1. 案例01：Junit单元测试

1.1 需求说明

1. 设置一个类Calculator，包含4个方法：加、减、乘、除，使用JUnit对4个方法进行单元测试。
2. 在每个方法运行之前创建Calculator对象，在测试方法运行完毕之后将对象设置为null 。

2. 案例02：反射案例1

2.1 需求说明

1. 现有集合：ArrayList list = new ArrayList();
2. 利用反射机制在这个泛型为Integer的ArrayList中存放一个String类型的对象。

3. 案例03：反射案例2

3.1 需求说明

- 定义一个Student类，用反射去创建一个Student对象，使用两种方式
    1. 通过Class对象的方法创建。
    2. 通过Constructor对象的方法创建。

4. 案例04：反射案例3

4.1 需求说明

1. 定义一个类，在类中定义一个成员方法`show`，方法功能是：打印一个字符串。
2. 使用反射机制创建该类的对象，并调用该对象的`show`方法。

5. 案例05：反射案例4

5.1 需求说明

1. 编写一个类A，定义一个实例方法`showString`，用于打印一个字符串。
2. 在编写一个类TestA，用键盘输入一个字符串，该字符串就是类A的全名，使用反射机制创建该类的对象，并调用该对象中的方法showString

6. 案例06：反射案例5

6.1 需求说明

按要求完成下面两个方法的方法体
写一个方法，此方法可将obj对象中名为propertyName的属性的值设置为value.

```
public void setProperty(Object obj, String propertyName, Object value){

}
```

写一个方法，此方法可以获取obj对象中名为propertyName的属性的值

```
public Object getProperty(Object obj, String propertyName){

}
```

7. 案例07：反射案例6

7.1 需求说明

1. 定义一个Person类，包含属性name、age。
2. 使用反射的方式创建一个实例、调用构造函数初始化name、age。使用反射方式调用setName方法对姓名进行设置，不使用setAge方法直接使用反射方式对age赋值。

8. 案例08：反射案例7

8.1 需求说明

已知一个类，定义如下

```
package com.itheima;
public class DemoClass {
	public void run() {
		System.out.println("welcome to heima!");
	}
}
```

(1)写一个Properties格式的配置文件，配置类的完整名称。
(2)写一个程序，读取这个Properties配置文件，获得类的完整名称并加载这个类，
(3)用反射的方式运行run方法。

9. 案例9：反射案例8

9.1 需求说明

有一个用于记录程序运行次数的属性文件，运行次数保存在一个count属性中，当到达指定次数3次时，则提示:"程序使用次数已满，请续费"
1. 开发思路:
1). 判断属性文件是否存在，如果不存在则创建一个。
2). 使用load()方法加载文件中所有的属性到Properties集合中。
3). 取得count属性，如果count属性为null，则设置count属性为0。
4). 将取得的字符串转成整型，并判断是否大于等于3次，大于3次则到期，退出。
5). 小于3则输出运行次数，并加1。
6). 将整数转成字符串后存到Properties集合中。
7). 创建输出流，并用store方法保存到文件中。

10. 案例10：注解01

10.1 需求说明

一、 创建新项目，按以下要求定义，并使用注解：

1. 请定义一个最简单的注解@MyAnno1
   1) 不需要任何属性。
   2) 此注解只能修饰“类”和接口
   3) 此注解要出现在源码和字节码中
   4) 定义测试类：Test1，并使用此注解修饰
2. 请定义注解@MyAnno2：
   1) 包含一个String类型的属性“type”，并且定义默认值“java”。
   2) 此注解只能修饰“字段”。
   3) 此注解只需要能够在源码中使用。
   4) 定义测试类：Test2，随意定义一个成员属性，并使用此注解；
3. 请定义注解@MyAnno3：
   1) 包含一个String类型的属性“type”，不定义默认值。
   2) 包含一个int[]数组类型的属性“intArr”，不定义默认值。
   3) 此注解只能修饰“方法”。
   4) 此注解要出现在源码和字节码中。
   5) 定义测试类：Test3，随意定义一个成员方法，并使用此注解；

11. 案例11：注解02

11.1 需求说明

1) 模拟JUnit测试的注释@Test，首先需要编写自定义注解@MyTest，并添加元注解，保证自定义注解只能修饰方法，且在运行时可以获得。
2) 其次编写目标类（测试类），然后给目标方法（测试方法）使用@MyTest注解，编写三个方法，其中两个加上@MyTest注解。
3) 最后编写调用类，使用main方法调用目标类，模拟JUnit的运行，只要有@MyTest注释的方法都会运行。

12. 案例12：注解解析

12.1 需求说明

定义一个注解：Book

```
        * 包含属性：String value()   书名
        * 包含属性：double price()  价格，默认值为 100
        * 包含属性：String[] authors() 多位作者
```

1. 定义类在成员方法上使用Book注解
2. 解析获得该成员方法上使用注解的属性值。