一.Map接口下的方法

Map中的元素是以"键值对"(key-Value)的形式出现

key是唯一的,value是可以重复的

```
Map<Integer, String> map = new LinkedHashMap();
```

当key值不存在时,向Map中添加元素(键值对)

```
1001是键,key
"张三是值.value
map.put(1001, "张三");
map.put(1002, "李四");
```

当key值不存在时,修改键对应的值

```
map.put(1002, "李");
map.put(1003, "王麻子");
System.out.println(map);
```

1.判断是否包含某个key

```
System.out.println(map.containsKey(1002));
System.out.println(map.containsKey(1001));
```

2.判断是否包含value

```
System.out.println(map.containsKey("小明"));
System.out.println(map.containsKey("小红"));
```

3.根据key获取value

```
System.out.println(map.get(1001));
System.out.println(map.get(1011));

```

4.获取map中的所有key

```
System.out.println(map.keySet());
```

5.获取map中所有value

```
System.out.println(map.values());
```

6. 获取所有的键值对

```
System.out.println(map.entrySet());
```

7. 根据key删除元素

```
map.remove(1001);
map.remove(1002);
map.remove(1001);
System.out.println(map);
```

8. 元素个数

```
System.out.println(map.size());
```

9. 遍历map

iter

```
for (Integer key : map.keySet()) {
 System.out.printf("%d-%s\n", key, map.get(key));
}
```

```
for (Iterator<Integer> iterator = map.keySet().iterator(); iterator.hasNext(); ) {
 Integer key = iterator.next();
 System.out.printf("%d-%s\n", key, map.get(key));
}
```

```
for (Map.Entry<Integer, String> entry : map.entrySet()) {
 Integer key = entry.getKey();
 String value = entry.getValue();
 System.out.printf("%d-%s\n", key, map.get(key));
}
```

```
for (Iterator<Map.Entry<Integer, String>> iterator = map.entrySet().iterator(); ((Iterator) iterator).hasNext(); ) {
 Map.Entry<Integer, String> entry = iterator.next();
 Integer key = entry.getKey();
 String value = entry.getValue();
 System.out.printf("%d-%s\n", key, map.get(key));
}
```

二. 数据结构 数据在计算机中的存储方式

1.数组:数组在内存中是连续存储的,比如,ArrayList

数组的特点:查找速度块,增,删,慢,需要使用连续空间,大量使用数组这个结构,容易产生内存碎片.

不适合存大量数据

2.链表:在内存中不连续存储,比如LinedList

链表的特点:查找慢,增删快,可以使用不连续 的内存,适合大量数据的数据,双向列表 的查找速度比单向列表快

3.栈:

特点:先进后出,比如:Stack

入栈/压栈:把数据存入栈中

栈分了栈顶和栈底,所有的操作只能在栈顶进行

4.队列:

特点:先进先出.(FIFO)比如:Queue

队列分了对头和对尾,在对尾可以添加数据,在队友可以删除数据

数组排序

```
Arrays.sort(a);
System.out.println(Arrays.toString(a));
Integer[] b={1,2,3,4,4,5,5,5,};
Comparator<Integer> comparator=new MyRule<>();
Comparator<Integer> comparator1=new MyRule<>();
Arrays.sort(b,comparator);
System.out.println(Arrays.toString(b));
```

练习:创建10个元素,用于储存Girl对象

姓名,年龄,升高.随机产生

```
ArrayList arrayList=new ArrayList();
String[] girl1={"木兰1","木兰2","木兰3","木兰4","木兰5","木兰6","木兰7","木兰8","木兰9","木兰10"};
Girl[] girls=new Girl[10];
Random random=new Random();
for (int i = 0; i < girl1.length; i++) {
```

```
6   Girl girl=new
Girl(girl1[i],random.nextInt(3)+21,random.nextInt(20)+160);
7   girls[i]=girl;
8   }
9   System.out.println(Arrays.toString(girls));
10
11   System.out.println("排序后:");
12   Comparator<Girl> comparator2=new GirlRule<>();
13   Arrays.sort(girls,comparator2);
14   System.out.println(Arrays.toString(girls));
```

类中应该写

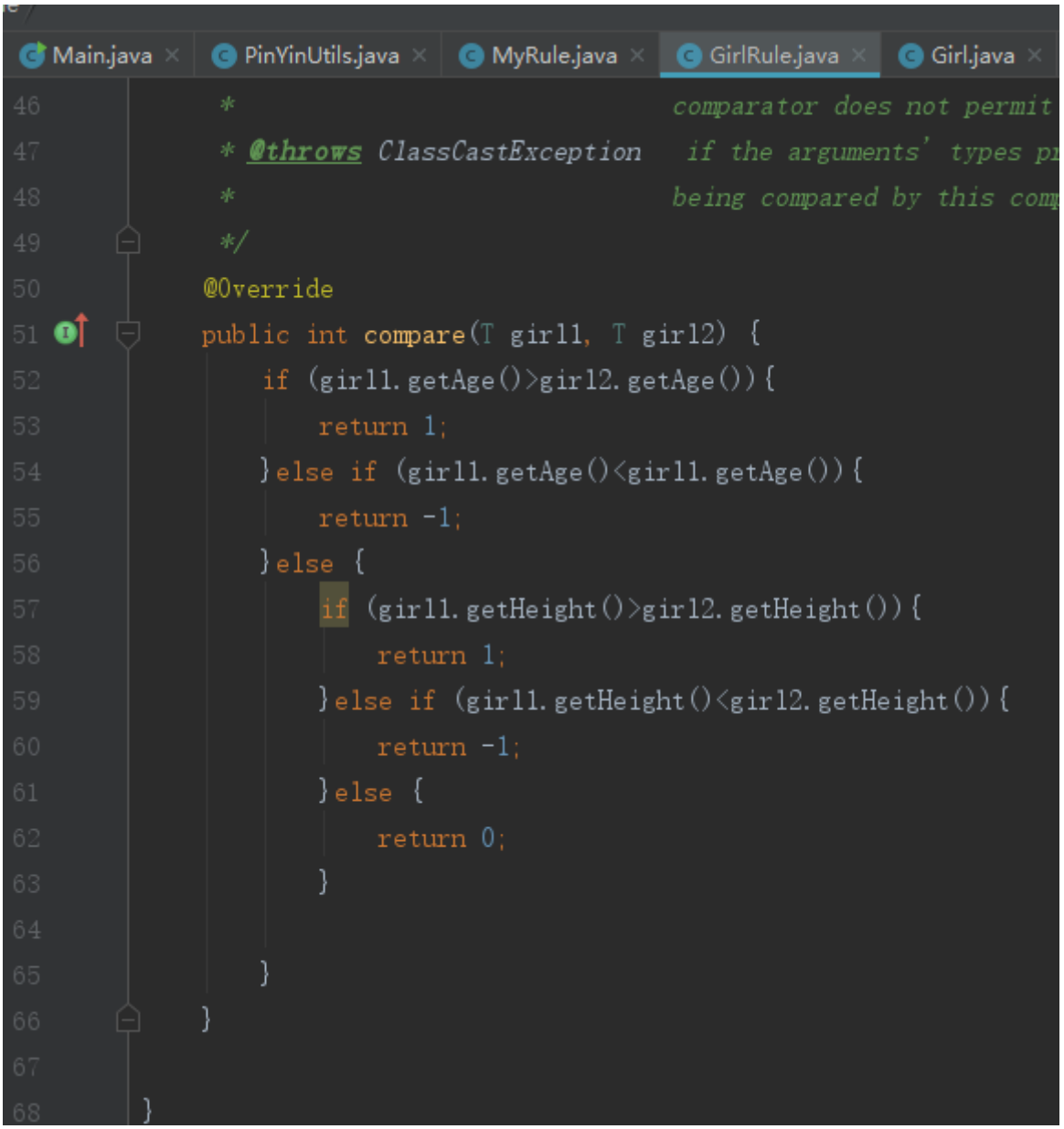
```
1  private String name;
2  private int age;
3  private int height;
4
5  public Girl() {
6  }
7
8  public Girl(String name, int age, int height) {
9   this.name = name;
10   this.age=age;
11   this.height=height;
12  }
13
14  public String getName() {
15   return name;
16  }
17
18  public void setName(String name) {
19   this.name = name;
20  }
21
22  public int getAge() {
23   return age;
24  }
25
26  public void setAge(int age) {
```

```
 this.age = age;
}

public int getHeight() {
 return height;
}

public void setHeight(int height) {
 this.height = height;
}

@Override
public String toString() {
 return "Girl{" +
 "name='" + name + '\'' +
 ", age='" + age + '\'' +
 ", height=" + height +
 '}';
```



二分查找法(拆分查找法)

前提:数组是有序的

```
System.out.println(Arrays.toString(a));
int i=Arrays.binarySearch(a,4);
System.out.println(i);
```

数组转集合

```
List<String>strings=Arrays.asList("哈啊哈","哈哈吗","hahhhahahha");
System.out.println(strings);
```

向数组中填充数据

```
int []c=new int[10];
Arrays.fill(c,66666);
System.out.println(Arrays.toString(c));
```

三.Collections:集合的工具类

```
ArrayList<String>arrayList1=new ArrayList<>();
Collections.addAll(arrayList,"张三","李四","王麻子","李四");
System.out.println(arrayList);
```

1.集合翻转

```
Collections.reverse(arrayList);
System.out.println(arrayList);

Collections.replaceAll(arrayList,"李四","替换");
System.out.println(arrayList);
```

2.生成空的Map,Set,List

```
List<Object>empylist=Collections.emptyList();
Map<Object,Object>emptyMap=Collections.emptyMap();
Set<Object>emptySet=Collections.emptySet();

Set<Integer>set=new HashSet<>();
Collections.addAll(set,13,14,45,3,5,356,456,6567);
System.out.println(set);
//最大值,最小值
Integer max=Collections.max(set);
 System.out.println("max"+max);
 Integer min=Collections.min(set);
 System.out.println("min"+min);
```

3.集合轮换

```
System.out.println(arrayList);
Collections.rotate(arrayList,1);
System.out.println(arrayList);
```

4. 交换集合中的元素

```
Collections.swap(arrayList,0,1);
System.out.println(arrayList);

//打乱集合中的元素
Collections.shuffle(arrayList);
System.out.println(arrayList);

}
```

list中的任意元素

```
static void test(List list) {
System.out.println(list);
}
```

list中的任意元素

```
static void test(List<?> list) {
System.out.println(list);
}
```

list中的元素是GIrl或是Girl是的子类

```
static void test(List<? extends Girl> list) {
System.out.println(list);
}
```

list中的元素是GIrl或是Girl是的父类

```
static void test(List<? super Girl> list) {
System.out.println(list);
}
```