**中软国际考题-笔试**

1. 单选题

1、用什么关键词修饰的类称为抽象类（A）

1. abstract;
2. interface;
3. extends;
4. static;

2、以下关于abstract的说法，正确的是（D）

1. abstract只能修饰类
2. abstract只能修饰方法
3. abstract类中必须有abstract方法
4. abstarct方法所在的类必须用abstract修饰

3、以下说法正确的是 ( B )

A 必须定义多个形参  B 可以没有  C 至少有一个  D 只能是简单变量

4、如下Java代码运行的结果是（ B ）

public class Test {

final int age;

public Test(){

age=10;

}

public static void main(String[ ] args) {

System.out.println(new Test().age);

}

}

A、 0

B、 10

C、 null

D、 抛异常

5、以下说法正确的有(D)

A拥有抽象方法的类为抽象类

B抽象类的方法时抽象方法

C继承抽象类的子类一定是抽象类

D继承抽象类的子类可以不重写父类的抽象方法

6、用abstract修饰的类为抽象类，他们（A）

A 只能用以派生新类，不能创建对象

B 只能用以创建对象，能不能派生新类

C 即可用以创建对象，也可用以派生新类

D 即不能用以派生新类，也不可用来创建对象

7、下列描述中正确的是（B）

抽象类关键词拼写

A. abstrict B. abstract C. absliact D. abstratc

8、下列关于抽象类描述错误的是（B）

A. 一个没有方法体的方法应该定义为抽象方法

B. 类中如果有抽象方法，该类可以定义为抽象类

C. 抽象类中可以定义抽象方法

D. 抽象类中允许编写实体方法。

9、java中抽象类描述正确的是（D）

A.抽象类因为是抽象的所以不允许被继承

B.抽象类可以实例化过程

C. 抽象类不存在构造方法

D.抽象类有构造方法

10、以下说法正确的是 ( B )

对于方法内得形参()

A 必须定义多个形参

B 可以没有

C 至少有一个

D 只能是简单变量

11、以下说法正确的是( C )

A 对于局部内部类，不能在class前加任何修饰符（public default private protected）来修饰 （static好像可以）

B 只要没有定义不带参数的构造函数，JVM都会为类生成一个默认的构造函数

C 向上转型中，父类与子类拥有相同名字的普通方法时，访问的是子类的成员方法

D单例模式中类的构造方法使用private修饰，声明为私有，这样就不能在类的外部使用new关键字来创建实例对象了

12、下列的说法错误的是（B）

A 抽象类必须被子类继承

B 对象向上转型以后，拥有相同名字的普通成员变量(静态变量)时,访问的是子类里面的

C super的主要功能是完成子类的调用父类的内容，并且只能放在子类构造方法的第一行

D 接口的方法必须定义为public权限，并且只能是抽象方法（不考虑jdk1.8）

13．下列有关抽象类的叙述正确的是哪项？（B）

A．抽象类中一定含有抽象方法

B．抽象类的声明必须包含abstract关键字

C．抽象类既能被实例化也能被继承

D．抽象类中不能有构造方法

14 现自：

1． interface Color { }

2. interface Weight { }

3． //insert code here

和以下足六个声明：

class Boat extends Color, extends Weight { }

class Boat extends Color and Weight { }

class Boat extends Color, Weight { }

class Boat implements Color, implements Weight { }

class Boat implements Color and Weight { }

class Boat implements Color, Weight { }

分别插入到第3行，有多少行可以编译？（B）

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

15. 用abstract修饰的类称为抽象类，它们：A

A） 只能用以派生新类，不能用以创建对象；

B） 只能用以创建对象，不能用以派生新类；

C） 既可用以创建对象，也可用以派生新类；

D） 既不能用以创建对象，也不可用来派生新类。

16、以下关于abstract的说法，正确的是？（D）

A. abstract只能修饰类

B．abstract只能修饰方法

C. abstract类中必须有abstract方法

D. abstarct方法所在的类必须用abstract修饰

17. 下列的说法错误的是（B）

A 抽象类必须被子类继承

B 对象向上转型以后，拥有相同名字的普通成员变量(静态变量)时,访问的是子类里面的

C super的主要功能是完成子类的调用父类的内容，并且只能放在子类构造方法的第一行

D 接口的方法必须定义为public权限，并且只能是抽象方法（不考虑jdk1.8）

1. 多选题
2. 下列抽象类语法正确的是？（A、B、E）
3. abstract class A{};
4. abstract public class A{};
5. static abstract public class A{};
6. abstract static public class A{};
7. public abstract class A{};
8. 以下关于接口，抽象类的说法正确的是（A、B、C、F）
9. 抽象类只能被继承而且只能被单继承
10. 接口需要被实现，而且可以多实现
11. 抽象类中可以定义非抽象的方法，子类可以直接使用
12. 非抽象的子类实现接口可以不重写所有的抽象方法
13. 接口不可以继承另一个接口
14. 抽象类可以继承另一个抽象类
15. 抽象类可以创建对象，而接口不可以创建对象

3、以下方法错误的是（A、D）

A 覆盖方法包括成员方法的覆盖和成员变量的覆盖

B 成员方法的覆盖是多态的一种表现形式

C 子类可以调用父类中被覆盖的方法

D 任何方法都可以被覆盖

4、在java中，能实现多继承的方式（ B、D ）

1. 父类
2. 接口

C. 同步

D. 抽象类

1. 下列选项中，关于Java的抽象类和抽象方法说法正确的是（ AC ）
2. 抽象类中可以含有0个或多个抽象方法
3. 抽象类中不可以有构造方法;
4. 一个类中若有抽象方法，则这个类必为抽象类;
5. 类必须重写父类所有的抽象方法;

**6、下列选项中，关于Java的抽象类和抽象方法说法正确的是（  AC  ）。**

**A** 抽象类中可以含有0个或多个抽象方法

**B** 抽象类中不可以有构造方法

**C** 一个类中若有抽象方法，则这个类必为抽象类

**D** 子类必须重写父类所有的抽象方法

7、下面哪些项描述不正确？（A、B）

1. 抽象类不允许被抽象子类继承
2. 被继承的子类其构造方法会被抽象父类覆盖
3. 抽象的子类重写所有的构造方法，这时候子类是一个具体的类
4. 抽象类的实例化靠具体的子类实现，是多态的方式。

8、下面说法不正确得是(A C D E)   
A 不能和abstract一起使用的修饰符有static final public final   
B 抽象类之间是继承关系,接口之间也是。   
C 接口不能存在普通方法和常量   
D局部内部类public，private修饰符修饰 default/static  
E 接口不可以直接实例化，抽象类可以

9、下面有那些选项是对的（B C E）   
A instanceof是一个二元修饰符，作用是用来判断左边的对象是否是右边类的实例，返回的是布尔类型   
B private protected 和static都不可以修饰interface接口 （接口的修饰符只有public）  
C this.属性名访问的是本类中的属性,如果本类中没有,就从父类中继续查找   
D对象是对类的抽象化   
E 接口中可以有常量

10、下列选项中，关于Java的抽象类和抽象方法说法正确的是（  AC  ）

A抽象类中可以含有0个或多个抽象方法   
B抽象类中不可以有构造方法

C一个类中若有抽象方法，则这个类必为抽象类

D子类必须重写父类所有的抽象方法

11、什么情况下，使用抽象类(ABC)

A.类中包含一个明确声明的抽象方法;

B.类的任何一个父类包含一个没有实现的抽象方法;

C.类的直接父接口声明或者继承了一个抽象方法，并且该类没有声明或者实现该抽象方法

12. 抽象类和接口的区别是什么(ABCDE)

A． 接口是抽象类的变体，接口中所有的方法都是抽象的。

B． 接口可以多继承，抽象类不行

C． 接口定义方法，不能实现，而抽象类可以实现部分方法。

D． 接口中基本数据类型为static 而抽类象不是的。

E. 接口中不能含有静态代码块以及静态方法，而抽象类可以含有静态方法和静态代码块。

1. 填空题
2. 抽象类的抽象方法只允许\_\_\_\_\_声明 \_\_\_\_\_ ,而不允许\_\_\_\_ 实现 \_\_\_\_\_。
3. 如果子类中某个变量的变量名与它的父类中的某个变量完全一样，则称子类中的这个变量 \_\_\_\_\_\_\_\_\_\_ 了父类的同名变量（继承）
4. 抽象方法只有方法头，没有\_\_\_\_\_\_\_\_。(方法体)
5. 声明为\_\_\_\_\_\_\_\_\_\_的类不能是超类. (final)
6. 定义抽象类的关键字是\_\_\_\_\_\_\_。(abstract)
7. 抽象类中的方法不能用： static、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_修饰。(private、final)

7、用abstract修饰的类称为抽象类 只能用以 ，不能用以 。（派生新类 创建对象）

8、Java接口(以及[抽象类](https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E7%B1%BB" \t "_blank))一般用来作为一个类型的等级结构的起点。

9. 接口中每一个方法也是隐式抽象的，声明时同样不需要abstract关键字

1. 判断题
2. 抽象类不能用new来创建对象，必须产生其子类，由子类创建对象（T）
3. 如果一个非抽象类是一个抽象类的子类，它可以直接调用父类的抽象方法（F）（它必须具体实现父类的抽象方法，即重写父类的抽象方法）
4. 抽象类必须要有抽象方法（F）（抽象类可以没有抽象方法，如果有抽象方法子类必须实现）
5. Java只支持单重继承，一个类可以实现多个接口。（T）
6. 抽象方法只有方法头，没有方法体。（T）
7. 接口和抽象类是一回事。（F）
8. 接口中的抽象方法只能是public的，而抽象类中的抽象方法可以用别的修饰符修饰（ F ）
9. 抽象方法不能用final来修饰 ，可以用satic来修饰（ F ）
10. 对计算公式已不需要更改的数学函数，常用public static final 来修饰（ T ）
11. 抽象类必须被子类继承（ T ）
12. 抽象方法要被实现，所以不能是静态的，也不能是私有的（ T ）
13. 抽象类的子类是可以被实例化的（ F ）
14. 任何子类必须重写父类的抽象方法，或者声明自身为抽象类（ T ）
15. final和abstract能不能同时修饰一个类。（ F ）
16. 子类继承父类必须重写父类的方法除非子类也是抽象类。（T ）
17. 抽象方法不能含有方法体，并且必须定义在抽象类中。（ F ）
18. 抽象类是指在class前加使用abstract关键字修饰，且可以存在抽象方法和普通方法的类。（ T ）
19. 抽象类和接口都不能实例化，都没有构造方法。（ F ）
20. 抽象类中可以没有抽象方法。（ T ）
21. 抽象方法不能含有方法体，并且必须定义在抽象类中。（  F  ）
22. 抽象方法必须被子类重写。（ F ）
23. 抽象类的声明格式 class ??(抽象类关键字) ??(类名) {}。（ N ）
24. 抽象类中的方法只能定义成抽象的方法（F）
25. 抽象类中不一定有抽象函数，但抽象函数一定在抽象类中（T）
26. 如果一个类继承了一个抽象类，但是没有完全覆写父类的所有的抽象方法那么这个类也必须是抽象类（T）
27. 当类实现接口的时候，类要实现接口中所有的方法。否则，类必须声明为抽象的类。（ T ）
28. 在Java中，类的多继承是不合法，但接口允许多继承（ T ）
29. 在接口的多继承中extends关键字可以使用多次，在其后跟着继承接口。（ F ）

五、简答题

# 1、[java抽象类和普通类的区别](https://www.cnblogs.com/unknows/p/7660617.html)？

答案：抽象类不能被实例化。

抽象类可以有构造函数，被继承时子类必须继承父类一个构造方法，抽象方法不能被声明为静态。

抽象类里可以有抽象方法，抽象方法只需申明，而无需实现，抽象类中可以允许普通方法有主体

含有抽象方法的类必须申明为抽象类

抽象的子类必须实现抽象类中所有抽象方法，否则这个子类也是抽象类。

2、抽象类和抽象方法有什么特点？（5分）

答案：简单的理解，Java的抽象就是只声明行为接口（方法签名）而不完成具体的实现。用abstract修饰的类就是抽象类。抽象类是抽象方法的容器，如果某个类中包含有抽象方法，那么该类就必须定义成抽象类。抽象类中也可以包含有非抽象的方法甚至抽象类中可以没有抽象方法（只要一个类不应该有具体对象，就应该是抽象类，不一定是有抽象方法的），抽象类不可以直接实例化，只可以用来继承作为其他类的父类存在，抽象类的派生子类应该提供对其所有抽象方法的具体实现，如果抽象类的派生子类没有实现其中的所有抽象方法，那么该派生子类仍然是抽象类，只能用于继承，而不能实例化，但可以有构造函数（用于帮助子类快速初始化共有属性）。

抽象方法同样用abstract说明,抽象方法没有方法体，只有方法签名，实际上就是由抽象基类强制要求其派生子类必须实现的方法原型

3、什么是抽象类（5分）

答案：抽象类就是不能使用new方法进行实例化的类，即没有具体实例对象的类。含有抽象方法的类称为抽象类，其不能生成对象。抽象类是不完整的，它只能用作基类。在面向对象方法中，抽象类主要用来进行类型隐藏和充当全局变量的角色。

4、抽象类的特点（5分）

答案：（1）定义中含有抽象方法的类叫做抽象类。

    （2）抽象类用abstract关键字来修饰。

    （3）抽象类代表一种抽象的对象类型。

    （4）抽象类不能实例化。

（5）抽象类中可以有具体方法，可以没有抽象方法。（也就是说一个类中只要有一个方法是抽象方法那么这个类一定是抽象类，反过来，一个抽象类中可以没有抽象方法，可以带有具体

实现的方法）

    （6）一旦一个类中有抽象方法，那么这个类必须也要用abstract来修饰，代表这个类是抽象类，它是不能被实例化的。

5、抽象类存在有什么必要性

答案：1.因为抽象类不能实例化对象，所以必须要有子类来实现它之后才能使用。这样就可以把一些具有相同属性和方法的组件进行抽象，这样更有利于代码和程序的维护。

2.当又有一个具有相似的组件产生时，只需要实现该抽象类就可以获得该抽象类的那些属性和方法。

6、抽象类的子类可以被实例化么？

答案：抽象类不能被实例化，也就是不能用new关键字去产生对象，抽象方法只需声明，而不需实现，抽象类的子类必须覆盖所有的抽象方法后才能被实例化，否则这个子类还是个抽象类，里面的方法是不是抽象的没有本质影响。

7、简述Java中抽象类的特点？

答案：1、[抽象类](http://www.so.com/s?q=%E6%8A%BD%E8%B1%A1%E7%B1%BB&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)不能直接[实例化](http://www.so.com/s?q=%E5%AE%9E%E4%BE%8B%E5%8C%96&ie=utf-8&src=internal_wenda_recommend_textn)，并且对抽象类使用 new 运算符是编译时[错误](http://www.so.com/s?q=%E9%94%99%E8%AF%AF&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)。虽然一些[变量](http://www.so.com/s?q=%E5%8F%98%E9%87%8F&ie=utf-8&src=internal_wenda_recommend_textn)和值在编译时的[类型](http://www.so.com/s?q=%E7%B1%BB%E5%9E%8B&ie=utf-8&src=internal_wenda_recommend_textn)可以是抽象的，但是这样的变量[和值](http://www.so.com/s?q=%E5%92%8C%E5%80%BC&ie=utf-8&src=internal_wenda_recommend_textn)必须或者为 null，或者含有对非抽象类的[实例](http://www.so.com/s?q=%E5%AE%9E%E4%BE%8B&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)的引用(此非抽象类是从抽象类派生的)。  
2、允许(但不要求)抽象类包含[抽象成员](http://www.so.com/s?q=%E6%8A%BD%E8%B1%A1%E6%88%90%E5%91%98&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://wenda.so.com/q/_blank)。  
3、抽象类不能被密封

8、简述抽象类和接口的区别？（5分）

答案：抽象类是抽象方法和非抽象方法的集合，抽象类不能被实例化（不能创建该类的对象），抽象方法不能被static修饰。如果该类中含有抽象方法，那么这个类必须被定义成抽象类。

接口（interface）是抽象类的变体。接口不能被实例化，接口中不能写构造方法。接口中所有的常量都是 public static final修饰的，所有的方法都是 public abstract 修饰的，即使定义接口的时候不写这些修饰符，编译器在编译期间也会默认加上这些修饰符。

9、final和abstract关键字的作用（5分）

答案：final关键字

final（最终）是一个修饰符

1.final可以修饰类，函数，变量（成员变量，局部变量）

2.被final修饰后的类不可以被其他类继承

3.被final修饰后的函数不可以被重写

4.被final修饰后的变量不允许被再次赋值，final在对变量进行修饰时一定赋值，

被public final static多个修饰符修饰的的变量我们称它为常量

abstract关键字

abstract：是一个修饰符可以修饰方法，类。当我们多个类存在相同的功能，但是功能的主体不同，这时我进行向上抽取只抽取功能的定义不抽取功能的主体

10、抽象方法不能使用那些关键字（5分）

答案：private、final、static

11、abstract可以用来修饰什么？

答案：类和方法

12、请简述抽象类的成员特点（5分）

答案：成员变量：既可以是变量，也可以是常量。

构造方法：有。(用于子类访问父类数据的初始化。)

成员方法：既可以是抽象的，也可以是非抽象的。

# 13、什么是抽象类，以及抽象类的作用和用法（5分）

答案：抽象类自身不能创建对象，但是它的子类可以创建对象。

抽象它的子键字是abstract

在父类中定义的抽象方法，在子类中必须实现（即方法重写）

抽象方法没有方法体。

抽象方法必须定义在抽象类中。

14、**Java抽象类可以有构造函数吗？**

可以有，抽象类可以声明并定义构造函数。因为你不可以创建抽象类的实例，所以构造函数只能通过构造函数链调用（Java中构造函数链指的是从其他构造函数调用一个构造函数）

六、程序实现题

1、编写一个抽象类A，里面有抽象方法a，编写一个类B继承抽象类A，实现A的抽象方法a，在a中打印hello

|  |
| --- |
| **public** **abstract** **class** A {  **public** **abstract** **void** a();    }  **public** **class** B **extends** A {  @Override  **public** **void** a() {  // **TODO** Auto-generated method stub  System.***out***.println("hello");  }  } |

2、 设计一个系统：XXX门的实现过程：

流程：设计一张抽象的门Door，那么对于这张门来说，就应该拥有所有门的共性，开门和关门；

然后对门进行另外的功能设计,防盗、防水、防弹--bulletproof()、防火、防锈，要求：利用继承、抽象类、接口的知识设计该门（10分）

|  |
| --- |
| abstract class Door {  public abstract void openDoor ();  public abstract void closeDoor ();  }  interface method {    public abstract void guard ();  public abstract void waterproof ();  public abstract void bulletproof ();  public  abstract void fireproof ();  public abstract void rustproof ();  }  class NewDoor extends Door implements method{    public void guard () {  System.out.println("防盗");  }    @Override  public void closeDoor() {    // TODO Auto-generated method stub  System.out.println("关门");  }    @Override  public  void openDoor() {  // TODO Auto-generated method stub  System.out.println("开门");  }      @Override  public void bulletproof() {  // TODO Auto-generated method stub    System.out.println("防弹");  }    @Override  public void fireproof() {  // TODO Auto-generated  method stub  System.out.println("防火");  }    @Override  public void rustproof() {    // TODO Auto-generated method stub  System.out.println("防锈");  }    @Override  public void  waterproof() {  // TODO Auto-generated method stub  System.out.println("防水");  }    }    public class Demo3 {    /\*\*  \* @param args  \*/  public static void main(String[] args) {  // TODO  Auto-generated method stub  NewDoor nd = new NewDoor();  nd.guard();  nd.bulletproof  ();  nd.fireproof();  nd.closeDoor();  }    } |

3、设计一个抽象类,并演示它的使用 (三级)

（10分）

|  |
| --- |
| package TT;     //抽象类  abstract class 数学老师标准{  abstract void 讲数学();  abstract int add(int a,int b);  }     //第二级抽象类  abstract class 普通数学老师标准 extends 数学老师标准{  void 讲数学(){  System.out.println("我能讲数学");  }  }    //第三级抽象类  class 真正的数学老师 extends 普通数学老师标准{ //注意这里能不能用abstract修饰，修饰了以后会对上下的影响  //可以重写，也可以不重写  void 讲数学(){  System.out.println("我能讲高数");  }  int add(int a,int b){  return a+b;  }  }  class Test{    public static void main(String[] args) {      数学老师标准 t1=new 真正的数学老师();      int result=t1.add(44, 4444);      System.out.println("计算结果是:"+result);      t1.讲数学();    }  } |

4、定义一个抽象的"Role"类，有姓名，年龄，性别等成员变量

1）要求尽可能隐藏所有变量(能够私有就私有,能够保护就不要公有)，

再通过GetXXX()和SetXXX()方法对各变量进行读写。具有一个抽象的play()方法，

该方法不返回任何值，同时至少定义两个构造方法。Role类中要体现出this的几种用法。

2）从Role类派生出一个"Employee"类，该类具有Role类的所有成员（构造方法除外），

并扩展salary成员变量，同时增加一个静态成员变量“职工编号（ID）”。

同样要有至少两个构造方法，要体现出this和super的几种用法，还要求覆盖play()方法，

并提供一个final sing()方法。

3）"Manager"类继承"Employee"类，有一个final成员变量"vehicle"

在main()方法中制造Manager和Employee对象,并测试这些对象的方法。

abstract class Role {

private String name;

private int age;

private String sex;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getSex() {

return sex;

}

public void setSex(String sex) {

this.sex = sex;

}

public abstract void play ();

public Role (){

}

public Role (String name, int age, String sex){

this.name = name;

this.age = age;

this.sex = sex;

}

}

class Employee extends Role {

private double salary;

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

private static String ID;

public static String getID() {

return ID;

}

public static void setID(String iD) {

ID = iD;

}

public Employee () {

}

public Employee (String name, int age, String sex, double salary, String ID) {

super(name, age, sex);

this.salary = salary;

this.ID = ID;

}

@Override

public void play() {

// TODO Auto-generated method stub

System.out.println("员工编号"+this.getID());

}

final void sing() {

System.out.println("员工姓名"+this.getName()+"员工性别"+this.getSex()

+"员工年龄"+this.getAge()+"员工工资"+this.getSalary());

}

}

class Manager extends Employee {

final String vehicle;

public Manager (String name, int age, String sex, double salary, String ID, String vehicle) {

super(name, age, sex, salary, ID);

this.vehicle = "保时捷";

}

}

public class Demo2 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

Employee e = new Employee("张三", 27, "男", 3000,"001");

e.play();

e.sing();

Manager m = new Manager("李四", 29, "男", 4000, "002", "");

m.play();

m.sing();

System.out.println("坐车"+m.vehicle);

}

}

5、需求：公司中程序员有姓名、工号、薪水、工作内容。

\* 项目经理除了有姓名、工号、薪水、还有奖金、工作内容。

\* 对给出需求进行数据建模。

（10分）

|  |
| --- |
| \* 分析：  \* 在这个问题领域中，先找出设计的对象。  \* 通过提炼法。  \* 程序员：  \* 属性：姓名、工号、薪水  \* 行为：工作  \* 经理：  \* 属性：姓名、工号、薪水、奖金。  \* 行为：工作  \*  \* 程序员和经理不存在着直接继承关系，但是程序员和经理却具有共性内容。可以进行抽取。因为他们都是公司的雇员可以将程序员和经理进行抽取，建立体系。  \*/  //描述雇员  abstract class Employee{  private String name;  private String id;  private double pay;  Employee(String name,String id,double pay){  this.name = name;  this.id = id;  this.pay = pay;  }  public abstract void work();  }  //描述程序员  class Programmer extends Employee{  Programmer(String name,String id,double pay){  super(name,id,pay);  }  public void work(){  System.out.println("code ...");  }  }  //描述经理  class Manager extends Employee{  private int bouns;  Manager(String name,String id,double pay,int bounus){  super(name,id,pay);  this.bouns = bouns;  }  public void work(){  System.out.println("manager");  }  }  public class AbstractTest {  /\*\*  \* @param args  \*/  public static void main(String[] args) {  // TODO Auto-generated method stub  }  }  class Person{  private String name;  private int age;  Person(String name,int age){  this.name = name;  this.age = age;  }  public String getName(){  return name;  }  public void setname(String name){  this.name = name;  }  }  class Student extends Person{  Student(String name,int age){  super(name,age);  }  } |

6、编写程序描述兔子和青蛙

需求说明：使用面向对象的思想，设计自定义类描述兔子和青蛙。

实现思路

1)        根据共性，定义抽象的动物类

属性：名字、颜色

方法：吃饭，发出叫声

2)         定义兔子和青蛙继承动物类，并重写动物类的所有抽象方法

（10分）

|  |
| --- |
| **public** **abstract** **class** Animal {  **private** String name;  **private** String color;  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getColor() {  **return** color;  }  **public** **void** setColor(String color) {  **this**.color = color;  }  **public** **abstract** **void** shout();  **public** **abstract** **void** eat();  }  **public** **class** Frog **extends** Animal {  @Override  **public** **void** shout() {  System.***out***.println("呱呱呱");  }  @Override  **public** **void** eat() {  System.***out***.println("吃昆虫");  }  }  **public** **class** Rabbit **extends** Animal {  @Override  **public** **void** shout() {  System.***out***.println("叽叽叽");  }  @Override  **public** **void** eat() {  System.***out***.println("吃胡萝卜");  }  } |

7、抽象继承：创建一个抽象类Animal，并创建抽象方法，并用Dog类继承，并在Test类利用主方法调用重写方法及普通成员方法

（10分）

|  |
| --- |
| abstract class Animal {  public abstract void show();  public void method() {  System.out.println("method");  }  }  class Dog extends Animal {  public void show() {  System.out.println("show Dog");  }  }  class AbstractDemo2 {  public static void main(String[] args) {  //创建对象  Animal a = new Dog();  a.show();  a.method();  }  } |

8、使用抽象类和抽象方法模拟不同人种

世界上有很多人种，例如中国人、美国人等，他们所说的语言各不相同。不管是什么人种，他们都是人类

实现步骤:

(1) 创建一个抽象类Person，属性：name 和age

(2) 在Person 类中定义一个抽象方法say()

(3) 创建Chinese 类继承Person

(4) 创建American 类继承Person

（10分）

|  |
| --- |
| package aaaw;  public abstract class Person {  public String name;  public int age;  public abstract void say() ;  }  package aaaw;  public class Chiness extends Person{  @Override  public void say() {  // TODO Auto-generated method stub  System.out.println("说中文");  }  }  package aaaw;  public class American extends Person{  @Override  public void say() {  // TODO Auto-generated method stub  System.out.println("说英语");    }  } |