Java

1、List类中的add方法

【基础】`List`接口中的`add`方法有两种重载方式：

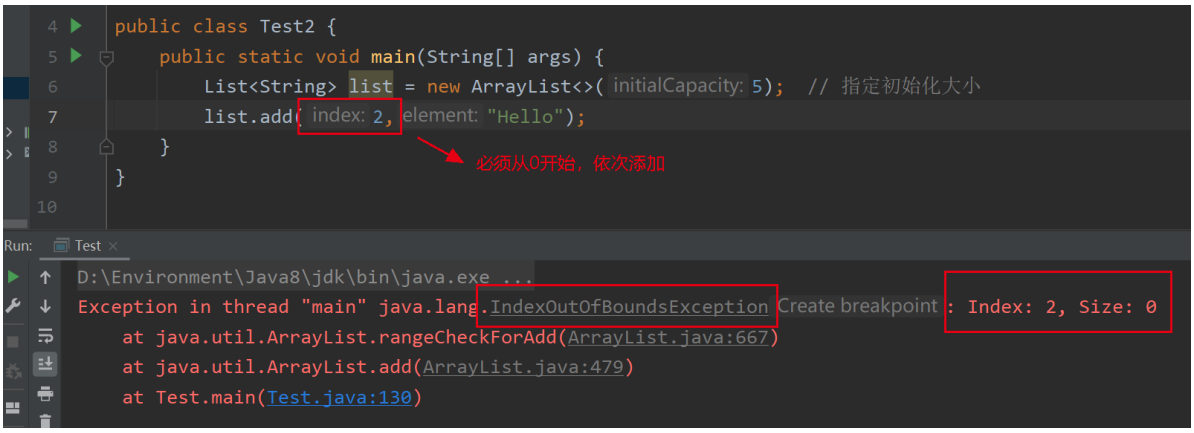
```
boolean add(E e);   // 用于向列表的末尾插入新元素，最常用的插入方法

void add(int index, E element);  // 在插入操作过程中指定插入的位置，此时会自动将当前位置及之后的元素后移
```

【易错】参数`index`的值不可大于当前`list`的容量，即在使用此方法填充一个`list`时，必须以0开始依次填充。

【注意】即使在创建`list`对象时指定了初始化大小，依旧会有问题。

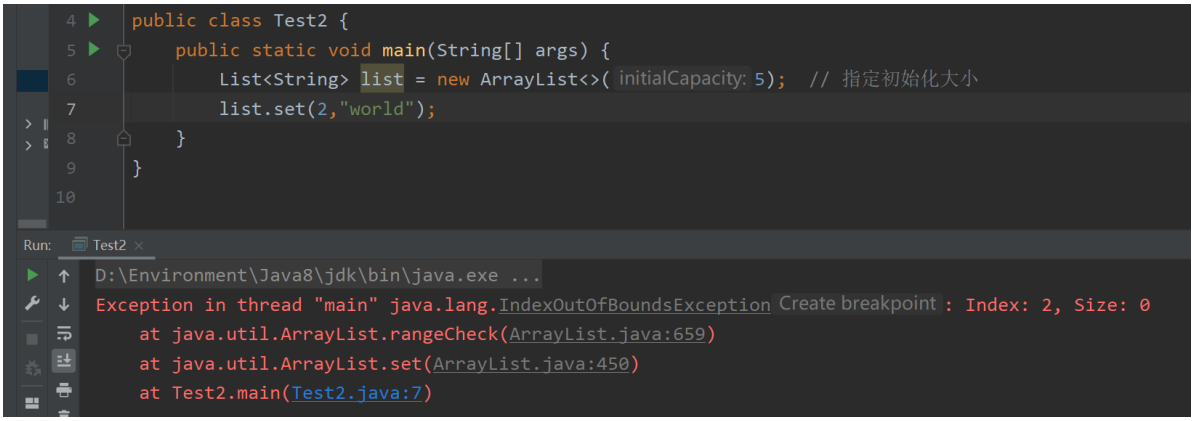
【示例】



2、List类中的set方法

【基础】`set`方法用来设置(替换)`list`中某个位置的元素，本质是一种替换操作。

【易错】要设置某个位置上的元素，这个位置在设置前必须有元素，否则会抛出异常。



【解决】采用预填充的方法先期填充`list`

```
// 填充方法1
List<String> list = new ArrayList<>(5);  // 指定初始化大小
list.add("A");
list.add("B");
list.add("C");
list.add("D");
list.add("E");
list.set(2,"world");

// 填充方法2
List<String> list = new ArrayList<>(Collections.nCopies(5, null)); // [null, null, null, null, null]
```

3、List对象集合中的属性排序

【描述】现有一个存放某种对象的集合`List<Object>`，对象拥有几个属性。现需要按照某个属性按照字母顺序排序。

【注意】对某个集合来说来说，集合中对象所属的类重新的`compareTo`方法，List中谁在前面，这个入参o就是谁。

```
// case1: 按照String属性进行字母排序
public class Student implements Serializable, Comparable<RowInfo> {  // (1)、实现Comparable接口

    private String name;

    // ... 其他属性
    // (2)、重写compareTo方法
    @Override
    public int compareTo(Student o) {
        return name.compareTo(o.name);;
    }
}

// case2: 按照属性nameList进行字母排序，涉及到两层List
public class Student implements Serializable, Comparable<RowInfo> {  // (1)、实现Comparable接口

    private List<String> nameList;
```

```
18
19      // .... 其他属性
20
21      // (2)、重写compareTo方法
22      @Override
23      public int compareTo(Student o) {
24          int compare = 0;
25          for (int i = 0; i < nameList.size(); i++) {
26              compare = nameList.get(i).compareTo(o.nameList().get(i));
27              if(compare < 0){
28                  return compare;
29              }
30          }
31          return compare;
32      }
33  }
```

4、Git中的三种后悔药

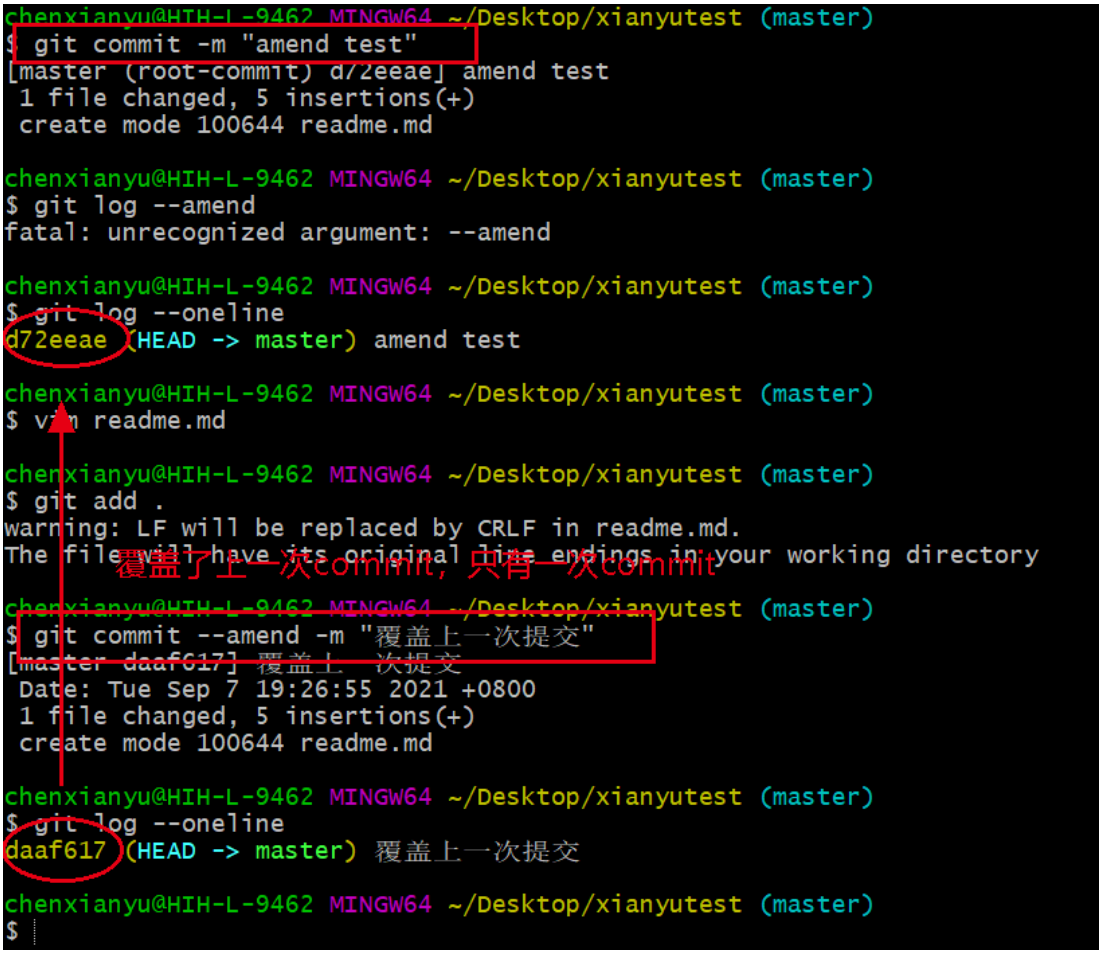
【基础】工作区 -- `git add .` --> 暂存区 -- `git commit -m "msg"` --> 版本库

```
1  git commit --amend -m "msg"  # 新的提交覆盖最近一次的commit内容，即现实项目中某个Task只有一次commit。commitID会变化
2  git revert    # 已经提交了一次新的commit，现在要来回滚到之前的commit
3  git reset xxx # 删除指定的commit
4      git reset soft   # 删除的内容既保存在工作区目录，又保存在暂存区
5      git reset mixed  # （默认）删除的内容保存在工作目录，但是暂存区没有
6      git reset hard   # 删除的内容丢失（不建议使用），工作目录和暂存区都没有
```

4.1、amend : 修正

`git commit --amend -m "msg"`

【注意】覆盖最近一次`commit`后，`commit ID`会变化



4.2、revert : 恢复、还原

【基础】`git revert commitID`

【注意】`revert`后，`commit ID`会变化

4.3、reset: 重置

【基础】不推荐使用--hard，因为会丢失修改的内容

`git reset --hard commitId` `commitId`表示需要重置到哪一次的提交

`HEAD is now at 6e90bcc` 此时第三次提交的内容会完全丢失

注意：提交是查看git log，不要提交带有merge的操作，如果有，使用reset进行回退。然后再进行add commit push。

```
D:\Code\low-code>git log --oneline
78eb4bf9 (HEAD -> dev-cxy-1, origin/dev-cxy-1) Bug-18960: oracle sql导入异常：String index out of range: -8; sql导入时无法获取主键错误
f5f3a1f5 Merge remote-tracking branch 'origin/dev-gaowenxiao-930' into dev-cxy-1
abc62918 (origin/dev-gaowenxiao-930, dev-gaowenxiao-930) Task-22808: change NativeFeignClient log level to debug & oracle parse error fixed & code review suggestion
0d925b3d Task-22808: change NativeFeignClient log level to debug & oracle parse error fixed
f0bf87ed Task-22808: support import entity deployment ddl
15807a58 Task-22806: support primary key property
2bae5d4a Task-22804: table to entity suport
1687c18d Task-22806: support primary key property
fbd6c353 Task-22804: table to entity suport
d6116cf7 Task-21816: support oracle database
fdcfd42b Task-21816: support oracle database
7be0a79b (dev-cxy) Merge branch 'dev-gwEnv' into 'dev'
e1acde6d (origin/dev-gwEnv) Task-22398: set env as Constant
ff08c787 Task-22398: add gateway interface call env header
```

```
D:\Code\low-code>git log
commit 78eb4bf972f189d2ccf32c5e420eb9152d56c0e1 (HEAD -> dev-cxy-1, origin/dev-cxy-1)
Author: chenxianyu <chenxianyu@corp.netease.com>
Date:   Sun Sep 26 09:54:55 2021 +0800

    Bug-18960: oracle sql导入异常：String index out of range: -8; sql导入时无法获取主键错误

commit f5f3a1f59aa297da9feee025461f1a2aab717d23
Merge: 0d925b3d abc62918
Author: chenxianyu <chenxianyu@corp.netease.com>
Date:   Fri Sep 24 11:38:32 2021 +0800

    Merge remote-tracking branch 'origin/dev-gaowenxiao-930' into dev-cxy-1

    # Conflicts:
    #       vusion/src/main/java/com/netease/cloud/lowcode/utils/OracleParser.java

commit abc62918abf41164af9004da92ba562c09115cda (origin/dev-gaowenxiao-930, dev-gaowenxiao-930)
Author: 高文晓 <gaowenxiao@corp.netease.com>
Date:   Wed Sep 22 11:03:18 2021 +0800

    Task-22808: change NativeFeignClient log level to debug & oracle parse error fixed & code review suggestion

D:\Code\low-code>git reset abc62918abf41164af9004da92ba562c09115cda
Unstaged changes after reset:
M       vusion/src/main/java/com/netease/cloud/lowcode/utils/OracleParser.java
M       vusion/src/test/java/com/netease/cloud/lowcode/facade/impl/SqlParserFacadeImplTest.java

D:\Code\low-code>
```

修改之后：

```
D:\Code\low-code>git log --oneline
2fb0d6da (HEAD -> dev-cxy-1, origin/dev-cxy-1) Bug-18960 oracle sql类型文件导入报错
abc62918 (origin/dev-gaowenxiao-930, dev-gaowenxiao-930) Task-22808: change NativeFeignClient log level to
f0bf87ed Task-22808: support import entity deployment ddl
15807a58 Task-22806: support primary key property
2bae5d4a Task-22804: table to entity suport
1687c18d Task-22806: support primary key property
fbd6c353 Task-22804: table to entity suport
d6116cf7 Task-21816: support oracle database
```

【注意】少用merge命令，使用rebase -i命令

git rebase -i bugfix-gaowenxiao-930 // 把930的bugfix的更新合并到自己的本地开发分支

git pull

git checkout -b xxx origin/xxx 切换本地分支

git restore low-code-app-archetype/make-archetype.sh // 撤销对某个文件的更改

5、Java对象序列化方式

网络传输的数据都必须是二进制数据，但是在Java中都是对象，是没有办法在网络中进行传输的，所以就需要对Java对象进行序列化，而且这个要求这个转换算法是可逆的。

5.1、Java原生序列化方式

实现方式：让类实现 Serializable 接口就可以了，具体的实现是由`ObjectOutputStream``和``ObjectInputStream`来实现的。

缺点：

- 1：序列化码流太大
- 2：序列化效率低
- 3：无法跨语言

优点：

- 1：简单

5.2、jackson、和fastjson（阿里）序列化

1、导入依赖

```
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-annotations -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.12.5</version>
</dependency>

<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.28</version>
</dependency>
```

2、jackson使用JsonProperty、fastjson使用JSONField 将属性名称序列化为另一个名称

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Student {

    // 把realName属性的名称序列化为另外一个名称real_name
    @JsonProperty(value = "real_name")
    private String realName;

    @JSONField(name = "AGE")
    private Integer age;
}
```

```java
// jackson序列化
Student student = new Student();
student.setRealName("zhangsan");
student.setAge(20);
System.out.println(new ObjectMapper().writeValueAsString(student));

// @JsonProperty不仅仅是在序列化的时候有用，
// 反序列化的时候也有用，比如有些接口返回的是json字符串，
// 命名又不是标准的驼峰形式，在映射成对象的时候，
// 将类的属性上加上@JsonProperty注解，里面写上返回的json串对应的名字
String jsonStr = "{\"real_name\":\"zhangsan\"}";
Student student1 = new ObjectMapper().readValue(jsonStr.getBytes(),
Student.class);
System.out.println(student1);


// @JsonProperty(value = "real_name")没有生效，因为fastjson不认识@JsonProperty注
解
System.out.println(JSON.toJSONString(student));



// result
{"age":20,"real_name":"zhangsan"}
Student(realName=zhangsan, age=null)
{"AGE":20,"realName":"zhangsan"}
```

6、合并Map

```java
map1.forEach((key, value) -> map2.merge(key, value, (v1, v2) -> (v2)));
```

7、docker

```
docker run -p 3306:3306 --name mysql8026 -e MYSQL_ROOT_PASSWORD=root -d mysql:8.0.26
docker exec -it mysql8026 bash

docker run -d -p 1521:1521 --name oracle oracleinanutshell/oracle-xe-11g
docker exec -it oracle bash

用户名/密码  xianyu/123456
```