**中软国际考题-笔试**

1. 单选题

1、关于重载和重写，以下说法错误的是（D）

A. 重载方法的返回值、访问修饰符以及抛出的异常都不是重载方法判断的决定因素

B一个静态方法既可以被重载为一个静态方法，也可以被重载为一个非静态方法

C.方法的返回值类型不能作为重载的依据

D.重写的方法返回值必须和源方法返回值类型保持完全一致

2、如果A extends Object，B extends A，现有基类中的一个方法：A method(){},请问以下哪些是子类中重写该方法的正确形式？（D）

A、void method(){} B、int method(){} C、A method(B){} D、B method(){}

3、下列描述错误的是（B）

A.返回值在方法名前修饰设定

B.返回值必须为八大基本数据类型

C.返回值在方法中可以设定无返回值

D.方法参数与返回值基本无概念练习

4、下列描述正确的是（A）

1. 返回值在方法调用时其数据类型已被设定
2. 返回值必须由参数计算而成
3. 参数与返回值不允许数据类型一致。
4. 返回值一定表示方法调用完成的结果

5、java中返回值描述错误的是（D）

A.返回值没有在方法内部直接的变量名申明

B.返回值与各自的方法一一对应

C.返回值可以在方法调用时获取

D.递归调用的返回值不属于八大基本数据类型

6、下列语法，哪些不属于JDK5.0开始支持的语法（B）

A.可变个数参数 B.重写 C.协变返回类型 D.泛型

7、下列父类中的变量不能够被子类所访问的是（A）

class Parent {

private int i = 20;

protected int j = 30;

public int k = 40;

int h = 50;

}

class Child extends Parent {

void f() {

}

}

---------------------

A.i B.j C.k D.h

8、Java可变返回类型（协变）是从java哪个版本开始有的（C）

A.jdk 1.1 B.jdk1.2 C.jdk1.5 D.java8新特性

9.以下说法正确的是（B）

A. 数组是不可协变的 B. 泛型没有内建的协变类型

C. 泛型是具备内建的协变类型 D. 以上都不对

10．下列语句正确的是(A)

A.形式参数可被视为local variable B.形式参数可被所有的字段修饰符修饰

C.形式参数为方法被调用时，是真正被传递的参数 D. 形式参数不可以是对象

11. 下面函数将返回?

public static int func(){

try{

return 1;

}catch{

return 2;

}finally{

return 3;

}

A.1 B.2 C.3 D.0

12、创建一个只能存放String的泛型ArrayList的语句是哪项（B）.

 A. ArrayList<int> al=new ArrayList<int>()；

    B.  ArrayList<String> al=new ArrayList<String>()

    C.  ArrayList al=new ArrayList<String>()；

    D.  ArrayList<String> al =new List<String>();

13、 以下对于方法覆盖的说法错误的有（A）

A 方法覆盖发生在同一类中 B 方法的覆盖发生在子类中

C 方法名一定要一样 D 返回类型可以不一样

14、关于return语句，正确的说法是？(C)

A.只能让方法返回数值B.方法都必须含有

C.方法中可以有多句return 　　 D.不能用来返回对象

15、在一个函数中，允许几个return语句？（D）

A.1 B.2 C.3 D不限制数量

16、Java 5.0添加了对协变返回类型的支持，即子类覆盖（即重写）基类方法时，返回的类型可以不必严格等同于父类中被重写的成员函数的返回值类型，但是要求返回类型必须（B）

A. 是父类返回类型的继承时才被容许。 B.是Java中的基本类型才可以。

C. 是Java中的引用类型才可以。 D.不是父类返回类型的继承时才可以。

1. 多选题
2. 下列说法正确的有？（A、B、D）

A．重写方法不能比被重写方法限制有更严格的访问级别

B. 参数列表必须与被重写方法的相同

C. 返回类型必须与被重写方法的返回类型相同

1. 重写方法不能抛出新的异常或者比被重写方法声明的检查异常更广的检查异常。但是可以抛出更少，更有限或者不抛出异常。

2、下面哪些项描述不正确？（A、B）

A. 递归调用的返回值在逻辑判断后可以为不同类型数据

B. 返回值不允许为null

C. 返回值在return 后还可调用方法

D. 返回值只有在方法调用后才可获取

3、对于方法重写，下面哪些项说法是正确？（A、B、C）

A. 子类重写父类的方法，方法的参数必须一致

B. 子类方法的返回值类型可以与父类方法的返回值类型不一致

C. 子类方法的访问范围修饰符可以与父类方法的不一致

D. 子类方法抛出的异常必须与父类方法抛出的一致

4、下面哪些项不是运行结果？（B、C、D）

class Base {

Base() {

int i = 100;

System.out.println(i);

}

}

public class Pri extends Base {

static int i = 200;

public static void main(String argv[]) {

Pri p = new Pri();

System.out.println(i);

}

}

A.100 B.200 C.100 D.200

200 100 100

5、以下说法正确的是？（A、B）

A.数组是协变的，即可变返回类型 B. .容器类是不可协变的

C. 枚举可协变 D. String是协变的

6、下列哪些项是泛型的优点？（ ABCD ） 。

A. 不用向下强制类型转换 B. 代码容易编写

C. 类型安全 D. 运行速度快

7、父类型中f方法的返回值类型为List,子类重写父类中的f方法 其返回值类型可以定义为一下哪种类型（A B）

A ArrayList B LinkedList C HashMap D HashSet

8、一下关于return语句的叙述错误的是（ABC）？

A. 没有return语句的自定义函数在执行结束时不能返回的调用处

B. 一个自定义函数中必须有一条return语句

C. 定义成void类型的函数中可以有带返回值的return语句

D. 一个自定义函数中可以根据不同的情况设置多条return语句

9、实现了协变返回类型的子类覆盖（即重写）基类方法时，以下符合重写规则的是？（A、B、D）

A. 重写方法不能比被重写方法限制有更严格的访问级别。

B. 参数列列表必须与被重写方法的相同;

C. 能重写被标识为final的方法

D. 如果一个方法不能被继承，则不能重写它。

1. 填空题（每空1分，共10分）
2. 即子类覆盖（即重写）基类方法时，返回的类型可以是基类方法返回类型的\_\_\_\_\_\_\_\_\_\_。(子类)

2、返回值语法 填写所在位置的中文说明：public \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ play(){\_\_\_\_\_\_\_\_\_ “1”；}。(静态的，返回值类型或String,return)

3、所谓协变返回类型：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

(就是在导出类中的被覆盖方法可以返回基类方法的返回类型的某种导出类型。)

4、什么样的方法不能被重写\_\_\_\_\_\_\_\_\_(final方法)

5、Java泛型是不变的，可有时需要实现协变，在两个类型之间建立某种类型的向上转型关系，用\_\_\_\_\_\_来解决此问题。(通配符）

6、单独的T 代表一个\_\_\_\_\_\_ ，而 Class<T>代表这个类型所对应的\_\_\_\_\_\_， Class<？>表示类型\_\_\_\_\_\_（类型，类，不确定的类）

7、 可变返回方法指的是一个子类在覆盖超类的方法的时候，方法的返回值可以被一个\_\_\_\_\_的类型所替代。(更窄)

8、3\*0.1==0.3返回值是什么?（false，因为有些浮点数不能完全精确的表示出来。）

9、Math.round(11.5) 等于多少？Math.round(-11.5)等于多少？Math.round(11.5)的返回值是12，Math.round(-11.5)的返回值是-11。

10、指的是子类中的成员函数的返回值类型不必严格等同于父类中被重写的成员函数的返回值类型，可以使基类方法返回类型的子类。(协变返回类型/可变返回类型)

11、 协变返回类型仅在子类方法的返回类型是父类方法返回类型的 时才被容许。（继承\子类）

1. 判断题
2. Java中的协变可变返回类型体现了里氏代换原则（任何基类可以出现的地方，子类一定可以出现）（ T ）

2、增加可变返回类型后，重写方法时，可以自由地获得更具体的返回类型，有助于避免混淆类层次结构中存在的类型转换，从而使代码易读，可用且可维护，帮助防止返回时运行时ClassCastExceptions（T）

**3、方法中可以设定多个返回值。（ N ）**

**4、在Java语言中，当子类继承父类时，子类中重写的方法可以返回父类方法返回值类型的子类。（ T ）**

**5、 在Java语言中，JDK5以后才开始支持协变返回类型。（ T ）**

**6、重写方法时不允许降低方法的访问权限。（ T ）**

**7、多态分三种，编译时多态，运行时多态，引用时多态（F）**

**8、数组是协变的（ T ）**

**9、容器类是协变的。（ F）**

**10、协变类型参数，在非泛型代码中，参数类型不能随子类型发生变化。方法只能重载不能重写（T）**

**11、 泛型是可变的（ T ）**

**12、 Java中数组是协变的，可以向子类型的数组赋予基类型的数组引用（T）**

**13、 泛型通过通配符来完成可变返回值类型（协变）（T）**

**14、 数组具备内建的可变返回值类型 (T）**

**15、jdk中的List类是个泛型类 （T）**

**16、 泛型是Java SE 1.5的新特性，泛型的本质是参数化类型，也就是说所操作的数据类型被指定为一个参数（T）**

**17、 泛型的声明，必须在方法的修饰符（public,static,final,abstract等）之后，返回值声明之前 （T）**

**18、 泛型的类型参数只能是类类型（包括自定义类），可以是简单类型（F）**

**19、 在Java 5之前，为了让类有通用性，往往将参数类型、返回类型设置为Object类型，当获取这些返回类型来使用时候，必须将其“强制”转换为原有的类型或者接口，然后才可以调用对象上的方法（T）**

**20、Java可变返回值类型是jdk5.0以后版本支持。（ 正确 ）**

**21，Java中重写方法时要求方法返回值的类也必须完全相同（错误）**

**22、**Overloaded的方法可以改变返回值的类型（）（T）

**23、 在Java SE5中导出类中的被覆盖方法可以返回基类方法的返回类型的某种导出类型。（ T ）**

**24、 在Java的最新版本中，要求子类方法如果要覆盖超类的某个方法，必须具有完全相同的方法签名，包括返回值必须完全一样（F）**

**25、 协变返回类型仅在子类方法的返回类型是父类方法返回类型的继承时才被容许（T）。**

五、简答题

1、什么是协变可变返回类型？

答案：在Java1.4及以前，子类方法如果要覆盖超类的某个方法，必须具有完全相同的方法签名，包括返回值也必须完全一样。

Java5.0放宽了这一限制，只要子类方法与超类方法具有相同的方法签名，或者子类方法的返回值是超类方法的子类型，就可以覆盖。

注意："协变返回(covariant return)"，仅在subclass（子类）的返回类型是superclass（父类）返回类型的extension（继承）时才被容许。

2、请简述有返回值方法与无返回值方法的主要特点（5分）

答案：（1）有返回值方法 方法名前必须设定返回值类型修饰 并且在方法体结束前添加return关键词 并设定具体返回值

（2）. 无返回值方法 方法名前添加void修饰符 方法体内不允许出现return关键字

3、jdk 5.0后，重写规则是什么？（5分）

答案：

1. 重写方法不能比被重写方法限制有更严格的访问级别。

2. 参数列表必须与被重写方法的相同。

3. 返回类型必须与被重写方法的返回类型相同。

4. 重写方法不能抛出新的异常或者比被重写方法声明的检查异常更广的检查异常。但是可以抛出更少，更有限或者不抛出异常。

5. 不能重写被标识为final的方法。

6. 如果一个方法不能被继承，则不能重写它。如private方法

4、Java语言提出了“儿子比父亲强”的理论，jdk 5.0后，重写规则什么地方体现了该理论？（5分）

答案：

1. 重写方法不能比被重写方法限制有更严格的访问级别。

2. 重写方法不能抛出新的异常或者比被重写方法声明的检查异常更广的检查异常。但是可以抛出更少，更有限或者不抛出异常。

3. 返回类型必须与被重写方法的返回类型相同或者是其子类。

5、请列举说明什么是协变返回值类型。

答案： 协变返回类型表示在导出类的被覆盖方法可以返回基类方法的返回类型的某种导出类型

6、多态的好处有哪些（任意写出两点即可）

答案：可替换性、可扩充性、接口性、灵活性、简化性

7、简述什么是可变返回类型

答：在面向对象程序设计中，协变返回类型指的是子类中的成员函数的返回值类型不必严格等同于父类中被重写的成员函数的返回值类型，而可以是更 "狭窄" 的类型。  
　　Java 5.0添加了对协变返回类型的支持，即子类覆盖（即重写）基类方法时，返回的类型可以是基类方法返回类型的子类。协变返回类型允许返回更为具体的类型

8. import java.io.ByteArrayInputStream;

import java.io.InputStream;

class Base

{

//子类Derive将重写此方法，将返回类型设置为InputStream的子类

public InputStream getInput()

{

　　return System.in;

}

}

public class Derive extends Base

{

@Override

public ByteArrayInputStream getInput()

{

return new ByteArrayInputStream(new byte[1024]);

}

public static void main(String[] args)

{

Derive d=new Derive();

System.out.println(d.getInput().getClass());

}

}

程序输出：

class java.io.ByteArrayInputStream

9、Java中的泛型是什么 ? 使用泛型的好处是什么?

这是在各种Java泛型面试中，一开场你就会被问到的问题中的一个，主要集中在初级和中级面试中。那些拥有Java1.4或更早版本的开发背景的人 都知道，在集合中存储对象并在使用前进行类型转换是多么的不方便。泛型防止了那种情况的发生。它提供了编译期的类型安全，确保你只能把正确类型的对象放入 集合中，避免了在运行时出现ClassCastException。

10. 泛型的使用

Java泛型编程是JDK1.5版本后引入的。泛型让编程人员能够使用类型抽象，通常用于集合里面。

只要在上例中将第1行代码改成如下形式，那么就会在编译list.add(100)时报错。

List<String> list = new ArrayList<String>();

通过List<String>，直接限定了list集合中只能含有String类型的元素

11 泛型的好处

a. 类型安全

b. 消除强制类型转换

c. 提高性能

12. 泛型使用的注意事项

a. 泛型的类型参数只能是类类型（包括自定义类），不能是简单类型

b. 泛型的类型参数可以有多个

c. 不能对确切的泛型类型使用instanceof操作。如下面的操作是非法的，编译时会出错

d. 不能创建一个确切的泛型类型的数组

13. Java的泛型是如何工作的 ? 什么是类型擦除 ?如何工作？

a.类型检查：在生成字节码之前提供类型检查

b.类型擦除：所有类型参数都用他们的限定类型替换，包括类、变量和方法（类型擦除）

c.如果类型擦除和多态性发生了冲突时，则在子类中生成桥方法解决

d.如果调用泛型方法的返回类型被擦除，则在调用该方法时插入强制类型转换

14. Java中的Object、T（泛型）、?区别

Object范围非常广，而T从一开始就会限定这个类型（包括它可以限定类型为Object）。

Object由于它是所有类的父类，所以会强制类型转换，而T从一开始在编码时（注意是在写代码时）就限定了某种具体类型，所以它不用强制类型转换

15、、什么可变（协变）返回值

答案：**导出类**（子类）覆盖（即重写）**基类**（父类）方法时，返回的类型可以是基类方法返回类型的子类

16、阅读代码题

查看一下代码中是否有错误 如果有请指出 如果没有请填写输出结果

class Grain{

public String toString(){

return "Grain";

}

}

class Wheat extends Grain{

public String toString(){

return "Wheat";

}

}

class Mill{

Grain process(){

return new Grain();

}

}

class WheatMill extends Mill{

Wheat process(){

return new Wheat();

}

}

public class CovariantReturn {

public static void main(String [] args){

Mill m = new Mill();

Grain g = m.process();

System.out.println(m.process());

m = new WheatMill();

System.out.println(m.process());

}

}

答案：输出结果为Grain Wheat

17、return语句的作用：?

1. 返回一个值，这个值可以是任意类型。
2. 使程序返回到操作系统（即终止程序）

18、怎么理解Java中的可变（协变）返回类型？

答案：在面向对象程序设计中，协变返回类型指的是子类中的成员函数的返回值类型不必严格等同于父类中被重写的成员函数的返回值类型，而可以是更 "狭窄" 的类型。

Java 5.0添加了对协变返回类型的支持，即子类覆盖（即重写）基类方法时，返回的类型可以是基类方法返回类型的子类。协变返回类型允许返回更为具体的类型。

19. 对于是否可以实现可变返回类型，Java1.4以前版本（包括1.4）和之后的版本有什么不同？

答案：在Java1.4及以前，子类方法如果要覆盖超类的某个方法，必须具有完全相同的方法签名，包括返回值也必须完全一样。

Java5.0放宽了这一限制，只要子类方法与超类方法具有相同的方法签名，或者子类方法的返回值是超类方法的子类型，就可以覆盖。

注意："协变返回(covariant return)"，仅在subclass（子类）的返回类型是superclass（父类）返回类型的extension（继承）时才被容许。

六、程序实现题

1、编写代码，根据实际情况的不同返回更合适的不同类型输入流

|  |
| --- |
| class Base  {  //子类Derive将重写此方法，将返回类型设置为InputStream的子类  public InputStream getInput()  {  　　return System.in;  }  }  public class Derive extends Base  {    @Override  public ByteArrayInputStream getInput()  {    return new ByteArrayInputStream(new byte[1024]);  }  public static void main(String[] args)  {  Derive d=new Derive();  System.out.println(d.getInput().getClass());  }  } |

2、递归调用：利用递归调用方法求1-100的和?

（10分）

|  |
| --- |
| public class Digui {  public static int digui0 (int i){  if(i<=0){  System.out.println("参数非法！");  return 0;  }else if(i==1){  return 1;  }  return i+digui0(i-1);  //100+99+98+97+.....+1  }  public static void main(String[] args) {  // TODO Auto-generated method stub  System.out.println(digui0(100));  }  } |

3、通过代码演示协变返回类型（10分）

|  |
| --- |
| class Base  {  //子类Derive将重写此方法，将返回类型设置为InputStream的子类  public InputStream getInput()  {  　　return System.in;  }  }  public class Derive extends Base  {  @Override  public ByteArrayInputStream getInput()  {  return new ByteArrayInputStream(new byte[1024]);  }  public static void main(String[] args)  {  Derive d=new Derive();  System.out.println(d.getInput().getClass());  }  } |

4、写一个java程序：主要体现父类子类见得继承关系。父类：鸟，子类：麻雀、鸵鸟、鹰。子类继承父类的一些特点，如都是鸟的话就都会有翅膀、两条腿等，但它们各自又有各自的特点(写出一个继承关系，并且能输出到控制台即可)

（10分）

|  |
| --- |
| class Birds {  int leg = 2;  int wings = 2;  public void fly() {  System.out.println("鸟儿能够飞翔");  }  public void grow() {  System.out.println("鸟儿都有两只翅膀和两条腿");  }  }  class Sparrow extends Birds {  public void printAge(int age) {  System.out.println("鸟儿的年龄是：" + age);  }  public void printHeight(int height) {  System.out.println("鸟儿飞行的高度是：" + height);  }  }  class Ostrich extends Birds {  public void printWight(int weight) {  System.out.println("鸵鸟的体重是：" + weight);  }  public void printSpeed(int speed) {  System.out.println("鸵鸟的速度是：" + speed);  }  }  class Eagle extends Birds {  public static void hunt() {  System.out.println("鹰会捕食");  }  public static void height(int height) {  System.out.println("鹰飞翔的高度是：" + height);  }  }  public class BirdTest {  public static void main(String[] args) {  Birds birds = new Birds();  birds.fly();  birds.grow();  //父类实例化调用  System.out.println(); //这是一个换行  Birds birdsOne = new Sparrow();  birdsOne.grow();  birdsOne.fly();  //子类调用父类的方式  ((Sparrow) birdsOne).printAge(5);  ((Sparrow) birdsOne).printHeight(200);  //子类调用  System.out.println(); //这是一个换行  Ostrich ostrich = new Ostrich();  ostrich.fly();  ostrich.grow();  ostrich.printSpeed(60);  ostrich.printWight(100);  //直接调用static方法  System.out.println(); //这是一个换行  Eagle.hunt();  Eagle.height(800);  }  } |

5、编写一个泛型案例

public class Generic<T>{

//key这个成员变量的类型为T,T的类型由外部指定

private T key;

public Generic(T key) { //泛型构造方法形参key的类型也为T，T的类型由外部指定

this.key = key;

}

public T getKey(){ //泛型方法getKey的返回值类型为T，T的类型由外部指定

return key;

}

}

6、请列举一个java可变返回值类型的实力

（10分）

|  |
| --- |
| class A {}  class B extends A {}  class Base  {  A fun()  {  System.out.println("Base fun()");  return new A();  }  }  class Derived extends Base  {  B fun()  {  System.out.println("Derived fun()");  return new B();  }  }  public class Main  {  public static void main(String args[])  {  Base base = new Base();  base.fun();  Derived derived = new Derived();  derived.fun();  }  }  } |

7、在一个方法中返回多个值

public class NewClass {  
public static void main(String[] args){  
int[] array = {0,1};  
System.out.println(a(array)[0]);  
System.out.println(a(array)[1]);  
}

8、编程实现Java中的可变返回数据类型。要求编写简单的父类和子类，父类方法和子类方法，并在调用子类后验证返回的是父类返回类型的子类。

（10分）

|  |
| --- |
| package com.mufeng.theeighthchapter;    class Grain {  @Override  public String toString() {  // TODO Auto-generated method stub  return "Grain";  }  }    class Wheat extends Grain {  @Override  public String toString() {  // TODO Auto-generated method stub  return "Wheat";  }  }    class Mill {  Grain process() {  return new Grain();  }  }    class WheatMill extends Mill {  Wheat process() {  return new Wheat();  }  }    public class CovariantReturn {  public static void main(String[] args) {  Mill m=new Mill();  Grain g=m.process();  System.out.println(g);    m=new WheatMill();  g=m.process();  System.out.println(g);  }    } |