# 核心部分

## 【单选题】

1. 反射中Class获得方式不包含哪一个( )

|  |
| --- |
| 1. 类名.class； 2. 对象名.getClass(）； 3. Class.forName("全类名”）； 4. 类名.getClass()； |

1. 有如下类定义

|  |
| --- |
| package cn.itcast.demo07;  public class Student {  public Student() {  System.out.println("a");  }  public void show(){  System.out.println("b");  }    }  2 测试方法:  package cn.itcast.demo07;  import java.lang.reflect.Method;  public class Demo {  public static void main(String[] args) throws Exception{  Class c = Class.forName("cn.itcast.demo07.Student");  Object obj =c.getConstructor().newInstance();  Method m =c.getMethod("show");  m.invoke(obj);  }  }  请选择一个正确答案( ) |

|  |
| --- |
| 1. 打印：a 2. 打印：b 3. 打印：ab 4. 编译错误 |

1. 有如下类定义

|  |
| --- |
| 1 有如下类定义:  package cn.itcast.demo01;  public class Student {  public Student() {  System.out.println("a");  }  public void show(){  System.out.println("b");  }    }  2 测试方法:  package cn.itcast.demo01;  import java.lang.reflect.Method;  public class Demo {  public static void main(String[] args) throws Exception{  Class c = Class.forName("cn.itcast.demo01.Student");  Object obj =c.getConstructor();  Method m =c.getMethod("show");  m.invoke(obj);  }  }  请选择一个正确答案( ) |

|  |
| --- |
| 1. 编译错误 2. 运行时异常 3. 打印: a 4. 打印: ab |

1. 关于反射，下面说法错误的是( )

|  |
| --- |
| 1. 反射调用方法时，如果有形参，全部是Object类型； 2. 通过反射，可以直接访问类对象中的私有成员 3. 反射调用方法时，如果有形参，调用时必须传递实参； 4. 反射的过程是先获取某个类的Class对象，然后创建此类对象，并访问成员； |
|  |

1. 有如下类定义

|  |
| --- |
| 1 有如下类定义:  package cn.itcast.demo01;  public class Phone {  public Phone() {  System.out.println("a");  }  public void call(String s){  System.out.println("b");  }  }  2 测试方法:  public class Demo {  public static void main(String[] args) throws Exception{  Class c =Phone.class;  Object obj =c.getConstructor().newInstance();  Method m =c.getMethod("call");  m.invoke(obj,"a");  }  }  请选择一个正确答案( ) |

|  |
| --- |
| * 1. 编译错误   2. 打印: a和运行时异常   3. 打印: a   4. 打印: ab |

## 【多选题】

1. 有如下类定义：

|  |
| --- |
| ① package cn.itcast.demo02；  public class Student{  ② package cn.itcast.demo02;  public class Demo{  public static void main(String[] args){  Classc=Class.forName("cn.itcast.demo02.Student");  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  }  }  请问，在横线处填写什么代码，可以构造一个Student对象( ) |

|  |
| --- |
| 1. Object o=c.createlnstance(); 2. Object o=c.newlnstance()； 3. Object o=c.getConstructor().newlnstance(); 4. Object o=c.getConstructor().createlnstance()； |

1. 有如下代码

|  |
| --- |
| ①package cn.itcast.demo03；  public class Student{  private String name；  private Student(String name){  this.name=name；  }  public void show(){  System.out.printin("show");  }  }  请问，以下说法正确的是：( ) |

|  |
| --- |
| 1. 获取Student的Class对象可以使用以下方式：   Classc=Student.class;   1. 获取Student的Class对象可以使用以下方式：   Classc=Class.forName("Student")；   1. 获取Student的Class对象可以使用以下方式：   Classc=Student.getClass()；   1. 反射构造Student对象可以使用以下方式：   Constructor cot=c.getDeclaredConstructor(String.class); |

1. 有如下类定义

|  |
| --- |
| ①public class Student{  private String name;  public void setName(String name){  this.name=name;  }  public String getName(){  return this.name;  }  }  ② public class Demo{  public static void main(Stringl[] args)throws Exception{  Classc=Student.class;  Object obj=c.newlnstance();  fset(obj,"b");  System.out.printin(f.get(obj);  }  }  请按顺序选择答案，补全代码，使程序最终打印：b( ) |

|  |
| --- |
| 1. Field f=c.getField("name"); 2. f.setAccessible(true); 3. Filed f=c.getDeclaredField("name"); 4. Filed.Accessible(true); |

1. 有如下类定义

|  |
| --- |
| public class Student{  public void show(){  System.out.printin("a");  }  }  请问，以下哪种方式可以正确调用show）方法（假设以下代码在同包下的Demo类的main方法中）：( ) |

|  |
| --- |
| 1. Classc=Student.class;   Object o=c.newlnstance();  o.show();  B. Classc=Student.class;  Student stu=(Student)c.newlnstance();  stu.show();  C. Classc=Student.class；  ((Student)c.newintance()).show);  D. Classc=Student.class;  Object o=c.getConstructor().newlntance();  o.show(); |

1. 有以下类定义

|  |
| --- |
| ③ package cn.itcast.demo01;  public class Student{  private String name;  }  ④ package cn.itcast.demo02;  public class Demo{  public static void main(String[] args){  Classc=Class.forName("cn.itcast.demo01.Student");  Object o=c.newlnstance();  f.setAccessible(true);  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  }  }  请问，在横线处填写什么代码，可以为name属性赋值为ack”:( ) 请按顺序选择 |

|  |
| --- |
| 1. c.getField("name"); 2. c.getDeclaredField("name"); 3. f.set(c”Jack"); 4. f.set(o,”Jack"); |

# 简答题

##### 基础加强【反射、BeanUtils】

1. **Java反射机制的作用？**

|  |
| --- |
| 1）在运行时判断任意一个对象所属的类。  2）在运行时判断任意一个类所具有的成员变量和方法。  3）在运行时任意调用一个对象的方法  4）在运行时构造任意一个类的对象 |

1. **反射机制的优缺点？**

|  |
| --- |
| 静态编译：在编译时确定类型，绑定对象，即通过  动态编译：运行时确定类型，绑定对象。动态编译最大限度的发挥了java的灵活性，体现了多态的应用，有利于降低类之间的耦合性。 |

1. **什么是反射机制？**

|  |
| --- |
| 简单说，反射机制值得是程序在运行时能够获取自身的信息。在java中，只要给定类的名字，那么就可以通过反射机制来获得类的所有信息。 |

1. **反射机制提供了什么功能？**

|  |
| --- |
| 在运行时能够判断任意一个对象所属的类  在运行时构造任意一个类的对象  在运行时判断任意一个类所具有的成员变量和方法  在运行时调用任一对象的方法  在运行时创建新类对象 |

1. **哪里用到反射机制？**

|  |
| --- |
| jdbc中有一行代码：Class.forName('com.mysql.jdbc.Driver.class').newInstance();那个时候只知道生成驱动对象实例，后来才知道，这就是反射，现在  很多框架都用到反射机制，hibernate，struts都是用反射机制实现的。 |

1. **获得一个类的类对象有哪些方式？**

|  |
| --- |
| 答： -1：类型.class，例如：String.class  -2：对象.getClass()，例如："hello".getClass()  -3：Class.forName()，例如：Class.forName("java.lang.String") |

1. **如何通过反射创建对象？**

|  |
| --- |
| 答： 1：通过类对象调用 newInstance()方法，例如：String.class.newInstance()  2：通过类对象的 getConstructor()或 getDeclaredConstructor()方法获得构造器（Constructor）对象并 调用其 newInstance()方法创建对象，例如：String.class.getConstructor(String.class).newInstance("Hello"); |

1. **如何通过反射调用对象的方法？**

|  |
| --- |
| 答：请看下面的代码： import java.lang.reflect.Method;  class MethodInvokeTest {  public static void main(String[] args) throws Exception { String str = "hello"; Method m = str.getClass().getMethod("toUpperCase"); System.out.println(m.invoke(str)); // HELLO } } |

1. **如何通过反射获取和设置对象私有字段的值？**

|  |
| --- |
| 答：可以通过类对象的 getDeclaredField()方法字段（Field）对象，然后再通过字段对象的 setAccessible(true) 将其设置为可以访问，接下来就可以通过 get/set 方法来获取/设置字段的值了。下面的代码实现了一个反 射的工具类，其中的两个静态方法分别用于获取和设置私有字段的值，字段可以是基本类型也可以是对象 类型且支持多级对象操作，例如 ReflectionUtil.get(dog, "owner.car.engine.id");可以获得 dog 对象的主人的汽  51  车的引擎的 ID 号。 import java.lang.reflect.Constructor; import java.lang.reflect.Field; import java.lang.reflect.Modifier; import java.util.ArrayList; import java.util.List;  /\*\* \* 反射工具类 \* @author 骆昊 \* \*/ public class ReflectionUtil {  private ReflectionUtil() { throw new AssertionError(); }  /\*\* \* 通过反射取对象指定字段(属性)的值 \* @param target 目标对象 \* @param fieldName 字段的名字 \* @throws 如果取不到对象指定字段的值则抛出异常 \* @return 字段的值 \*/ public static Object getValue(Object target, String fieldName) { Class<?> clazz = target.getClass(); String[] fs = fieldName.split("\\.");    try { for(int i = 0; i < fs.length - 1; i++) { Field f = clazz.getDeclaredField(fs[i]); f.setAccessible(true); target = f.get(target); clazz = target.getClass(); }    Field f = clazz.getDeclaredField(fs[fs.length - 1]); f.setAccessible(true); return f.get(target); } catch (Exception e) { throw new RuntimeException(e); } } /\*\* \* 通过反射给对象的指定字段赋值 \* @param target 目标对象 \* @param fieldName 字段的名称 \* @param value 值 \*/ public static void setValue(Object target, String fieldName, Object value) { Class<?> clazz = target.getClass(); String[] fs = fieldName.split("\\."); try { for(int i = 0; i < fs.length - 1; i++) { Field f = clazz.getDeclaredField(fs[i]); f.setAccessible(true); Object val = f.get(target); if(val == null) { Constructor<?> c = f.getType().getDeclaredConstructor(); c.setAccessible(true); val = c.newInstance(); f.set(target, val); } target = val; clazz = target.getClass(); } Field f = clazz.getDeclaredField(fs[fs.length - 1]); f.setAccessible(true); f.set(target, value); } catch (Exception e) { throw new RuntimeException(e);  52    } } } |

1. **反射怎么理解，说一下反射经典的应用**

|  |
| --- |
| 答案：  反射是什么呢？当我们的程序在运行时，需要动态的加载一些类这些类可能之前用不到所以不用加载到jvm，而是在运行时根据需要才加载，这样的好处对于服务器来说不言而喻，举个例子我们的项目底层有时是用mysql，有时用oracle，需要动态地根据实际情况加载驱动类，这个时候反射就有用了，假设 com.java.dbtest.myqlConnection，com.java.dbtest.oracleConnection这两个类我们要用，这时候我们的程序就写得比较动态化，通过Class tc = Class.forName("com.java.dbtest.TestConnection");通过类的全类名让jvm在服务器中找到并加载这个类，而如果是oracle则传入的参数就变成另一个了。这时候就可以看到反射的好处了，这个动态性就体现出java的特性了！举个例子，使用spring中会发现当你配置各种各样的bean时，是以配置文件的形式配置的，你需要用到哪些bean就配哪些，spring容器就会根据你的需求去动态加载，你的程序就能健壮地运行。 |

1. **讲讲反射机制**

|  |
| --- |
| 答案  JAVA有着一个非常突出的动态相关机制：Reflection，用在Java身上指的是我们可以于运行时加载、探知、使用编译期间完全未知的classes。换句话说，Java程序可以加载一个运行时才得知名称的class，获悉其完整构造（但不包括methods定义），并生成其对象实体、或对其fields设值、或唤起其methods。 |