1.List

1. Collection的常用子接口
2. List的常用子类：ArrayList、LinkedList、Vecotr
3. List常用子类的区别、特征

2.Set

常用子类： HashSet、TreeSet

2.1HashSet

基本使用

```
/**
 * 作者: jack
 * 时间: 2021-04-28 0028 08:48
 * 描述: Test
 * 特征:
 * 底层是hashcode
 * 如何判定是重复元素： 1. 比较hashcode,如果不同则保存  2. 如果hashcode相同则调用 equals 方法
 */
public class Test {

    public static void main(String[] args) {

        HashSet<String> set = new HashSet<>();

        // 添加元素
        set.add("A");
        set.add(null);
        set.add("admin");
        set.add("jack");
        set.add("admin");
        set.add("123");

        // 大小
        int size = set.size();
        System.out.println(size);

        // set 没有 get(index); 不能查询指定的元素

        set.remove("A"); // 通过值进行删除
        System.out.println(set.size());

        System.out.println("----------------------");
        // 遍历 ：  fori 循环????
        for (String e : set) {
            System.out.println(e);
        }
```

```
        System.out.println("----------------------");
        // 迭代
        Iterator<String> its = set.iterator();
        while (its.hasNext()) {
            System.out.println(its.next());
        }


    }

}
```

hashset保存对象

```
/**
 * 作者: jack
 * 时间: 2021-04-28 0028 09:02
 * 描述: User
 */
public class User {
    private Integer id;
    private String name;


    public User() {
    }

    public User(Integer id, String name) {
        this.id = id;
        this.name = name;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
```

```
    // 将次方法注释调用，让hashset调用Object的hashCode方法。则hashCode不同，不会调用
equals
    @Override
    public int hashCode() {
        System.out.println("hashCode()");
        return 1;
    }

    @Override
    public boolean equals(Object obj) {
        System.out.println("equals(Object obj)");
        return true;
    }
}
```

```
public class HashSetUser {
    public static void main(String[] args) {

        HashSet<User> users = new HashSet<>();

        users.add(new User(1, "admin"));
        users.add(new User(2, "jack"));

        System.out.println(users.size());
        // 简单的输出方式
        System.out.println(users.toString());

    }
}
```

2.2TreeSet

基本使用

```
/**
 * 作者: jack
 * 时间: 2021-04-28 0028 09:11
 * 描述: Test
 * TreeSet: 元素唯一, 底层是二叉树 (自然有序) 不能有 null
 */
public class Test {

    public static void main(String[] args) {

        TreeSet<String> set = new TreeSet<>();

        // compareTo()  比较大小
        set.add("D");
        set.add("C");
        set.add("A");
        set.add("B");
        set.add("D");

        // set.add(null); // null.compareTo() => NullPointException

        Iterator<String> its = set.iterator();
```

```
        while (its.hasNext()) {
            System.out.println(its.next());
        }


    }

}
```

TreeSet保存对象：

写法1： 让对象类实现Comparable接口

```
public class User implements Comparable<User> {
    private Integer id;
    private String name;


    public User() {
    }

    public User(Integer id, String name) {
        this.id = id;
        this.name = name;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

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

    @Override
    public int compareTo(User o) {
        // 自定义比较规则
        if (this.id > o.id) return 1;
        if (this.id < o.id) return -1;
        return this.name.compareTo(o.name);
```

```
    }
}
```

```
public class TreeSetUser {
    public static void main(String[] args) {

        TreeSet<User> treeSet = new TreeSet<>();

        //java.lang.ClassCastException: com.iweb.lesson02.User cannot be cast to
java.lang.Comparable
        treeSet.add(new User(1, "jack"));
        treeSet.add(new User(2, "tom"));

        System.out.println(treeSet);

    }
}
```

写法2：外部比较器 Comparator接口

```
public class UserComparator implements Comparator<User> {
    @Override
    public int compare(User o1, User o2) {
        if (o1.getId() > o2.getId()) return 1;
        if (o1.getId() < o2.getId()) return -1;
        return o1.getName().compareTo(o2.getName());
    }
}
```

```
public class Test {
    public static void main(String[] args) {

        // TreeSet<User> users = new TreeSet<>(new UserComparator());

        // 匿名内部类
        TreeSet<User> users = new TreeSet<>(new Comparator<User>() {
            @Override
            public int compare(User o1, User o2) {
                if (o1.getId() > o2.getId()) return 1;
                if (o1.getId() < o2.getId()) return -1;
                return o1.getName().compareTo(o2.getName());
            }
        });

        users.add(new User(1, "jack"));
        users.add(new User(2, "tom"));

        System.out.println(users);

    }
}
```