到有可能几个字符串它们前几个字母都是相同的，所以单纯取一个字母是不可以的。于是我想到再使用循环获取，再加上长度的判断，但这未免过于冗长，于是我发现了compareTo（借助了科技的力量！）太完美了，直接完美解决我的问题，至此这个小任务算是完成。

加入对象

代码如下：

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

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

@Data
@AllArgsConstructor
// 这里的注解代表类的构造器（全参）以及所有数据的get和
// set函数,建议自己写代码实现构造器和get
@NoArgsConstructor
// 略略略，手写是不可能滴
public class Song{
    private String title;
    private String artist;
    private int bpm;

    public static void main (String[] args){

        List<Song> songs = MockMusics.getSongObjects();

        // 以歌曲名字的字母顺序排序
        //BubbleSort(songs); // 仍然可以使用冒泡排序
        songs.sort(Comparator.comparing(Song::getTitle));

        System.out.println(songs); // toString方法
        songs.forEach(song -> System.out.print(song));// lambda表达式
        songs.forEach(System.out::print); //
    }
```

```
    public static void BubbleSort(List<Song> songs) {
        // 冒泡算法中的变量，用于判断是否排序完成
        boolean swapped = false;

        // 最多冒泡n-1次
        for (int i = 0; i < songs.size() - 1; i++) {
            swapped = false;
            //每次排序参与的元素依次减少一
            for (int j = 0; j < songs.size() - i - 1; j++) {
                //判断j与j+1号歌曲名字的字母顺序
                if (songs.get(j).getTitle().compareTo(songs.get(j+1).getTitle())
> 0) {
                    // 将j与j+1的元素互换
                    Song temp = songs.get(j);
                    songs.set(j, songs.get(j + 1));
                    songs.set(j + 1, temp);
                    swapped = true; // 说明排序没有结束，防止退出循环
                }
            }

            //此时集合排序已经完成，退出循环
            if (!swapped) {break;}
        }
    }

}

class MockMusics {
    public static List<Song> getSongObjects() {
        List<Song> songs = new ArrayList<>();
        // 模拟将要处理的列表
        songs.add(new Song("sunrise", "Artist A", 120));
        songs.add(new Song("noprice", "Artist B", 130));
        songs.add(new Song("thanks", "Artist C", 110));
        songs.add(new Song("havana", "Artist D", 140));
        songs.add(new Song("114514", "Artist E", 100));
        return songs;
    }
}
```

笔记：因为题目的表述不是很清楚，我暂且理解为对Song类型的集合以title的自然字母排序。那么我首先想到的是这仍然可以冒泡解决。我最初的思路是将每个Song的title提取出来放在一个List中，然后可以使用上个任务所写的算法，唯一需要改变的是要同时改变Song_List中的元素，这也很容易实现。但是代码会显得很繁琐，于是我意识到完全不需要吧title全部抽出来，只需要在比较时调用从对应的Song类中调取即可，于是就变成了如上代码展示。

此外查询获知Comparator接口的comparing方法可以对字符串以字母的自然顺序排序，于是先使用lambda表达式写

```
songs.sort((song1, song2) -> song1.getTitle().compareTo(song2.getTitle()));
```

idea 提示可以修改为上述代码中的那一行，我探索了一番后有点明白但是有点云里雾里，可能这需要在以后的学习中再加强了。

最后的输出我主要是巩固为主，@Data重写了toString方法所以直接打印对象就可以输出；再用lambda遍历输出，idea有提示可以简化为下行的代码。在这里我隐约感觉到这是在直接调用类的方法而省去了创建实例啥的吗？？有待考证

哦还有个小插曲，我在获取List时意外写在了方法外面，导致我底下构造对象的那几行直接红温，我看那个警告信息说四个参数只传了三个还感到莫名其妙，哪来的四个。原来那个@AllArgsConstructor把那一行也算成参数给到构造器的形参了...