

# Q1

---

## 接口

Collection：所有集合的根接口，定义了基本操作如添加、删除、遍历等

List：有序集合，允许重复元素，可通过索引访问

Set：不允许重复元素的集合

Map：存储键值对的结构，键唯一，值可重复

## 实现类

ArrayList：基于动态数组的List实现，查询快，增删慢

LinkedList：基于链表的List实现，增删快，查询慢

HashSet：基于哈希表的Set实现，无序但查找效率高

TreeSet：基于红黑树的Set实现，自动排序

HashMap：基于哈希表的Map实现，键值对存储

TreeMap：基于红黑树的Map实现，按键排序

数组与集合的相同点和不同点：

### 相同点：

都用于存储多个数据元素

都可以存储引用数据类型

都可以通过某种方式访问元素

不同点：

集合长度是动态的，可以自动扩容；但是数组的长度是固定的，需要手动扩容很麻烦

集合只能存储引用类型，数组可以存储基本数据类型和引用类型

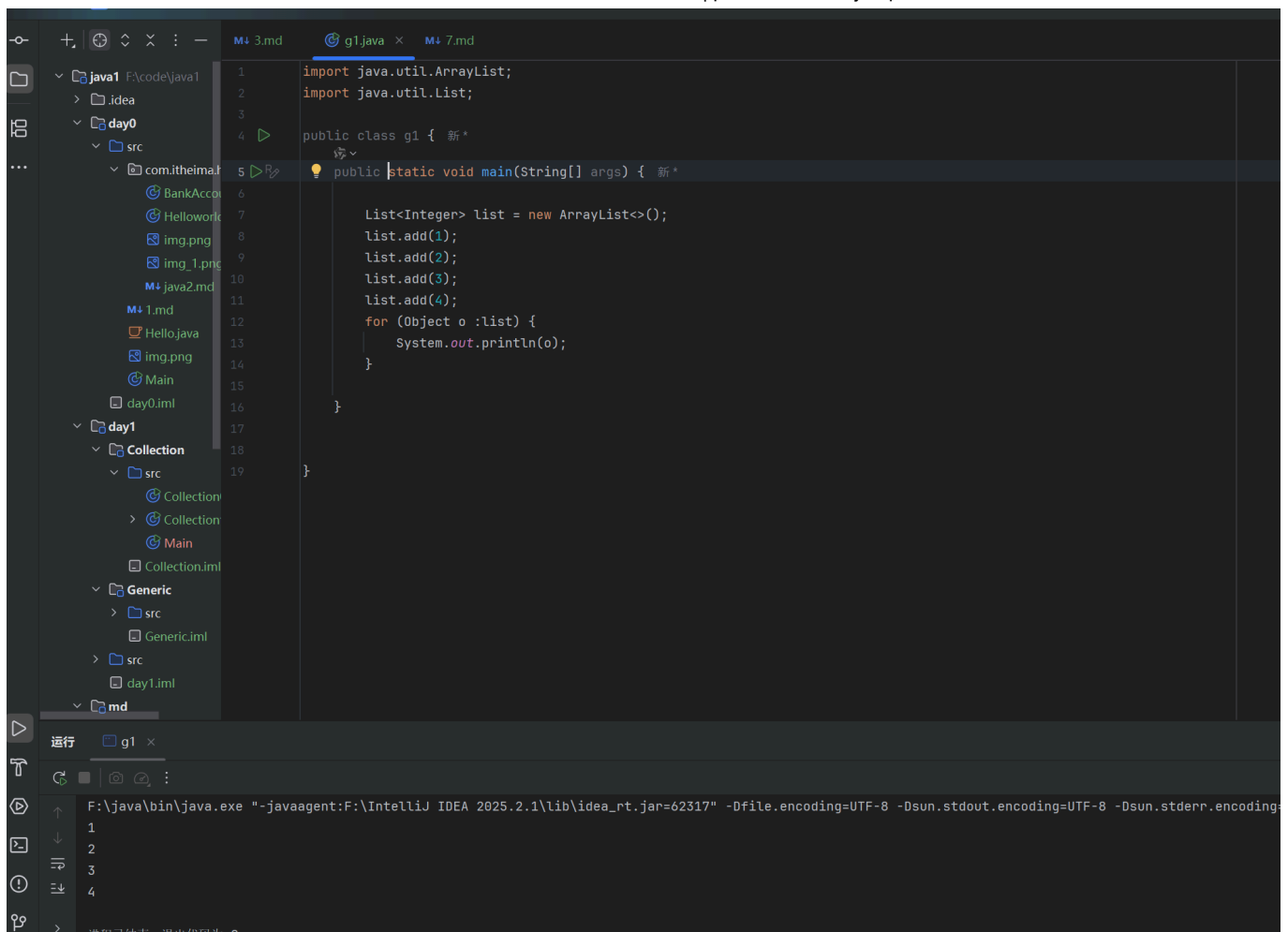
集合提供了丰富的操作方法（如增删查改：排序）数组只提供基础的存储功能

### 观点：

我认为集合相较于数组具有更好的封装性，更多的方法：更灵活的大小，使用起来更简便实用

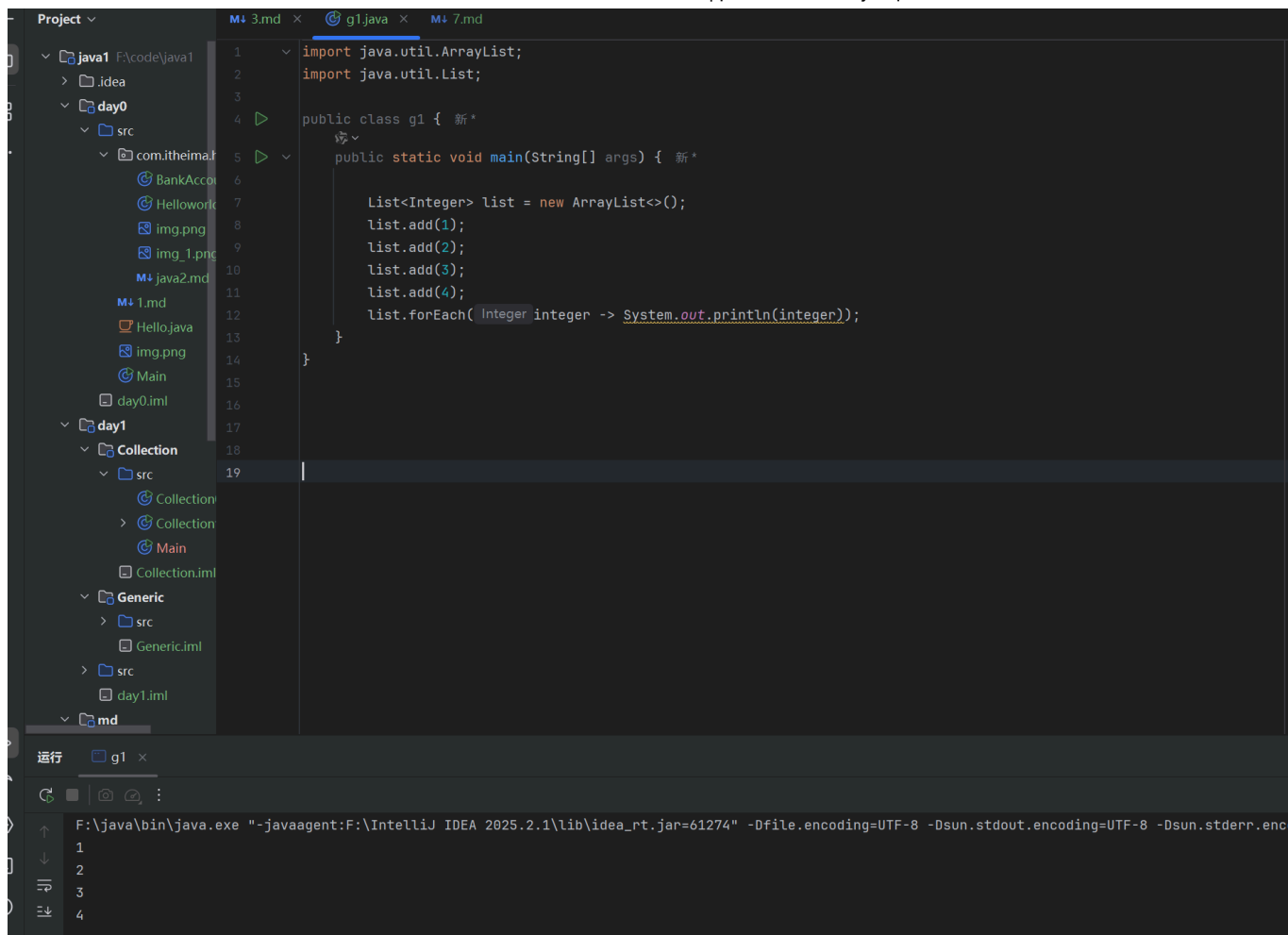
# Q2

---



## Q3

匿名内部类是没有名字的类，通常用于只需使用一次的类创建，可以使程序更简洁更容易维护  
函数式接口是只包含一个抽象方法的接口，用于支持Lambda表达式



## Lambda表达式的用法总结:

基本语法结构

标准形式: (参数列表) -> { 方法体 }

简化形式:

单个参数可省略括号: 参数 -> {方法体 }

单行语句可省略大括号: (参数) -> 方法调用

单行返回语句可省略return关键字: (参数) -> 表达式

使用条件:

只能用于实现函数式接口（只包含一个抽象方法的接口）

常见应用场景

集合遍历

事件处理

比较

线程创建

## Q4:

### Repository接口:

```
1. public interface Repository <E>{
```

```
2.     void save(E Data);
3.     E getById(int id);
4.
5. }
```

## MyRepository类

```
1. import java.util.HashMap;
2. import java.util.Map;
3. public class MyRepository<E> implements Repository<E>{
4.     private Map<Integer,E> storage = new HashMap<>();
5.     private int Id=0;
6.
7.     @Override
8.     public void save(E Data) {
9.         storage.put(Id,Data);
10.        Id++;
11.
12.    }
13.    @Override
14.    public E getById(int id) {
15.        return storage.get(id);
16.    }
17.    public void print(){
18.        for (Map.Entry<Integer, E> entry : storage.entrySet()) {
19.            System.out.println("ID " + entry.getKey() + ": " + entry.getValue());
20.        }
21.    }
22. }
```

## User类

```
1. public class User {
2.     private String name;
3.     private int age;
4.
5.     public User(String name, int age) {
6.         this.name = name;
7.         this.age = age;
8.     }
9.
10.    @Override
11.    public String toString() {
```

```
12.         return "User{name='" + name + "', age=" + age + "}";
13.     }
14. }
```

## Use测试类:

```
1. public class Use {
2.     public static void main(String[] args) {
3.         MyRepository<String> storage1 = new MyRepository<>();
4.         storage1.save("Hello");
5.         storage1.save("World");
6.         storage1.save("Java");
7.         System.out.println("storage1:");
8.         storage1.print();
9.
10.
11.         MyRepository<Integer> storage2 = new MyRepository<>();
12.         storage2.save(100);
13.         storage2.save(200);
14.         storage2.save(300);
15.         System.out.println("storage2");
16.         storage2.print();
17.
18.
19.
20.
21.
22.         MyRepository<User> storage3 = new MyRepository<>();
23.         storage3.save(new User("Alice", 25));
24.         storage3.save(new User("Bob", 30));
25.         storage3.save(new User("Charlie", 35));
26.         System.out.println("storage3");
27.         storage3.print();
28.
29.     }
30. }
```

## 运行截图

```
F:\java\bin\java.exe "-javaagent:F:\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=62063" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath F
storage1:
ID 0: Hello
ID 1: World
ID 2: Java
storage2:
ID 0: 100
ID 1: 200
ID 2: 300
storage3:
ID 0: User{name='Alice', age=25}
ID 1: User{name='Bob', age=30}
ID 2: User{name='Charlie', age=35}

进程已结束，退出代码为 0
```

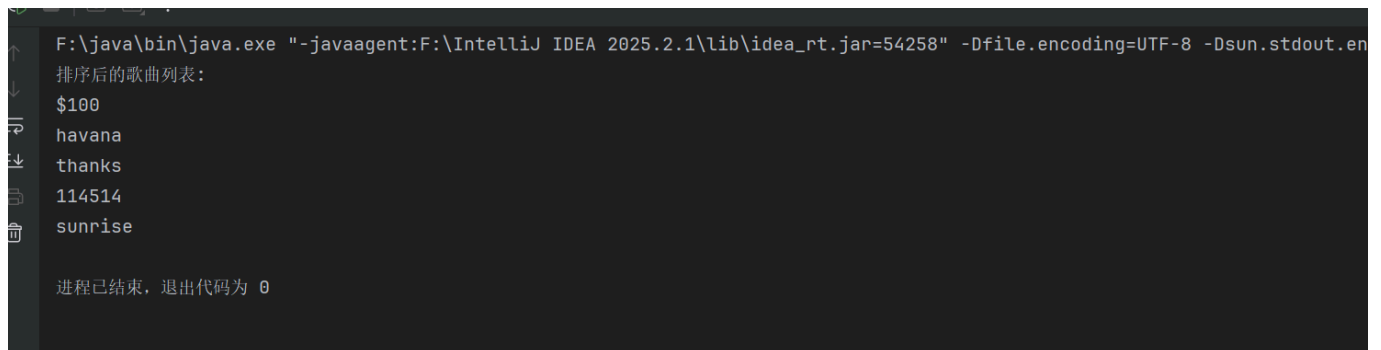
## Q5:

### 代码:

```
1. import java.util.ArrayList;
2. import java.util.Collections;
3. import java.util.List;
4.
5. public class MockSongs {
6.     public static List<String> getSongStrings(){
7.         //模拟将要处理的列表
8.         List<String> songs = new ArrayList<>();
9.         songs.add("sunrise");
10.        songs.add("thanks");
11.        songs.add("$100");
12.        songs.add("havana");
13.        songs.add("114514");
14.        //TODO
15.        //在这里完成你的代码
16.        Collections.sort(songs, (s1, s2) -> {
17.            // 首先按字符串长度排序
18.            if (s1.length() != s2.length()) {
19.                return s1.length() - s2.length();
20.            }
21.            // 长度相同时，按字符类型排序：字母 < 数字 < 其他符号
22.            char c1 = s1.charAt(0);
23.            char c2 = s2.charAt(0);
24.            int type1 = getCharType(c1);
25.            int type2 = getCharType(c2);
26.
27.            if (type1 != type2) {
28.                return type1 - type2;
29.            }
30.            // 类型相同则按字典序排序
31.            return s1.compareTo(s2);
```

```
32.         });
33.         //END
34.         return songs;
35.     }
36.     private static int getCharType(char c) {
37.         if (Character.isLetter(c)) {
38.             return 0;
39.         } else if (Character.isDigit(c)) {
40.             return 1;
41.         } else {
42.             return 2;
43.         }
44.     }
45.     // 添加测试方法
46.     public static void main(String[] args) {
47.         List<String> Songs = getSongs();
48.         System.out.println("排序后的歌曲列表:");
49.         for (String song : Songs) {
50.             System.out.println(song);
51.         }
52.     }
53. }
```

运行截图:



```
F:\java\bin\java.exe "-javaagent:F:\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=54258" -Dfile.encoding=UTF-8 -Dsun.stdout.en
排序后的歌曲列表:
$100
havana
thanks
114514
sunrise

进程已结束，退出代码为 0
```