Java07

Task1

Q1

集合接口以及实现类各自的功能

- **Collection**是Java集合框架中最顶层的接口之一，它定义了所有单列集合（即每个元素都是一个独立对象，如 List, Set, Queue）所共有的、最核心的操作。其**核心功能**包括 添加元素 、 删除元素 、 查询和检查 、 遍历集合 、 转换为数组。
- **List**是 Collection 接口下一个非常重要且常用的子接口，它的核心特点是**有序**（元素存入的顺序和取出的顺序是一致的）、**可重复**（允许存放相同的元素）、**带索引**（每个元素都有一个整数索引（从0开始），这是List区别于其他Collection（如Set）的最主要特征）。其**操作**包括 按索引取元素 、 增加或插入元素 、 按索引或对象删元素 、 修改指定位置的元素 、 查找元素首次出现的位置。
- **ArrayList**是 List 接口最常用的一种实现，它的本质是一个 **动态增长的数组**。由于它实现了 List 接口，所以拥有所有List的核心方法。
- **LinkedList**是 List 接口的另一个重要实现，它的本质是一个 **双向链表**，即每个元素（节点）都保存着自身的数据以及指向前一个和后一个元素的指引。除了继承自 List 的核心功能外，它还实现了 Deque（双端队列）接口，提供了一系列在头部和尾部进行操作的高效方法，因此 LinkedList 可以被用作 栈(Stack)、队列(Queue) 或 双端队列(Deque)。
- **Set**是 Collection 接口下一个非常重要的子接口，它的核心特点是 **不可重复**（集合中不允许包含相同的元素） 、 **最多一个null元素**（大多数Set实现允许包含一个null元素） 、 **无序性**（大多数实现（如HashSet）不保证元素存入和取出的顺序一致，不过也有专门保证顺序的实现（如LinkedHashSet, TreeSet））。其**功能**包括 添加元素 、 删除元素 、 查询和检查（效率通常非常高） 、 批量操作（判断集合包含关系以及求交集、并集、差集） 、 遍历。其**用途**主要有三个： 去重（例如List转换成Set可以轻松去除所有重复项） 、 快速成员检查（判断某个元素是否存在于一个大型集合中） 、 数学集合运算。
- **HashSet**是 Set 接口最常用、最经典的实现类。它的核心功能就是**基于哈希表**（Hash Table） 来实现一个**无序、不可重复的集合**。
- **TreeSet**是 Set 接口的一个实现类，它的核心特点是 **自动排序** 和 **不可重复**(基于一种叫做**红黑树**的自平衡二叉查找树)。
- **Map**是Java集合框架中一个与 Collection 接口并列的**顶级接口**。它的核心功能是存储 **键值对** 。其**核心特点**包括 双列存储 、 键唯一，值可重复 、一个null键（大多数Map实现允许一个null键和多个null值） 、 通过键操作值。其**核心功能**包括 添加和修改元素 、 删除元素 、查询和检查 、获取视图（可以分别获取其键、值或键值对的集合视图） 、 遍历（通过视图遍历） 。 其**核心用途**包括 高效键值查找 、 实现映射关系 、 元素计数。
- **HashMap**是 Map 接口最常用、最经典的实现类。它的核心功能是**基于哈希表**（Hash Table） 来存储键值对（Key-Value Pairs），并提供极快的查找速度。其核心特点包括 无序性（不保证元素的顺序） 、 允许Null键和Null值 。其适用于绝大多数需要快速通过一个键来查找值的场景。
- **TreeMap**是 Map 接口的一个实现类，它的核心特点是 **按键排序**（基于红黑树）。除了继承自 Map 的核心功能外，它还提供了一系列用于访问“第一个”、“最后一个”、以及某个“范围”内映射关系的方法（因为它是有序的）。

数组与集合的区别

相同点：

- 都是容器：两者都可以用来存储一组数据（对象或基本类型）。
- 都可以通过索引/键进行访问：例如，数组通过 array[index]，List 集合通过 list.get(index)。

不同点：

特性	数组 (Array)	集合 (Collection)
大小	固定长度。一旦创建，容量就不能改变。	动态大小。大多数集合类可以根据需要自动扩容。
能存储的数据类型	既可以存储基本数据类型（如 int, char），也可以存储对象。	只能存储对象（引用类型）。如果要存储基本类型，需要使用其对应的包装类（如 Integer, Character）。
功能丰富性	功能非常基础，只提供了长度（length）属性和基本的存取操作。	功能极其丰富。提供了大量的现成方法，如排序、搜索、插入、删除、过滤、遍历（迭代器）等。
底层实现	是Java语言中最基础、最高效的线性存储结构。	提供了多种多样的数据结构实现，如链表（LinkedList）、哈希表（HashMap）、树（TreeSet）等，以适应不同的场景。
存储特点	存储有序、可重复的元素。	提供了多种存储特点：List: 有序，可重复。 Set: 无序（或有序），不可重复。 Queue: 队列，先进先出。 Map: 键值对存储。

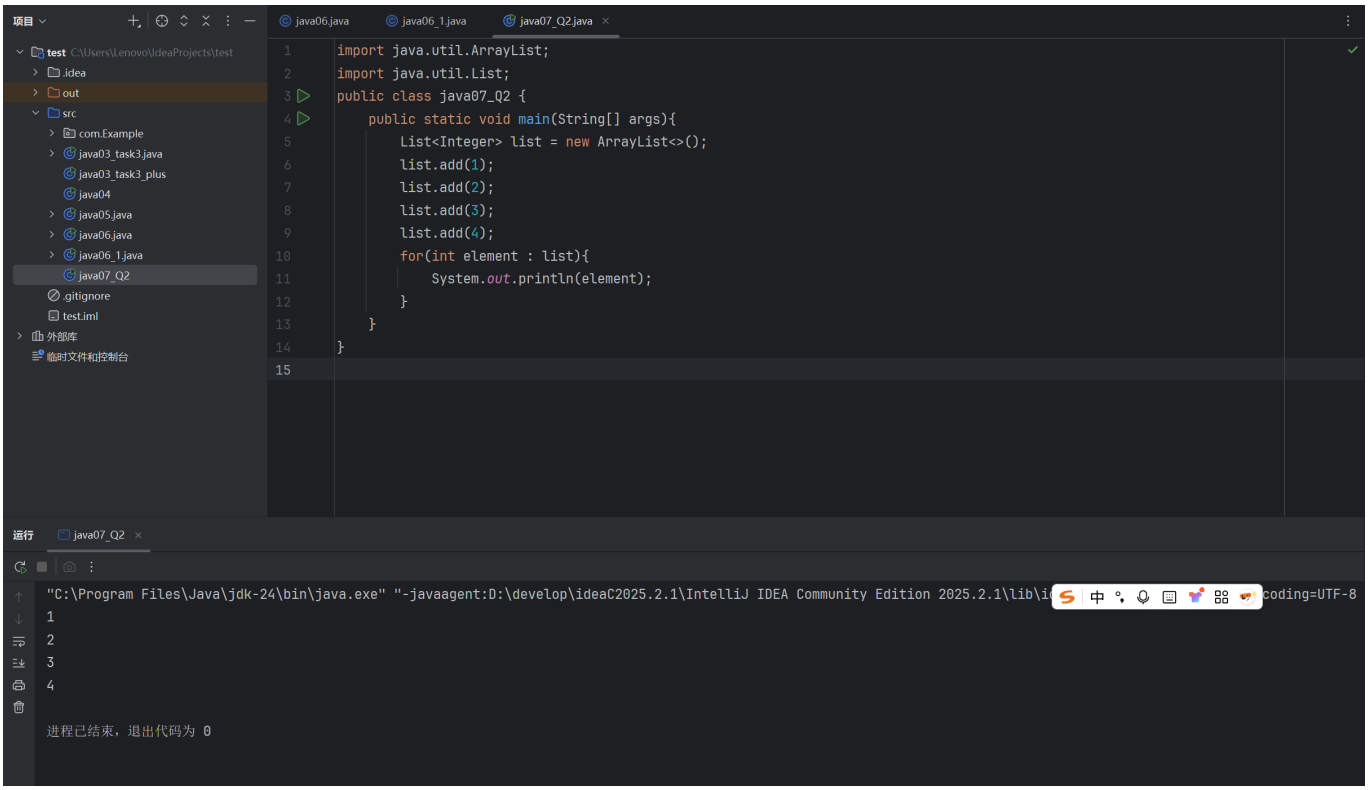
Task2

Q2

代码

```
import java.util.ArrayList;
import java.util.List;
public class java07_Q2 {
    public static void main(String[] args){
        List<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        for(int element : list){
            System.out.println(element);
        }
    }
}
```

运行结果截图



Q3

什么是匿名内部类

匿名内部类就是"即定义即用"的类——一个在创建对象的同时定义这个对象的类，不需要写出类名。以List的foreach方法举例，如果不使用匿名内部类，那么就需要一系列繁琐的流程：定义一个实现了Consumer接口的新的类，在该类中重写accept方法，然后为该类创建一个对象（实例），最后再将该对象作为参数传入foreach方法中。匿名内部类简化了这一流程。

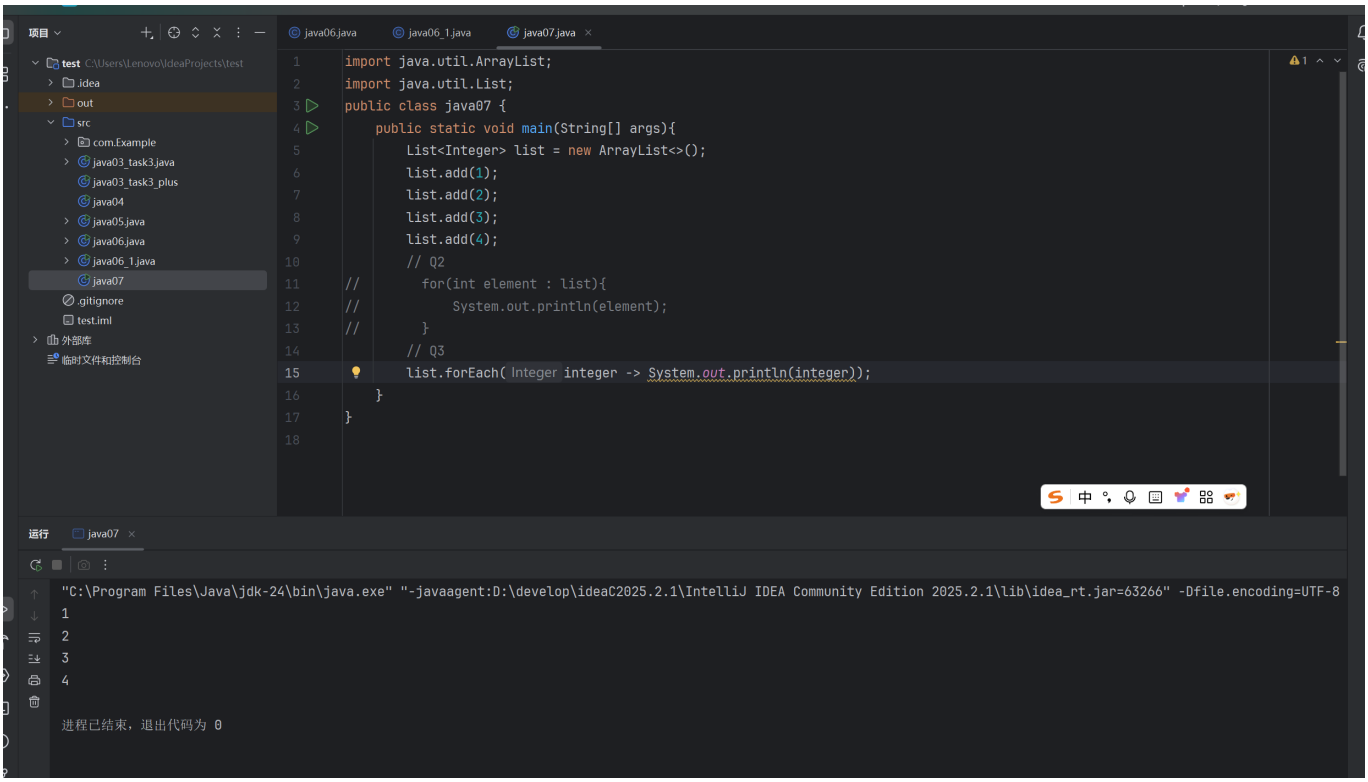
什么是函数式接口

函数式接口是只有一个抽象方法的接口，通过这一特性，编译器就无需辨别调用的是接口中的哪个方法（对比接口中定义了多种抽象方法的情况），使得其能支持Lambda，简化代码表达。这样明确的输入输出关系让该类接口类似数学上的函数。

lambda表达式改写遍历代码

```
list.forEach(integer -> System.out.println(integer));
```

运行结果截图



常见的函数式接口

- `Consumer<T>`包含方法void accept(T t)，接受一个参数t，执行操作
- `Supplier<T>`包含方法T get()，无参数，返回一个T类型的结果
- `Function<T,R>`包含方法R apply(T t)，接受T类型的t参数，返回R类型结果（注：R代表"返回类型"，它可以是任何引用类型（类、接口、数组等），但不能是基本类型（如int,double等））
- `Predicate<T>`包含方法boolean test(T t)，接受T类型的参数t，返回值为boolean类型
- `Runnable`包含方法void run()，无参数无返回的操作
- `Comparator<T>`包含方法int compare(T o1, T o2)，用于比较两个对象

lambda表达式的用法的总结

```
//Lambda表达式的基本格式:
//(参数) -> { 表达式或代码块 }
//注：单个参数时可以省略圆括号，表达式或代码块只有一条语句时可以省略花括号和return关键字。

//用法:
// 1. 无参数，无返回值
Runnable task = () -> System.out.println("HelloWorld");
// 2. 一个参数
Consumer<String> printer = msg -> System.out.println(msg);
// 3. 多个参数
Comparator<Integer> comparator = (a, b) -> a - b;
// 4. 带类型声明
Comparator<Integer> typed = (Integer a, Integer b) -> a.compareTo(b);
// 5. 多行代码块
Function<String, String> processor = str -> {
    String trimmed = str.trim();
```

```
    return trimmed.toUpperCase();
};

//一个常见场合：排序
List<String> words = Arrays.asList("banana", "apple", "cherry", "date");
    // 按字母顺序排序
words.sort((s1, s2) -> s1.compareTo(s2));
    // 按字符串长度排序
words.sort((s1, s2) -> s1.length() - s2.length());
```

Task3

Q4

代码：

```
//Main.java
package java04_Q4;
import java04_Q4.User;
import java04_Q4.MyRepository;
public class Main {
    public static void main(String[] args){
        MyRepository<String> stringMyRepository=new MyRepository<>();
        MyRepository<User> userMyRepository=new MyRepository<>();
        MyRepository<Integer> integerMyRepository=new MyRepository<>();
        //储存String User Integer类型的三组数据：
        stringMyRepository.save("stringTestData1");
        stringMyRepository.save("stringTestData2");
        stringMyRepository.save("stringTestData3");
        userMyRepository.save(new User("userTestData1",11));
        userMyRepository.save(new User("userTestData2",22));
        userMyRepository.save(new User("userTestData3",33));
        integerMyRepository.save(1111);
        integerMyRepository.save(2222);
        integerMyRepository.save(3333);
        //打印仓库的所有内容：
        System.out.println("string:");
        stringMyRepository.printAll();
        System.out.println("user:");
        userMyRepository.printAll();
        System.out.println("integer:");
        integerMyRepository.printAll();
        //根据ID获取数据：
        for(int i=0;i<3;i++){
            System.out.println("ID="+i+"\n "+stringMyRepository.getById(i)+"\n
"+userMyRepository.getById(i)+"\n "+integerMyRepository.getById(i));
        }
    }
}
```

```java
//Repository,java
package java04_Q4;
public interface Repository <T>{
    void save(T item);
    T getById(int id);
}
```

```java
//MyRepository.java
package java04_Q4;
import java.util.HashMap;
import java.util.Map;
public class MyRepository<T> implements Repository<T> {
    private Map<Integer,T> storage;
    private int count;
    public MyRepository(){
        this.storage=new HashMap<>();
        this.count=0;
    }
    @Override
    public void save(T item){
        storage.put(count,item);
        count++;
    }
    public T getById(int id){
        return storage.get(id);
    }
    public void printAll(){
        for(Map.Entry<Integer,T> entry : storage.entrySet()){
            System.out.println("ID:"+entry.getKey()+" ITEM:"+entry.getValue());
        }
        System.out.println("END");
    }
}
```

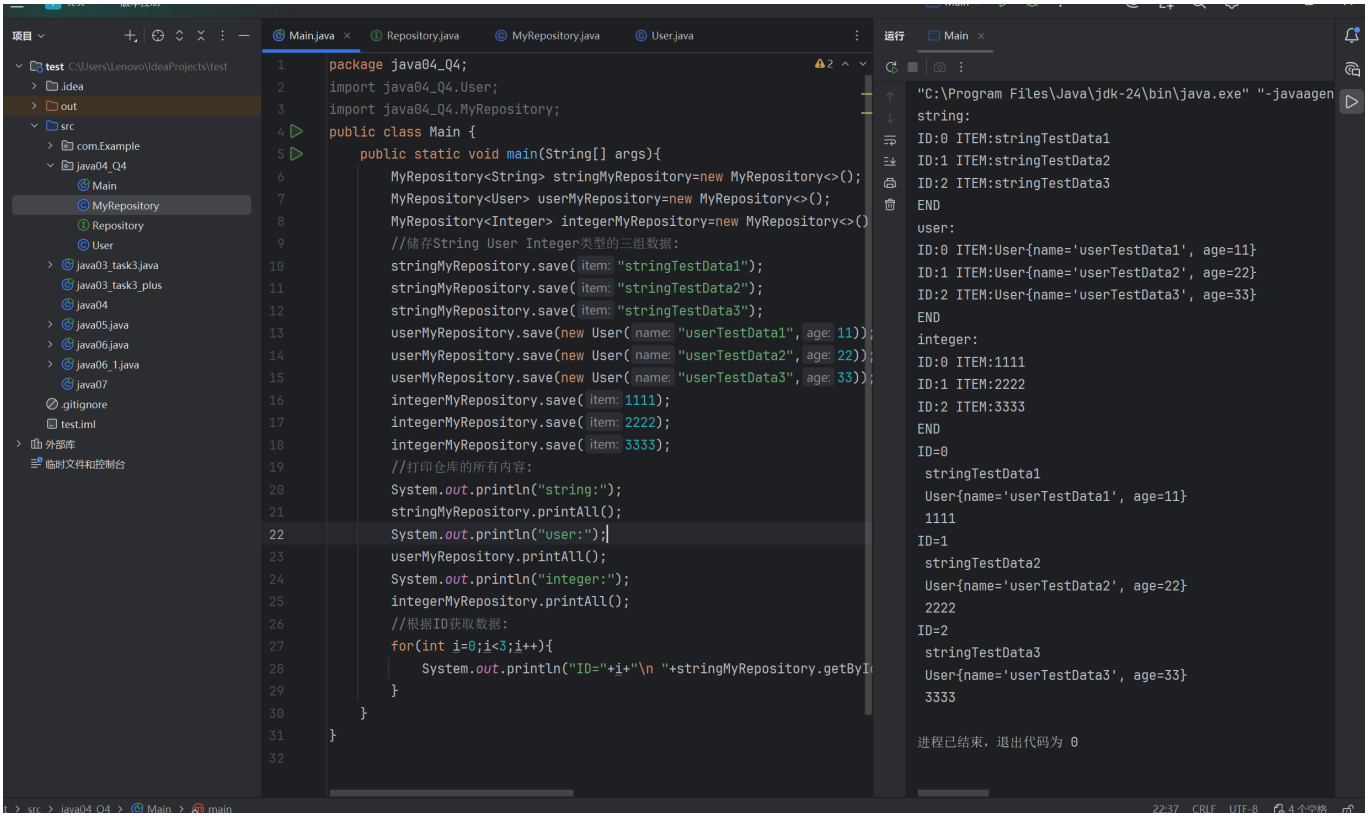
```java
//User.java
package java04_Q4;
public class User {
    private String name;
    private int age;

    public User(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
```

```
        return "User{name='" + name + "', age=" + age + "}";
    }
}
```

运行结果截图：



Task4

Q5

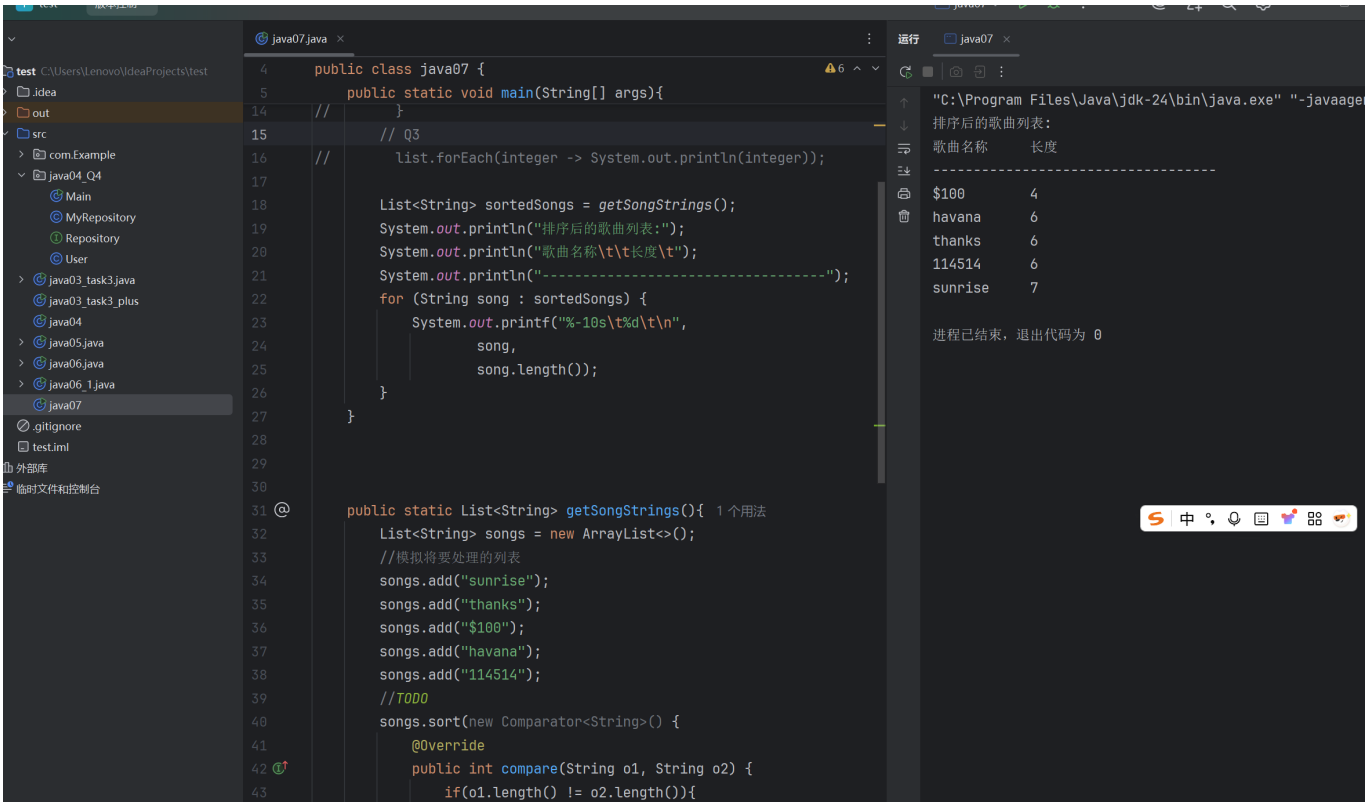
代码：

```
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("thanks");
        songs.add("$100");
        songs.add("havana");
        songs.add("114514");
        //TODO

        songs.sort(new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                if(o1.length() != o2.length()){
```

```
                    return o1.length()-o2.length();
                }
                for(int i=0;i<o1.length();i++){
                    char c1=o1.charAt(i) ,c2=o2.charAt(i);
                    if(Character.isLetter(c1)){
                        if(Character.isLetter(c2)){
                            continue;
                        }else{
                            return -1;
                        }
                    }else if(Character.isDigit(c1)){
                        if(Character.isLetter(c2)){
                            return 1;
                        }else if(Character.isDigit(c2)){
                            continue;
                        }else{
                            return -1;
                        }
                    }else{
                        if(Character.isDigit(c2)||Character.isLetter(c2)){
                            return 1;
                        }else{
                            continue;
                        }
                    }
                }
                return o1.compareTo(o2);
            }
        });

        //END
        return songs;
    }
}
```


test C:\Users\Lenovo\IdeaProjects\test
.idea
out
src
com.Example
java04_Q4
Main
MyRepository
Repository
User
java03_task3.java
java03_task3_plus
java04
java05.java
java06.java
java06_1.java
java07
.gitignore
test.iml
外部库
临时文件和控制台
java07.java
public class java07 {
public static void main(String[] args){
// }
// Q3
// list.forEach(integer -> System.out.println(integer));
List<String> sortedSongs = getSongStrings();
System.out.println("排序后的歌曲列表:");
System.out.println("歌曲名称\t\t长度\t");
System.out.println("----------");
for (String song : sortedSongs) {
System.out.printf("%-10s\t%d\t\n",
song,
song.length());
}
}
public static List<String> getSongStrings(){ 1 个用法
List<String> songs = new ArrayList<>();
//模拟将要处理的列表
songs.add("sunrise");
songs.add("thanks");
songs.add("$100");
songs.add("havana");
songs.add("114514");
//TODO
songs.sort(new Comparator<String>() {
@Override
public int compare(String o1, String o2) {
if(o1.length() != o2.length()){
运行
java07
"C:\Program Files\Java\jdk-24\bin\java.exe" "-javaage
排序后的歌曲列表:
歌曲名称 长度
$100 4
havana 6
thanks 6
114514 6
sunrise 7
进程已结束，退出代码为 0