反射

回顾：

```

```

今天任务

```
1.可变参数
2.反射
```

教学目标

```
1.掌握可变参数
2.掌握反射
```

第一节：可变参数

从JDK 5开始，Java 允许为方法定义长度可变的参数。

语法：

```
public void foo(int … args){ //int...表示数组

}
```

注意事项：

- 调用可变参数的方法时, 编译器将自动创建一个数组保存传递给方法的可变参数，因此，程序员可以在方法体中以数组的形式访问可变参数
- 可变参数只能处于参数列表的最后, 所以一个方法最多只能有一个长度可变的参数

案例一：

```
public static void main(String[] args) {
        int[] nums={10,20,30};
        showNumber(nums);
        //----可变参数调用 好处：调用是可以传递数组元素
        //注意：1 可以必须放在方法参数列表的最后
         //    2 一个只能有一个可变参数
        showNumber(10,20);
        //showNames("zhangsan","lisi","wangwu");
    }
    public static void showNumber(int... nums){ // ...  []
        System.out.println(nums.length);
        for (int i : nums) {
            System.out.println(i);
        }
    }
    public static void showNames(String...names){
        for (String n : names) {
            System.out.println(n);
        }
    }
```

第二节：反射

2.1 为什么使用反射

需求：
　　我公司定义了一组接口，然后第三方公司按照我公司的接口实现了一套功能，然后交给我们，但是我们公司的项目已经结束，如何实现动态加载第三方公司提供的功能。

2.2 什么是反射

反射就是把Java类中的各种成分映射成一个个的java对象。例如，一个类有：成员变量，方法，构造方法，包等等信息，利用反射技术可以对一个类进行解剖，把各个组成部分映射成一个个对象。

2.3 反射常用类

- Class类—可获取类和类的成员信息
- Field类—可访问类的属性
- Method类—可调用类的方法
- Constructor类—可调用类的构造方法

2.4 使用反射的基本步骤

1.导入java.lang.reflect.*

2.获得需要操作的类的Java.lang.Class对象

3.调用Class的方法获取Field、Method等对象

4.使用反射API进行操作(设置属性、调用方法）

第三节：Class类

3.1 Class类是反射机制的起源和入口

- 每个类都有自己的Class对象
- 用于获取与类相关的各种信息
- 提供了获取类信息的相关方法
- Class类继承自Object类

3.2 Class类存放类的结构信息

- 类名
- 父类、接口
- 方法、构造方法、属性
- 注释

3.3 获取 Class对象的方式

第一种方式

```
//方法1：对象.getClass()
Student stu=new Student();
Class clazz=stu.getClass();
```

第二种方式

```
//方法2：类.class
clazz= Student.class;
clazz=String.class;
```

第三种方式

```
//方法3：Class.forName()
clazz=Class.forName("java.lang.String");
clazz=Class.forName("java.util.Date");
```

3.4 获取类的其他结构信息

```
Class clazz = Class.forName("java.lang.Object");
Field fields[ ] = clazz.getDeclaredFields();//获取Field 对象
Method methods[] = clazz.getDeclaredMethods();//获取Method 对象
Constructor constructors[ ] = clazz.getDeclaredConstructors();//获取Constructor对象
```

3.5 动态创建对象

方法一：使用Class的newInstance()方法，仅适用于无参构造方法

```
Class clazz=Class.forName("com.qf.reflection.Student");
Object obj=clazz.newInstance();
```

方法二：调用Constructor的newInstance()方法，适用所有构造方法

```
Constructor cons = clazz.getConstructor(new Class[]{ String.class,  int.class, float.class });
Object obj = cons.newInstance(new Object[ ] {"lkl", 32, 56.5f });
```

练习一：

1 定义Student类，包含：姓名和年龄等属性，有参和无参构造方法，输出所有信息的方法

2 使用多种方法生成一个Student类的Class对象

3 使用Class类获取Student类的属性并输出

4 通过有参(无参)构造方法动态创建Student类的对象

3.6 动态执行方法

调用方法基本步骤：

1.通过Class对象获取Method 对象

2.调用Method对象的invoke()方法

例如：

```
Object invoke(Object obj,Object [] args);
//object 返回值
//obj 当前方法所属对象
//args 当前方法的参数列表
```

3.7 反射动态操作属性值

操作属性的基本步骤

1.通过Class对象获取Field 对象

2.调用Field 对象的方法进行取值或赋值操作

方法	说　　明
Xxx getXxx(Object obj)	获取基本类型的属性值
Object get(Object obj))	得到引用类型属性值
void setXxx(Object obj,Xxx val)	将obj对象的该属性设置成val值
void set(Object obj,object val)	将obj对象的该属性设置成val值
void setAccessible（bool flag）	对获取到的属性设置访问权限

第四节：反射技术的优点和缺点

优点：

1.提高了Java程序的灵活性和扩展性，降低了耦合性，提高自适应能力

2.允许程序创建和控制任何类的对象，无需提前硬编码目标类

缺点：

1.性能问题

2.代码维护问题

第五节：综合案例

```java
public class Demo2 {
    public static void main(String[] args) throws Exception{
//        getClassObj();
        //getConstructor();
        getMethod();
    }
    /**
     * 获取类对象
     */
    public static void getClassObj() throws Exception{
        //第一种方法:使用类名.class 获取类对象
        Class<?> class1= Person.class;
        System.out.println(class1.hashCode());
        //第二种方法:使用对象来获取类对象
        Person zhangsan=new Person();
        Class<?> class2=zhangsan.getClass();
        System.out.println(class2.hashCode());
        //第三种方式:使用Class.forname()获取类对象
        Class<?> class3=Class.forName("com.qf.day25_2.Person");
        System.out.println(class3.hashCode());
    }
    /**
     * 获取构造方法
     */
    public static void getConstructor() throws Exception{
        //1获取类对象
        Class<?> class1=Class.forName("com.qf.day25_2.Person");
        //2获取构造方法
//        Constructor<?>[] constructor=class1.getConstructors();
//        //3遍历
//        for (Constructor<?> c : constructor) {
//            System.out.println(c);
//        }
        //4获取一个无参构造方法
        Constructor<?> constructor=class1.getConstructor();
        System.out.println(constructor);
        //5获取带参构造方法
        Constructor<?> constructor2=class1.getConstructor(String.class,int.class,String.class);
        System.out.println(constructor2);

        //6使用构造方法构造对象
        Object zhangsan=constructor.newInstance();
        Object lisi=constructor2.newInstance("李四",20,"女");
        Object wangwu=class1.newInstance();//调用无参构造
        //Person wangwu=new Person("王五",20, "男");
        System.out.println(zhangsan.toString());
        System.out.println(lisi.toString());
        System.out.println(wangwu);

        //System.out.println(wangwu);
```

```
    }
    /**
     * 获取方法
     */
    public static void getMethod() throws Exception{
        //1获取类对象
        Class<?> class1=Class.forName("com.qf.day25_2.Person");
        //2获取方法 (获取所有的公开的方法，包括从父类继承过来的)
        //Method[] methods=class1.getMethods();
        //获取类中所有的方法，包括私有的，不包括继承的方法
//      Method[] methods=class1.getDeclaredMethods();
//      for (Method method : methods) {
//          System.out.println(method);
//      }

        Constructor<?> constructor=class1.getConstructor(String.class,int.class,String.class);
        Object zhangsan=constructor.newInstance("zhangsan",20,"男");

        //3获取一个无参方法
        Method method=class1.getDeclaredMethod("show");
        //4调用方法
        method.invoke(zhangsan);
        //5获取一个带参方法
        Method method2=class1.getDeclaredMethod("show", String.class);
        method2.invoke(zhangsan, "郑州");
        //6获取带返回值的方法
        Method method3=class1.getDeclaredMethod("getInfo");
        String s=(String) method3.invoke(zhangsan);
        System.out.println(s);
        //7获取静态方法
        Method method4=class1.getDeclaredMethod("showCountry");
        method4.invoke(null);
        //8获取私有方法
        Method method5=class1.getDeclaredMethod("privateMethod");
        //取消访问性检查
        method5.setAccessible(true);
        method5.invoke(zhangsan);
        //9获取属性
        Field field=class1.getDeclaredField("name");
        field.setAccessible(true);
        Object lisi=class1.newInstance();
        field.set(lisi, "李四");
        String name= (String) field.get(lisi);
        System.out.println(name);
        }
}
```

第七节：默写：

1获取类对象的三种方式

第八节：作业：

1 设计一个Student类，属性有：id，name,age,borndate,使用反射创建对象。

2 使用反射调用方法show显示学生信息。

3 使用反射调用name属性。

第九节：面试题：

1 简述Java中的反射使用

2 JAVA常用反射API