1.Java语言中类的继承性有什么特点?

a.单一继承性：子类只能有一个超类，而超类可以有多个子类；

b.子类继承超类的所有成员；

c.子类可以创建自己的成员；

d.子类不能继承超类的构造器，只能在构造器中通过super()调用超类的构造器；

e.子类的构造器首先要调用超类的构造器；

f.多态性①：子类的成员隐藏和覆盖超类中的相同成员；

g.多态性②：超类的对象可以对子类的实例引用；

h.由abstract和final修饰的类指示类的是否必须或不能被继承。

2.子类可以继承超类的什么，不能继承的如何在子类中访问？

子类可以继承超类的所有成员，这些成员就像子类创建的成员一样的使用。但是超类的构造器是不能被继承的，它是通过在子类的构造器中super()的形式调用的，并由它来对被继承的超类成员实例化。

3.子类对超类的扩展体现在哪些方面？

a.子类可以继承超类的所有成员，也可以创建自己的类成员，比如超类Animal，其子类dog可以继承超类的tall，age等成员，也可以创建自己的bark()等成员。

b.子类的成员隐藏和覆盖超类中的相同成员。

c.超类的对象可以对子类的实例引用。

4.方法覆盖是什么？与方法过载有什么区别？

方法过载是指在同一个类中，用新一种方法取代原有的方法，它们的方法名字相同。通过方法过载，可以直接从原有方法中衍生能够完成所要求的功能的新的方法。在过载一个方法时，必须为过载后的方法建立新的构造方法，以初始化示例变量。

方法覆盖是指在子类中，利用与超类有完全相同的方法名、返回类型和参数表，重新实现超类的成员方法。

5.在Java中有哪些是多态性的表现？举例说明。

a.子类的成员隐藏和覆盖超类中相同的成员；

b.超类的对象可以对子类的实例引用。

|  |
| --- |
| class ClassA{  void CallMe(){  System.out.println(“在ClassA中的callMe()方法”);  }  }  class ClassB{  void CallMe(){  System.out.println(“在ClassB中的callMe()方法”);  }  }  public class TestConvert{  public static void main(String args[]){  ClassA vA = new ClassB();  va.callMe();  }  } |
| 运行结果:  在ClassB中的callMe()方法 |

6.用final和abstract修饰的类各有什么特点？

①final类不能被继承(又称为终结类)。

②抽象类是指由abstract修饰符声明的类，它的方法中有未实现的方法，即抽象方法。与final类相反，abstract类必须被子类继承。当一个类的定义完全表示抽象概念时，它不能够被实例化为一个对象。

7.用final和abstract修饰的方法各有什么特点？举例说明。

①final修饰的方法不能被覆盖。

②abstract修饰的方法被称为抽象方法，抽象方法不具备实际功能，它只能衍生出子类，抽象方法则由衍生子类时所覆盖。抽象方法只需声明，而不需实现。

|  |
| --- |
| abstract class AbstClass{  abstract void callMe();  void meToo(){  System.out.println(“AbstClass中的meToo()方法”);  }  }  class ClassD extends AbstClass{  void callMe(){  System.out.println(“ClassD中的callMe()方法”);  }  }  public class TestAbst{  public static void main(String args[]){  AbstClass varC = new ClassD();  varC.callMe();  varC.meToo();  }  } |
| 运行结果：  ClassD中的callMe()方法  AbstClass中的meToo()方法 |

8.举例说明Object类的方法equals()的使用情况。

equal()方法用来比较两个对象类是否相同，如果相同则方法返回值true，否则返回值false。比较的内容是两个对象的状态和功能上的相同，不是两个对象引用上的相同。而运算符“==”则是判断对引用上的是否相同。