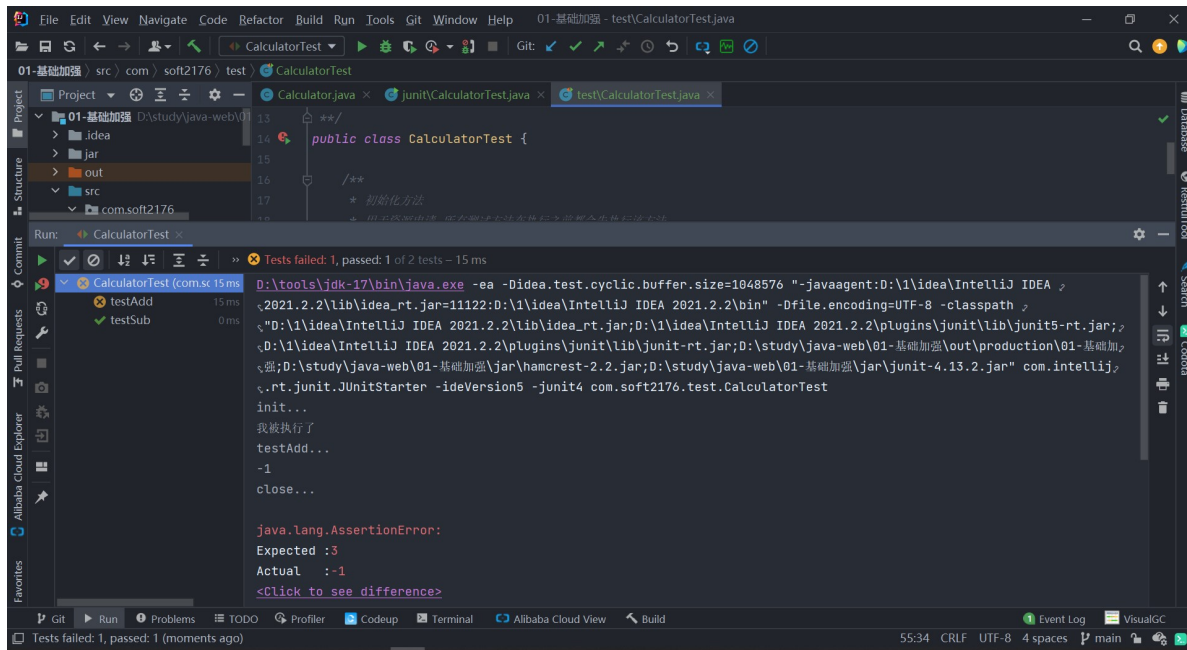


基础加强练习笔记

Junit单元测试

- @Before初始化方法 用于资源申请 所有测试方法在执行之前都会先执行该方法
- @After释放资源方法 在所有测试方法执行完毕后 都会自动执行该方法

使用单元测试对Calculator的方法进行测试 期望值不匹配 断言失败



反射

获取Class对象的三种方式

- `Class.forName("包名.类名")`: 将字节码文件加载进内存, 返回Class对象
- `类名.class`: 通过类名的class属性获取
- `对象.getClass()`: `getClass()`方法在Object类中定义

```
ReflectDemo1.java
@create: 2022-02-19 23:45
/**
 *
 */
public class ReflectDemo1 {
    public static void main(String[] args) throws ClassNotFoundException {
        //Class.forName("全类名")
        Class<?> cls1 = Class.forName("com.soft2176.domain.Person");
        System.out.println(cls1);
        //类名.class
        Class<?> cls2 = Person.class;
        System.out.println(cls2);
        //对象.getClass()
        Person p = new Person();
        Class<?> cls3 = p.getClass();
        System.out.println(cls3);
        //通过==比较三个对象 结果均为true
        System.out.println(cls1 == cls2);
        System.out.println(cls1 == cls3);
        //创建Student类的Class对象 然后比较 结果为false
        Class<?> c = Student.class;
        System.out.println(c == cls1);
    }
}
```

```
Run: ReflectDemo1
D:\tools\jdk-17\bin\java.exe
"-javaagent:D:\1\idea\IntelliJ IDEA
2021.2.2\lib\idea_rt
.jar=11421:D:\1\idea\IntelliJ IDEA
2021.2.2\bin" -Dfile.encoding=UTF-8
-classpath D:\study\java-web\01-基础
加强\out\production\01-基础加强
;D:\study\java-web\01-基础加强
\jar\hamcrest-2.2.jar;
D:\study\java-web\01-基础加强
\jar\junit-4.13.2.jar com.soft2176
.reflect.ReflectDemo1
class com.soft2176.domain.Person
class com.soft2176.domain.Person
class com.soft2176.domain.Person
true
true
false

Process finished with exit code 0
```

- Field[] getFields()可以获取所有public修饰的成员变量
- Field getField(String name)根据字段名获取
- Field[] getDeclaredFields()可以不考虑修饰符获取所有的成员变量
- Field getDeclaredFields(String name)根据字段名获取所有字段 需要设置setAccessible(true)将字段设置成可访问 否则会出现IllegalAccessException非法访问异常

```
public static void main(String[] args) throws Exception {
    //获取Person的Class对象
    Class<Person> personClass = Person.class;
    //Field[] getFields() 获取所有public修饰的成员变量
    Field[] fields = personClass.getFields();
    for(Field field : fields) {
        System.out.println(field);
    }
    System.out.println("=====");
    //Field getField(String name)
    Field a = personClass.getField("a");
    //获取成员变量a的值
    Person p = new Person();
    Object value = a.get(p);
    System.out.println(value);
    //设置a的p属性的值
    a.set(p, "张三");
    System.out.println(p);
    System.out.println("=====");
    //Field[] getDeclaredFields() 获取所有的成员变量 不考虑修饰符
    Field[] declaredFields = personClass.getDeclaredFields();
    for(Field declaredField : declaredFields) {
        System.out.println(declaredField);
    }
}
```

```
.ReflectDemo2
public java.lang.String com.soft2176
.domain.Person.a
=====
null
Person{name='null', age=0, a='张三',
b='null', c='null', d='null'}
=====
private java.lang.String com.soft2176
.domain.Person.name
private int com.soft2176.domain.Person
.age
public java.lang.String com.soft2176
.domain.Person.a
protected java.lang.String com.soft2176
.domain.Person.b
java.lang.String com.soft2176.domain
.Person.c
private java.lang.String com.soft2176
.domain.Person.d
null

Process finished with exit code 0
```

获取所有构造方法

- Constructor<?>[] getConstructors()
- Constructor getConstructor(类<?>... parameterTypes)
- Constructor getDeclaredConstructor(类<?>... parameterTypes)
- Constructor<?>[] getDeclaredConstructors()

```
ReflectDemo3.java x Calculator.java x
10 create: 2022-02-20 13:56
11
12 ▶ ic class ReflectDemo3 {
13 ▶ public static void main(String[] args) throws Exception {
14     // 获取Person的Class对象
15     Class<Person> personClass = Person.class;
16     // 获取带参数构造方法
17     Constructor<Person> constructor = personClass.getConstructor(S
18     System.out.println(constructor);
19     // 通过构造方法创建对象 (注意参数对应
20     Object person = constructor.newInstance("张三", 23);
21     System.out.println(person);
22     System.out.println("=====");
23     // 获取无参构造方法
24     Constructor<Person> constructor1 = personClass.getConstructor(
25     System.out.println(constructor1);
26     // 创建对象
27     Object person1 = constructor1.newInstance();
28     System.out.println(person1);
29     // 强制将构造方法设置为可以访问
30     constructor1.setAccessible(true);
31
32 }
33
```

```
Run: ReflectDemo3 x
D:\tools\jdk-17\bin\java.exe
"-javaagent:D:\1\idea\IntelliJ IDEA
2021.2.2\lib\idea_rt
.jar=3824:D:\1\idea\IntelliJ IDEA
2021.2.2\bin" -Dfile.encoding=UTF-8
-classpath D:\study\java-web\01-基础加强
\out\production\01-基础加强
\jar
\hamcrest-2.2.jar;
D:\study\java-web\01-基础加强\jar\junit
-4.13.2.jar com.soft2176.reflect
.ReflectDemo3
public com.soft2176.domain.Person(java
.lang.String,int)
Person{name='张三', age=23, a='null',
b='null', c='null', d='null'}
=====
public com.soft2176.domain.Person()
Person{name='null', age=0, a='null',
b='null', c='null', d='null'}

Process finished with exit code 0
```

获取所有成员方法

- Method[] getMethods()
- Method getMethod(String name, 类<?>... parameterTypes) 获取参数类型为parameterTypes 方法名为name的方法 可以使用.invoke()执行方法
- Method[] getDeclaredMethods()
- Method getDeclaredMethod(String name, 类<?>... parameterTypes)

```
10 * @create: 2022-02-20 14:13
11 **/
12 ▶ public class ReflectDemo4 {
13 ▶ public static void main(String[] args) throws Exception{
14     // 获取Person的Class对象
15     Class<Person> personClass = Person.class;
16     // 获取指定名称的方法eat(无参的)
17     Method eatMethod = personClass.getMethod( name: "eat");
18     Person p = new Person();
19     // 通过invoke执行目标方法
20     eatMethod.invoke(p);
21     // 获取带string类型参数的eat方法
22     Method eatMethod2 = personClass.getMethod( name: "eat",String.class);
23     // 执行带参的eat方法
24     eatMethod2.invoke(p, "饭");
25     System.out.println("=====");
26     // 获取所有Public修饰的方法
27     Method[] methods = personClass.getMethods();
28     for(Method method : methods) {
29         System.out.println(method);
30         String name = method.getName();
31         System.out.println(name);
32         method.setAccessible(true);
33     }
34 }
```

```
eat...
eat...饭
=====
public java.lang.String com.soft2176.domain
.Person.getName()
getName
public java.lang.String com.soft2176.domain
.Person.toString()
toString
public com.soft2176.domain.Person com.soft2176
.domain.Person.setName(java.lang.String)
setName
public com.soft2176.domain.Person com.soft2176
.domain.Person.setA(java.lang.String)
setA
public com.soft2176.domain.Person com.soft2176
.domain.Person.setB(java.lang.String)
setB
public java.lang.String com.soft2176.domain
.Person.getC()
getC
public com.soft2176.domain.Person com.soft2176
.domain.Person.setC(java.lang.String)
setC
public java.lang.String com.soft2176.domain
.Person.getD()
getD
```

通过读取配置文件获取对象

```
11  /**
12  ▶ public class ReflectTest {
13  ▶     public static void main(String[] args) throws Exception{
14      // 创建Properties对象
15      Properties pro = new Properties();
16      // 获取class目录下的配置文件
17      ClassLoader classLoader = ReflectTest.class.getClassLoader();
18      InputStream is = classLoader.getResourceAsStream( name: "pro.properties");
19      // 加载配置文件
20      pro.load(is);
21      // 获取文本文件中定义的数据
22      String className = pro.getProperty("className");
23      String methodName = pro.getProperty("methodName");
24      // 加载类进内存
25      Class<?> cls = Class.forName(className);
26      // 创建对象
27      Object obj = cls.getConstructor().newInstance();
28      // 获取方法对象
29      Method method = cls.getMethod(methodName);
30      // 执行目标方法
31      method.invoke(obj);
32  }
33  }
34  }
```

```
D:\tools\jdk-17\bin\java.exe
"-javaagent:D:\1\idea
\IntelliJ IDEA
2021.2.2\lib\idea_rt
.jar=12208:D:\1\idea
\IntelliJ IDEA
2021.2.2\bin"
-Dfile.encoding=UTF-8
-classpath
D:\study\java-web\01-
基础加强\out\production
\01-基础加强;
D:\study\java-web\01-
基础加强\jar\hamcrest-2
.2.jar;D:\study\java
-web\01-基础加强
\jar\junit-4.13.2.jar
com.soft2176.reflect
.ReflectTest
sleep...

Process finished with
exit code 0
```

注解

通过自定义check注解检查程序异常

在catch中通过getCause()得到异常信息并记录到文件中

```
CheckJava x Calculator.java x TestCheck.java x bugtest x
Plugins supporting *.test files found. Install plugins Ignore extension

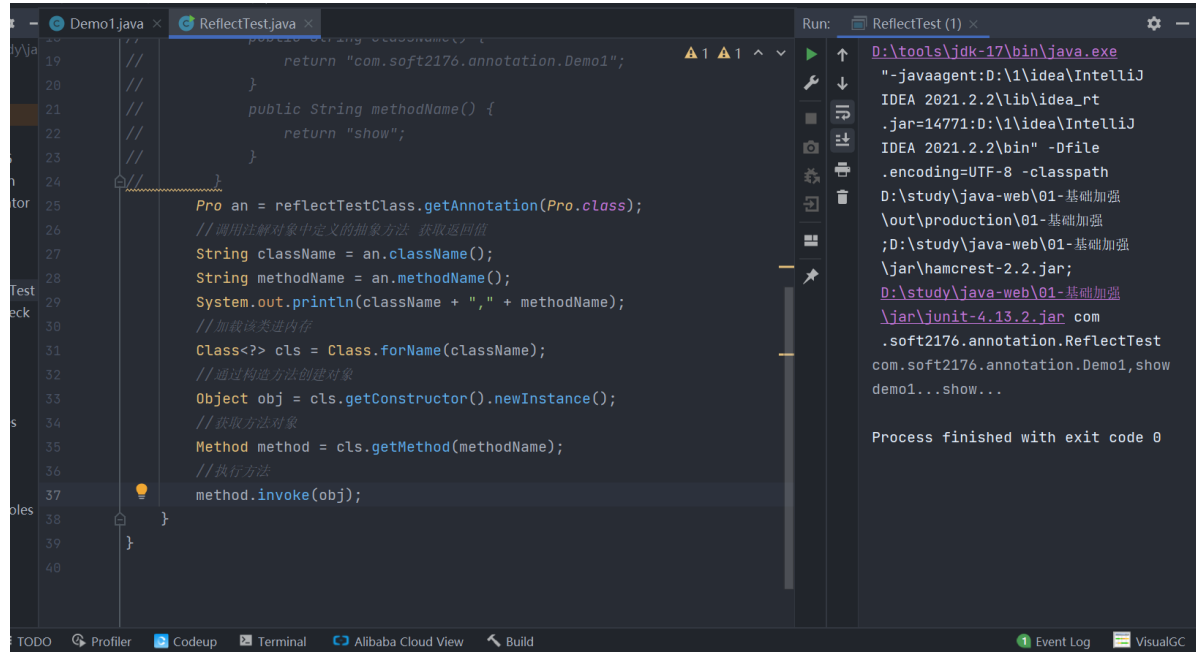
1  add方法出异常了
2  异常的名称:NullPointerException
3  异常的原因:Cannot invoke "String.length()" because "str" is null
4  -----
5  div方法出异常了
6  异常的名称:ArithmeticException
7  异常的原因:/ by zero
8  -----
9  本次测试一共出现2 次异常

Run: TestCheck x
D:\tools\jdk-17\bin\java.exe
"-javaagent:D:\1\idea\IntelliJ
IDEA 2021.2.2\lib\idea_rt
.jar=3147:D:\1\idea\IntelliJ I
2021.2.2\bin" -Dfile
.encoding=UTF-8 -classpath
D:\study\java-web\01-基础加强
\out\production\01-基础加强
;D:\study\java-web\01-基础加强
\jar\hamcrest-2.2.jar;
D:\study\java-web\01-基础加强
\jar\junit-4.13.2.jar com
.soft2176.annotation.TestCheck
1 - 0 =1
1 * 0 =0

Process finished with exit code
```

使用自定义注解描述需要执行的类名和方法名

通过类的getAnnotation()方法获取类的注释对象



The screenshot displays an IDE with two main panels. The left panel shows the source code of `ReflectTest.java`, and the right panel shows the output of the `Run` operation.

Source Code (ReflectTest.java):

```
19 //         return "com.soft2176.annotation.Demo1";
20 //     }
21 //     public String methodName() {
22 //         return "show";
23 //     }
24 // }
25 Pro an = reflectTestClass.getAnnotation(Pro.class);
26 // 调用注解对象中定义的抽象方法 获取返回值
27 String className = an.className();
28 String methodName = an.methodName();
29 System.out.println(className + "," + methodName);
30 // 加载该类进内存
31 Class<?> cls = Class.forName(className);
32 // 通过构造方法创建对象
33 Object obj = cls.getConstructor().newInstance();
34 // 获取方法对象
35 Method method = cls.getMethod(methodName);
36 // 执行方法
37 method.invoke(obj);
38 }
39 }
40 }
```

Run Output (ReflectTest (1)):

```
D:\tools\jdk-17\bin\java.exe
"-javaagent:D:\1\idea\IntelliJ
IDEA 2021.2.2\lib\idea_rt
.jar=14771:D:\1\idea\IntelliJ
IDEA 2021.2.2\bin" -Dfile
.encoding=UTF-8 -classpath
D:\study\java-web\01-基础加强
\out\production\01-基础加强
;D:\study\java-web\01-基础加强
\jar\hamcrest-2.2.jar;
D:\study\java-web\01-基础加强
\jar\junit-4.13.2.jar com
.soft2176.annotation.ReflectTest
com.soft2176.annotation.Demo1,show
demo1...show...

Process finished with exit code 0
```