# JAVA面向对象

##### 1-5、单选题

有关Java中的类和对象，以下说法错误的是（B）（选择一项）

A同一个类的所有对象都拥有相同的特征和行为

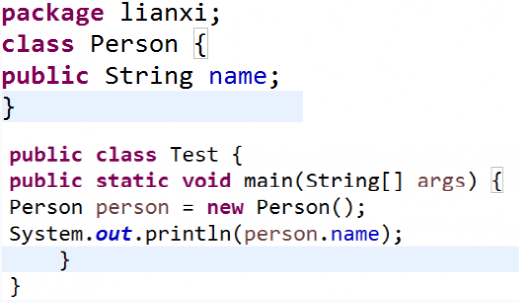
B类和对象一样，只是说法不同

C对象是具有属性和行为的实体

D类规定了对象拥有的特征和行为

##### 1-6单选题

在java中，以下程序的运行结果是（A）（选择一项）



A 输出：null

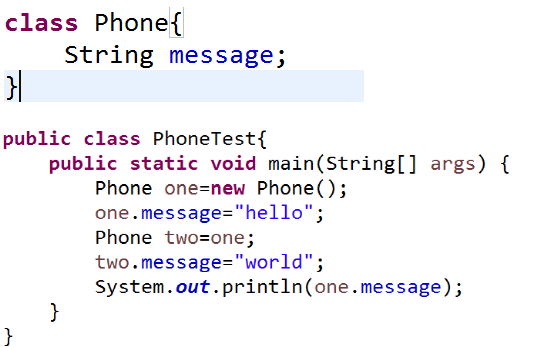
B 正常运行，但不会输出任何内容

C 编译出错，不能运行

D 能运行，但运行时会出现异常

##### 1-9 单选题

下面代码运行的正确结果是（C）（选择一项）



A 编译错误，无法正常运行

B 编译正确，但运行时产生错误

C hello

D world

##### 1-10. 单选题

（A）空间用于存储使用new关键字所创建的对象（选择一项）A

A 堆

B 栈

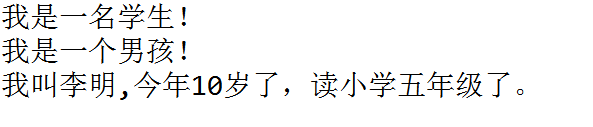
C 方法区

D 实例区

##### 1-11 编程练习

编写自定义Person类，根据提示以及效果图补全编辑器中代码。

程序运行参考效果图如下：



任务

1、创建Person类

       属性：名字（name），年龄（age）,年级（ grade）

       方法：1、无参无返回值的student方法，描述为：我是一名学生！

           2、带参数（性别sex）的方法，描述为：我是一个\*\*孩！（其中，\*\*为传入参数）

           3、无参无返回值的mySelf方法，介绍自己的姓名、年龄、年级（参数参考效果图）

2. 创建测试类

      实例化对象，传入参数，调用无参无返回值的student和mySelf方法及带参方法sex

public class Test {

public static void main(String[] args) {

//使用new关键字实例化对象

String name;

**int** age;

String grade;

**public** **void** student() {

System.***out***.println("我是一名学生");

}

**public** String sex(String sex) {

System.***out***.println("我是一名"+sex+"孩！");

**return** sex;

}

**public** **void** mySelf() {

System.***out***.println("我叫"+name+"，今年"+age+"岁了，读小学"+grade+"了");

}

//传入name、age、grade的参数值

//分别调用student、sex、mySelf方法

}

}

public class Person {

//属性：name、age、grade

Person p=**new** Person();

p.age=10;

p.grade="五年级";

p.name="李明";

p.student();

p.sex("男");

p.mySelf();

//创建无参无返回值的student方法，描述为：我是一名学生！

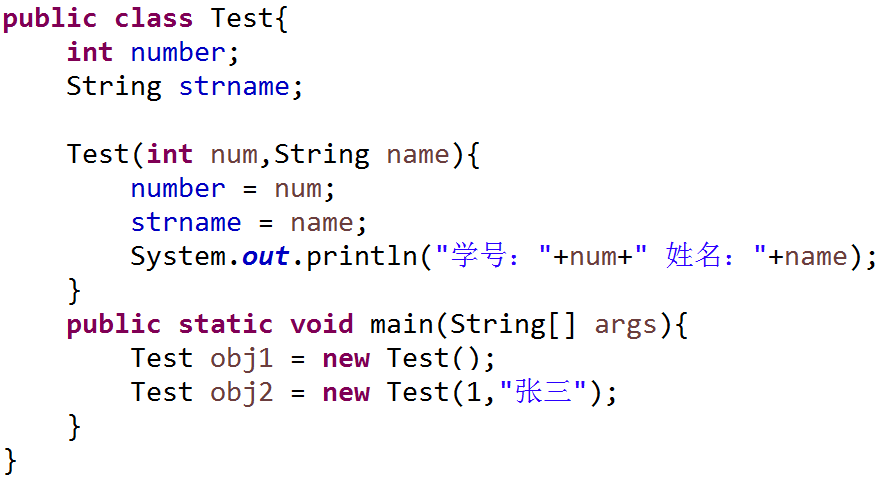
//创建带参（性别sex）的sex方法，描述为：我是一个\*\*孩！

//创建无参无返回值的mySelf方法，分别介绍姓名（name），年龄（age），年级（grade）

}

##### 2-3. 单选题

分析下面的Java代码，编译运行结果是（C）（选择一项）



A运行结果为：学号：1  姓名：张三

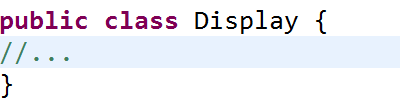
B运行结果为：学号：null 姓名：张三

C程序出现编译错误

D程序出现运行时异常

##### 2-4. 多选题

下面的哪几项是合法的构造方法重载（AC）（选择两项）



A public Display(String s){}

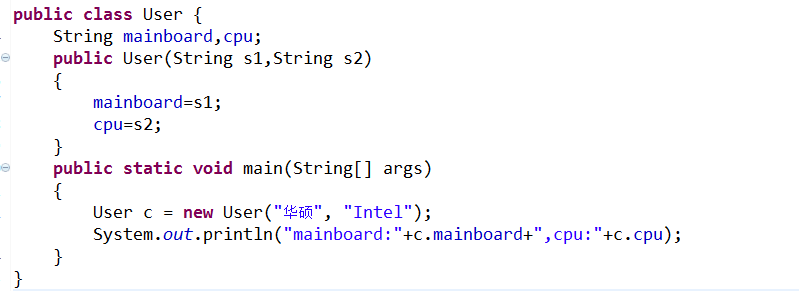
B public int display(int n1,int n2){}

C Display (int …a){}

D public display(Strnig s,int a){}

##### 2-5. 单选题

运行结果为（A）（选择一项）



A mainboard：华硕，cpu：Intel

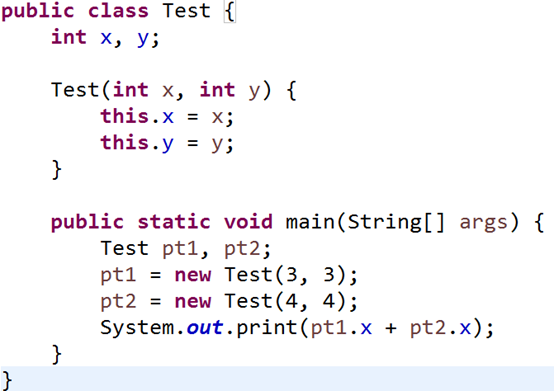
B mainboard：s1，cpu：s2

C mainboard：Intel，cpu：华硕

D 华硕，Intel

##### 2-7. 单选题

在Java中，以下程序编译运行后的输出结果为（D ）（选择一项）



A 6

B 3  4

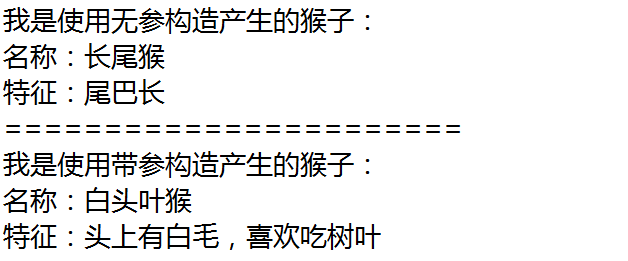
C 8

D 7

##### 2-9 编程练习

编写自定义猴子类，按照效果图，在编辑器中对应空白区域编写代码。

程序参考运行效果图如下：



任务

1、创建Monkey类

     属性：名称（name）和特征（feature）

     方法：

         1） 无参构造方法（默认初始化 name 和 feature 的属性值，属性值参考效果图）

         2） 带参构造方法，接收外部传入的参数，分别向 name 和 feature 赋值

2、创建测试类

      分别通过无参构造方法和带参构造方法，完成对象实例化实例化对象，并打印输出对象信息，输出格式如效果图。

public class Test {

public static void main(String[] args) {

// 调用无参构造方法实例对象

Monkey m1=**new** Monkey();

// 打印输出对象属性

System.***out***.println("我是使用无参构造产生的猴子：");

System.***out***.println("名称："+m1.name+"\n"+"特征："+m1.feature);

// 调用带参构造方法实例对象

Monkey m2=**new** Monkey("白头叶猴","头上有白毛，喜欢吃树叶");

// 打印输出对象属性

System.***out***.println("==============================");

System.***out***.println("我是使用带参构造产生的猴子：");

System.***out***.println("名称："+m2.name+"\n"+"特征："+m2.feature);

}

}

public class Monkey {

//属性：姓名（name）、特征（feature）

String name;

String feature;

//无参的构造方法（默认初始化name和feature的属性值，属性值参考效果图）

**public** Monkey() {

**this**.name="长尾猴";

**this**.feature="尾巴长";

}

//带参的构造方法（接收外部传入的参数，分别向 name 和 feature 赋值）

**public** Monkey(String name,String feature) {

**this**.name=name;

**this**.feature=feature;

}}

##### 2-10. 多选题

在Java中，下列关于this的说法错误的选项是（C）（选择两项）

A 在构造方法中如果使用this调用其他构造方法，只能是第一条语句

B 不能在构造方法中调用同一个类的其他构造方法

C 在构造方法在中如果使用this调用其他构造方法，语句可以放在任意位置

D 可以使用“this.方法名()”或“this.属性名”来引用当前对象的成员

# JAVA封装

##### 1-4. 单选题

当类中的一个成员方法被下面哪个修饰符修饰后，该方法只能在本类中被访问。（B）（选择一项）

A public

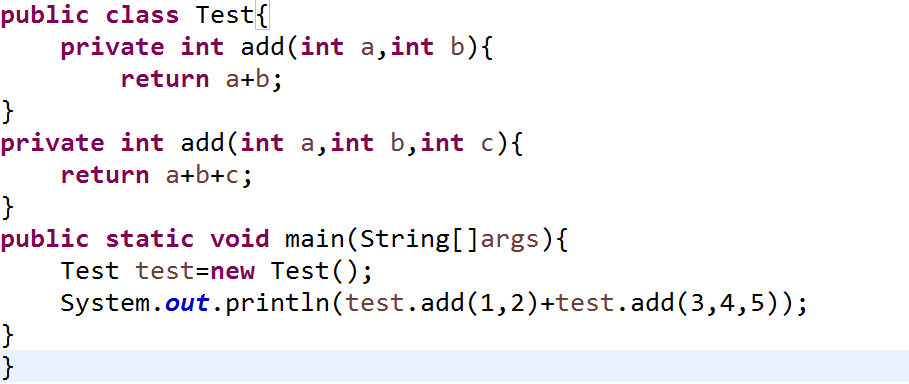
B private

C final

D default

##### 1-5. 单选题

运行以下Java代码,说法正确的是(A)



A 15

B 程序运行正常,但无输出

C 编译报错

D 运行报错

##### 1-6. 多选题

java封装的意义：（ABC）（选择三项）

A 防止使用者错误修改系统的属性

B 提高系统的独立性

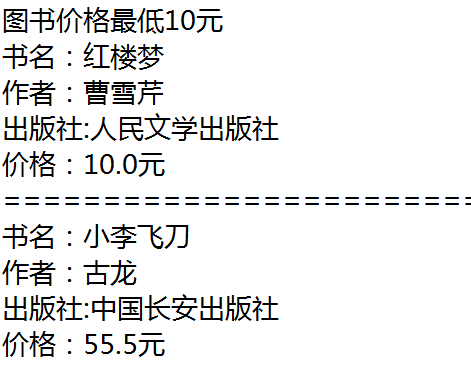
C 提高软件的可重用性

D 提高构建大型系统的风险

##### 1-7 编程练习

编写自定义类实现图书信息设置。请按照编辑器中的注释，在对应的空白区域填写对应的代码，完成程序设计。

运行参考效果如下所示：



任务

属性：书名、作者、出版社、价格

方法：信息介绍

要求：

1、 设计构造函数实现对属性赋值

2、 设置私有属性，get/set方法实现对属性的访问

3、 限定图书价格必须大于10，如果无效需进行提示，并强制赋值为10

4、 限定作者、书名均为只读属性

5、信息介绍方法描述图书所有信息

答案：

public class BookTest {

// 测试方法

public static void main(String[] args) {

//实例化对象，调用相关方法实现运行效果

Book book = **new** Book("大书","大作者","大出版社",100);

book.show(book);

// book.setName("小书");

// book.setAuther("小作者");

book.setPrice(8);

book.setPublish("小出版社");

book.show(book);

}

}

public class Book {

//私有属性：书名、作者、出版社、价格

**private** String name;

**private** String auther;

**private** String publish;

**private** **double** price;

**public** Book() {

}

**public** Book(String name, String auther, String publish, **double** price) {

**this**.name = name;

**this**.auther = auther;

**this**.publish = publish;

**this**.price = price;

}

**public** String getName() {

**return** **this**.name;

}

**private** **void** setName(String name) {

**this**.name = name;

}

**public** String getAuther() {

**return** **this**.auther;

}

**private** **void** setAuther(String auther) {

**this**.auther = auther;

}

**public** String getPublish() {

**return** **this**.publish;

}

**public** **void** setPublish(String publish) {

**this**.publish = publish;

}

**public** **double** getPrice() {

**return** **this**.price;

}

**public** **void** setPrice(**double** price) {

**if**(price<10) {

System.***out***.println("【无效信息，强制10元】");

**this**.price=10;

}**else** {

**this**.price = price;

}

}

**public** **void** show(Book book) {

System.***out***.println("书名："+book.getName());

System.***out***.println("作者："+book.getAuther());

System.***out***.println("出版社："+book.getPublish());

System.***out***.println("价格："+book.getPrice());

}

//通过构造方法实现属性赋值

/\*通过公有的get/set方法实现属性的访问，其中：

1、限定图书价格必须大于10，如果无效需进行提示，并强制赋值为10

2、限定作者、书名均为只读属性

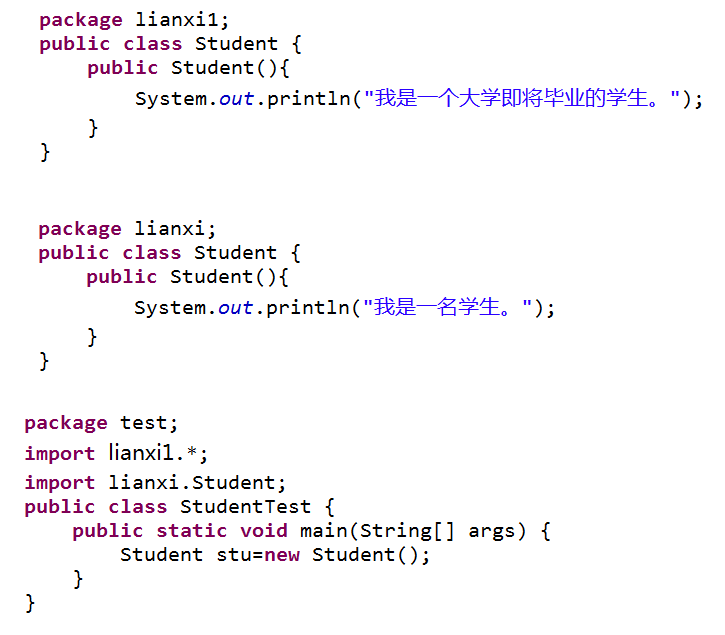
\*/

//信息介绍方法，描述图书所有信息

}

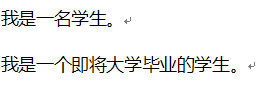
##### 1-10. 单选题

下面代码的运行结果是（B）（选择一项）



A 我是一名学生。

B 我是一个即将大学毕业的学生。

C 

D 编译错误

##### 1-11. 单选题

关于package+包名；的说法不正确的是(A）（选择一项）

A 一个java源文件可以有多个package语句

B 建议包名应该全部英文小写

C 建议包名命名方式：域名倒叙+模块+功能

D "package+包名；"必须放在java源文件中的第一行

##### 1-12. 单选题

下面关于import，class和package的声明顺序哪个是正确的（选择一项）A

A package，import，class

B class，import，package

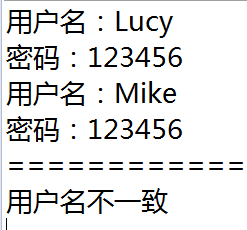
C import，package，class

D package，class，import

##### 1-13 编程练习

编写自定义类实现用户信息类。请按照编辑器中的注释，在对应的空白区域填写对应的代码，完成程序设计。

程序参考运行效果图如下：



任务

用户类：

 属性：用户名、密码

用户管理类：

       方法：用户信息验证

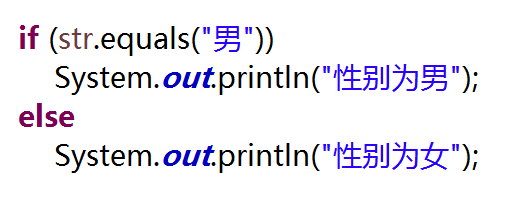
要求：

        1、 设计构造函数实现对属性赋值

        2、 设置私有属性，get/set方法实现对属性的访问

        3、用户信息验证判断两个用户信息是否一致

PS: 可以通过equals（）方法进行字符串内容的判断，如果内容相等返回值为true，反之为false，如当str代表用户性别时，可以通过如下代码，判断性别为“男”还是“女”



答案：

public class Test{

// 测试方法

public static void main(String[] args) {

//实例化对象，调用相关方法实现运行效果

User one = **new** User();

one.setUsername("Lucy");

one.setPassword("123456");

User two = **new** User();

two.setUsername("Mike");

two.setPassword("123456");

System.***out***.println("用户名："+one.getUsername());

System.***out***.println("密码："+one.getPassword());

System.***out***.println("用户名："+two.getUsername());

System.***out***.println("密码："+two.getPassword());

System.***out***.println("=====================");

UserManager u=**new** UserManager();

u.checkUser(one, two);

}

}

public class User{

//定义属性用户名、密码

**private** String username;

**private** String password;

**public** User() {

}

**public** User(String username, String password) {

**this**.username = username;

**this**.password = password;

}

**public** String getUsername() {

**return** **this**.username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** **this**.password;

}

**public** **void** setPassword(String password) {

**this**.password=password;

}

}

public class UserManager{

// 用户信息验证的方法

public String checkUser(User one, User two) {

String hint = **null** ;

// 判断用户名是否为空，是否一致

**if**(one.getUsername().equals(hint))

System.***out***.println("用户名为空");

**else** **if**(one.getUsername().equals(two.getUsername()))

System.***out***.println("用户名："+one.getUsername());

**else**

System.***out***.println("用户名不一致");

// 判断密码是否为空，是否一致

**if**(one.getPassword().equals(hint))

System.***out***.println("密码为空");

**else** **if**(one.getPassword().equals(two.getPassword()))

System.***out***.println("密码："+one.getPassword());

**else**

System.***out***.println("密码不一致");

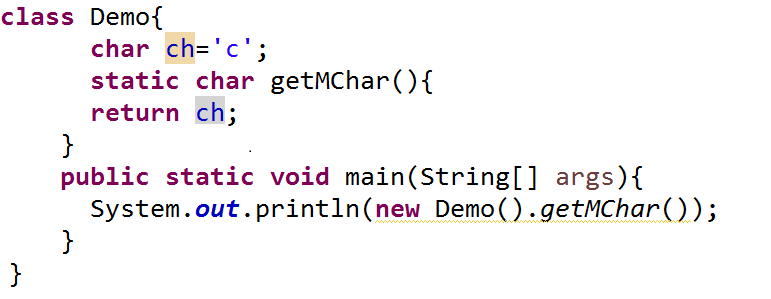
**return** hint;

return hint;

}

##### 2-3. 单选题

以下代码运行结果为：（B）.（选择一项）



A 编译出错

B 输出：c

C 编译正常，运行时报错

D 编译正常，运行时无输出结果

##### 2-4. 单选题

下列关于static的说法不正确的是（A）（选择一项）

A 可以定义静态方法，但不能定义静态变量

B class前不可以是用static作为修饰符

C static可以把普通的成员方法变为一个静态方法

D static可以把一个普通方法变为一个类方法

##### 2-5. 单选题

关于成员方法（也称实例方法）和类方法（也称静态方法），以下描述正确的是（A）（选择一项）

A 成员方法只能访问成员变量

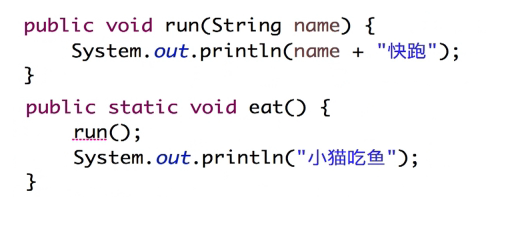
B 类方法既可以直接访问本类的类变量，也可以直接访问本类的成员变量

C 类方法在类外只能通过类名来调用

D成员方法在类外只能通过对象来调用

##### 2-6. 多选题

下面这段代码中，报错的地方原因是（A）（选择两项）



A 没有传入参数

B 没有使用类方法来调用

C 不能使用静态引用去访问非静态的方法

D 没有实例化对象

##### 2-8 编程练习

请根据效果图以及任务要求完成代码。

程序参考运行效果图如下：



任务

创建类Code，类中编写构造块、静态代码块以及构造方法

创建CodeBlock，类中编写的构造块、静态代码块以及构造方法

在主函数中测试他们的执行的优先顺序（运行结果参见效果图）

答案：

public class CodeBlock{

//创建CodeBlock的构造块

{

System.***out***.println("CodeBlock的构造块");

}

//创建CodeBlock的静态代码块

**static** {

System.***out***.println("CodeBlock的静态代码块");

}

//创建CodeBlock的构造方法

**public** CodeBlock() {

System.***out***.println("CodeBlock的构造代码");

}

**public** **static** **void** main(String[] args){

//编写代码测试运行优先顺序，运行结果参见效果图

System.***out***.println("我是CodeBlock的主方法");

System.***out***.println("产生Code实例的对象");

Code c1=**new** Code();

System.***out***.println("产生CodeBlock实例的对象");

CodeBlock c2=**new** CodeBlock(); }

}

class Code{

//创建外部类Code构造块（构造块）

{

System.***out***.println("Code的构造块");

}

//创建外部类Code静态代码块（静态代码块）

**static** {

System.***out***.println("Code的静态代码块");

}

//创建外部类Code构造方法（构造方法）

**public** Code(){

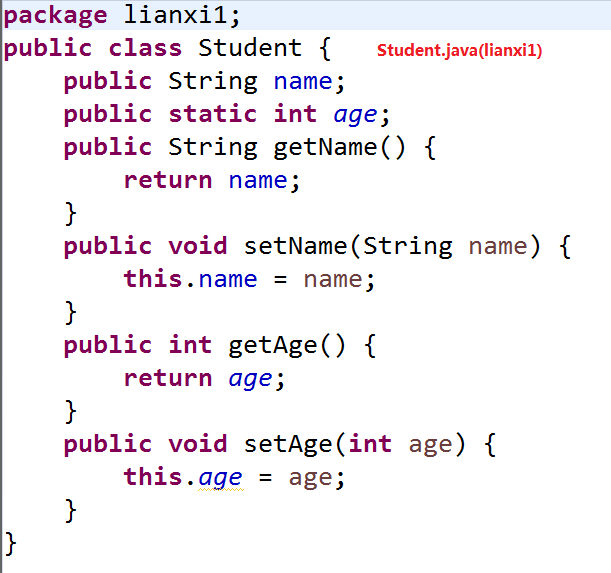
System.***out***.println("Code的构造方法");

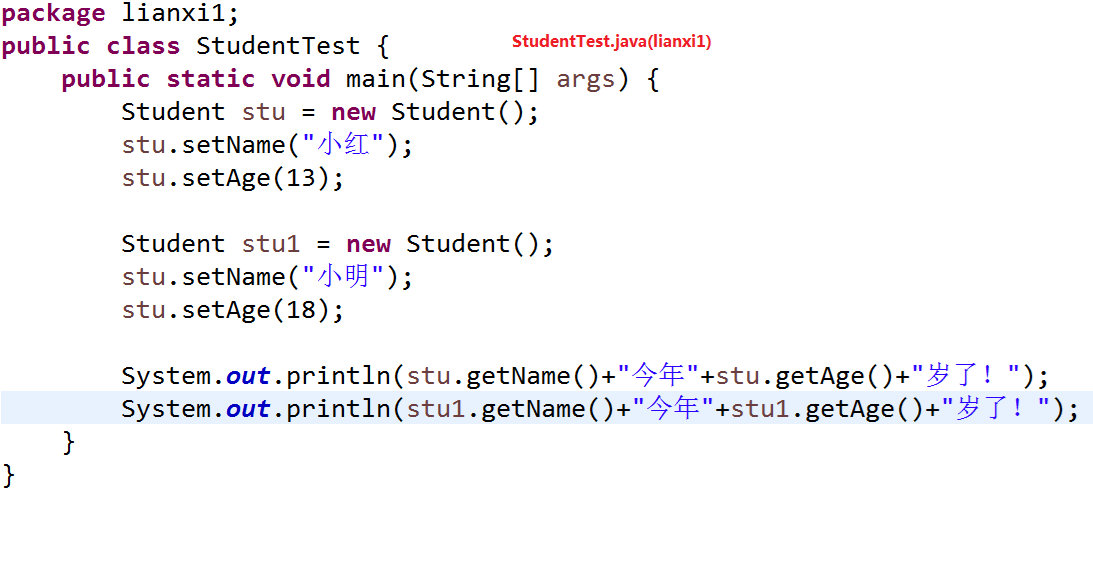
}

}

##### 2-10. 单选题

下面代码的运行结果是：（D）（选择一项）





A 小红今年13岁了！

B 小明今年18岁了！

C

小明今年18岁了！

null今年18岁了！

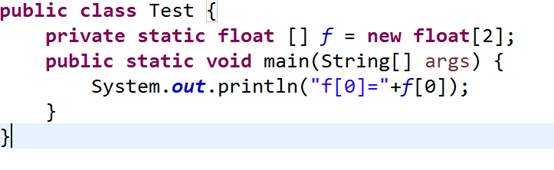
D

小红今年18岁了！

小明今年18岁了！

##### 3-4. 单选题

执行下面代码后，哪个结论是正确的（B）（选择一项）



A f[0]

B f[0] = 0.0

C 编译失败

D 在运行时抛出异常

##### 3-6. 多选题

执行下面代码后，哪几个结论是正确的（AD）（选择两项）

String[ ] s = new String[10];

A s[9]为null

B s[10]的内容为空字符串

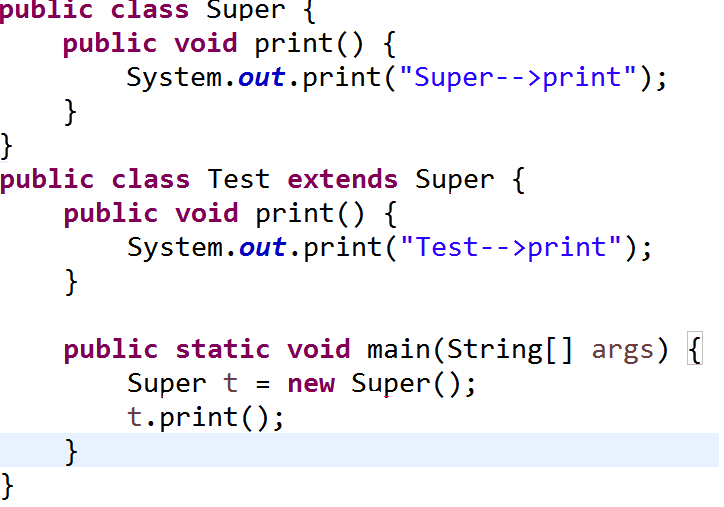
C 没有s[0]

D s.length=10

# JAVA继承

##### 2-4. 单选题

在JAVA中，以下程序的输出结果是（A）



A Super-->print

B Test-->print

C Super-->print Test-->print

D 编译错误

##### 2-5. 单选题

在JAVA中，以下关于方法重载和方法重写描述正确的是（D）（选择一项）

A 方法重载和方法重写实现的功能相同

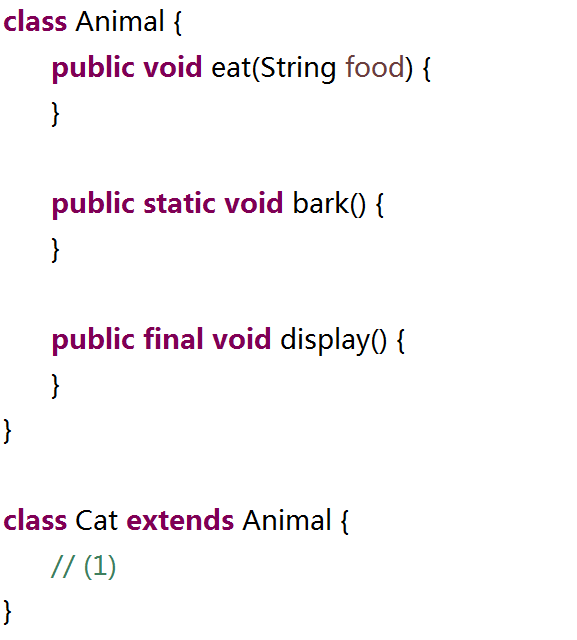
B 方法重载出现在父子关系中，方法重写是在同一类中

C 方法重载的返回类型必须一致，参数项必须不同

D 方法重写需要出现在满足继承关系的子类中

##### 2-7. 单选题

哪个选项中的方法插入到（1）处可以正确实现方法重写（C）（选择一项）



A public static void bark(){}

B public final void display(){}

C public void eat(String food){}

D public boolean eat(String food){}

##### 2-8 编程练习

编程练习：请使用面向对象的思想，设计自定义类完成如下功能要求：

接收用户输入的信息，选择需要完成的工作任务。其中，可供选择的有：测试工作和研发工作。关于类型设定描述如下：

测试工作

       属性：工作名称、编写的测试用例个数、发现的Bug数量

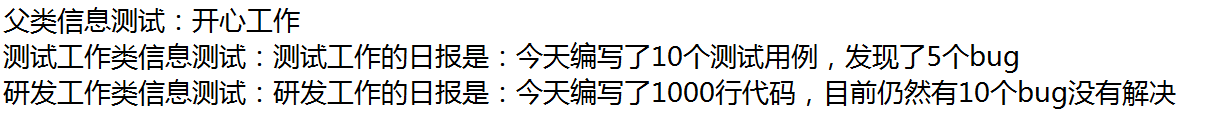
       方法：工作描述

研发工作

       属性：工作名称、有效编码行数、目前没有解决的Bug个数

       方法：工作描述

程序运行参考效果图如下：



任务

思路分析：

第一步：分析测试工作和研发工作的共性：

都是工作类型

都有工作名称的属性，工作描述的方法

第二步：根据共性，定义工作类

属性：工作名称

方法：

编写无参构造方法、带参构造方法完成对属性的赋值

编写工作描述的方法，描述内容为：开心工作。

第三步：定义测试工作类、研发工作类分别继承工作类，要求：

测试工作类：

 增加属性：编写的测试用例个数、发现的Bug数量

在构造方法中调用父类相关赋值方法，完成属性赋值

重写运行方法，描述内容为：\*\*的日报是：今天编写了\*\*个测试用例，发现了\*\*bug。其中\*\*的数据由属性提供

 研发工作类：

增加属性：有效编码行数、目前没有解决的Bug个数

在构造方法中调用父类相关赋值方法，完成属性赋值

重写运行方法，描述内容为：\*\*的日报是：今天编写了\*\*行代码，目前仍然有\*\*个bug没有解决。其中\*\*的数据由属性提供

public class Test {

public static void main(String[] args) {

Work w = **new** Work();

TestWork t = **new** TestWork(10,5);

DevelopmentWork d = **new** DevelopmentWork(1000,10);

System.***out***.println("父类信息测试："+w.work());

System.***out***.println("测试工作类信息测试："+t.work());

System.***out***.println("研发工作类信息测试："+d.work());

}

}

public class Work {

// 属性：工作ming

**private** String name;

// 无参构造方法

**public** Work() {

}

// 带参构造方法，完成工作类型的赋值

**public** Work(String name) {

**this**.name = name;

}

// 公有的get\*\*\*/set\*\*\*方法完成属性封装

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

// 方法：工作描述，描述内容为：开心工作

**public** String work() {

**return** "开心工作";

} }

public class TestWork extends Work {

// 属性：编写的测试用例个数、发现的Bug数量

**private** **int** tnum;

**private** **int** bnum;

// 编写构造方法，并调用父类相关赋值方法，完成属性赋值

**public** TestWork(**int** tnum, **int** bnum) {

**super**.setName("测试工作");

**this**.tnum = tnum;

**this**.bnum = bnum;

}

// 公有的get\*\*\*/set\*\*\*方法完成属性封装

**public** **int** getTnum() {

**return** tnum;

}

**public** **void** setTnum(**int** tnum) {

**this**.tnum = tnum;

}

**public** **int** getBnum() {

**return** bnum;

}

**public** **void** setBnum(**int** bnum) {

**this**.bnum = bnum;

}

// 重写运行方法，描述内容为：\*\*的日报是：今天编写了\*\*个测试用例，发现了\*\*bug。其中\*\*的数据由属性提供

**public** String work() {

**return** **super**.getName() + "的日报是：今天编写了" + **this**.getTnum() + "个测试用例，发现了" + **this**.getBnum() + "bug。";

}

public class DevelopmentWork extends Work {

// 属性：有效编码行数、目前没有解决的Bug个数

**private** **int** eff;

**private** **int** dbug;

// 编写构造方法，并调用父类相关赋值方法，完成属性赋值

**public** DevelopmentWork(**int** eff, **int** dbug) {

**super**.setName("研发工作");

**this**.eff = eff;

**this**.dbug = dbug;

}

// 公有的get\*\*\*/set\*\*\*方法完成属性封装

**public** **int** getEff() {

**return** eff;

}

**public** **void** setEff(**int** eff) {

**this**.eff = eff;

}

**public** **int** getDbug() {

**return** dbug;

}

**public** **void** setDbug(**int** dbug) {

**this**.dbug = dbug;

}

// 重写运行方法，描述内容为：\*\*的日报是：今天编写了\*\*行代码，目前仍然有\*\*个bug没有解决。其中\*\*的数据由属性提供

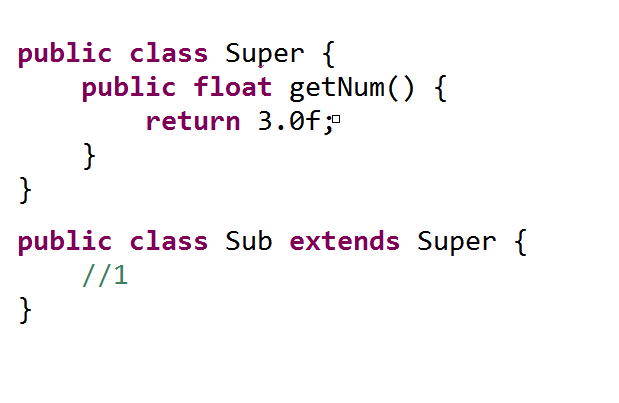
**public** String work() {

**return** **super**.getName()+"的日报是：今天编写了"+**this**.getEff()+"行代码，目前仍然有"+**this**.getDbug()+"个bug没有解决。";

} }

##### 2-11. 单选题

在下面程序的注释1处补充上下列（C）方法，会导致在编译过程中发生错误（选择一项）



A public float getNum() { return 4.0f; }

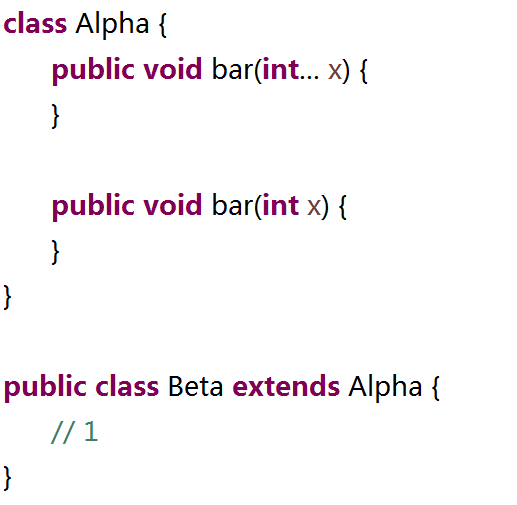
B private float getNum() {return 4.0f;}

C public void getNum（double d）{}

D public double getNum（float d）{ return 4.0f; }

##### 2-12. 多选题

代码位置（1）处填写（F）选项，可以使代码编译通过。（选择三项）



A private void bar(int x){}

B public void bar(int x){}

C public int bar(String x){return 1;}

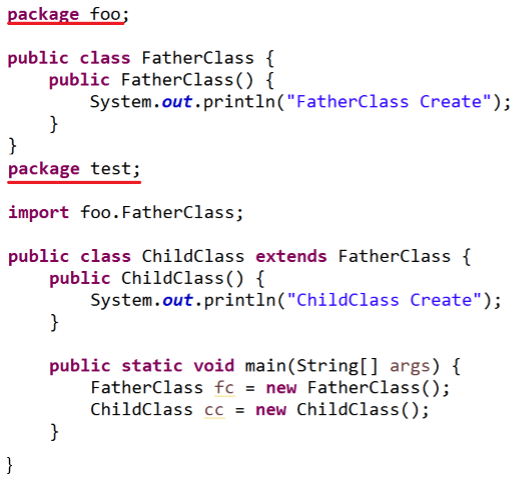
D public Alpha bar(int x){}

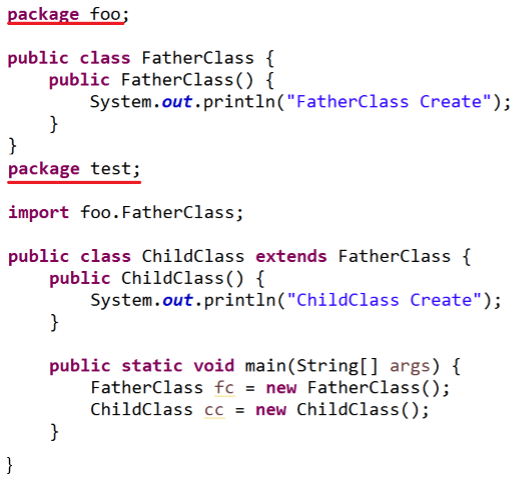
E public void bar(int x,int y){}

F public int bar(int x){return x;}

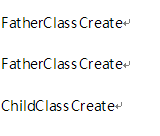
##### 2-15. 单选题

以下程序正确的输出结果是：（A）（选择一项）

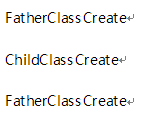




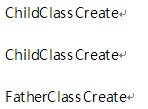
A



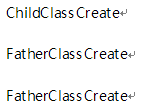
B



C

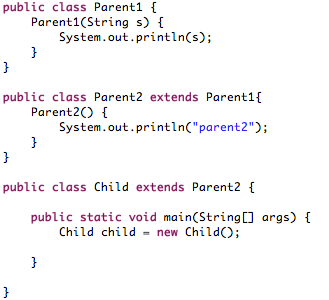


D



##### 2-16. 单选题

如下是一个Java源文件Child.java，编译并运行Child.java，以下结果正确的是（C ）（选择一项）



A 编译错误：没有找到构造器Child()

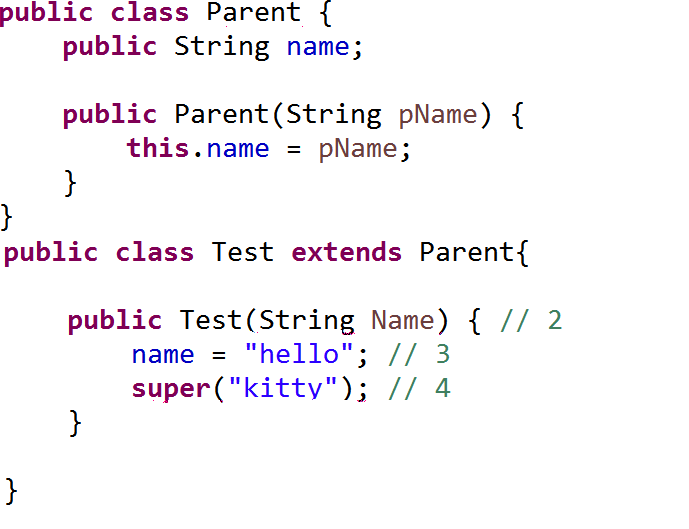
B 编译错误：没有找到构造器Parent1()

C 正确运行，没有输出值

D 正确运行，输出结果为：parent2

##### 2-18. 单选题

分析如下所示的Java代码，则选项中的说法正确的是（C）(选择一项)



A 第2行错误，Test类的构造函数中参数名称应与其父类构造函数中的参数名相同

B 第3行错误，应使用super关键字调用父类的name属性，改为super.name="hello"

C 第4行错误，调用父类构造方法的语句必须放在子类构造方法中的第一行

D 程序编译通过，无错误

##### 2-19. 单选题

关于super的说法正确的是（C）（选择一项）

A 是指当前子类的对象

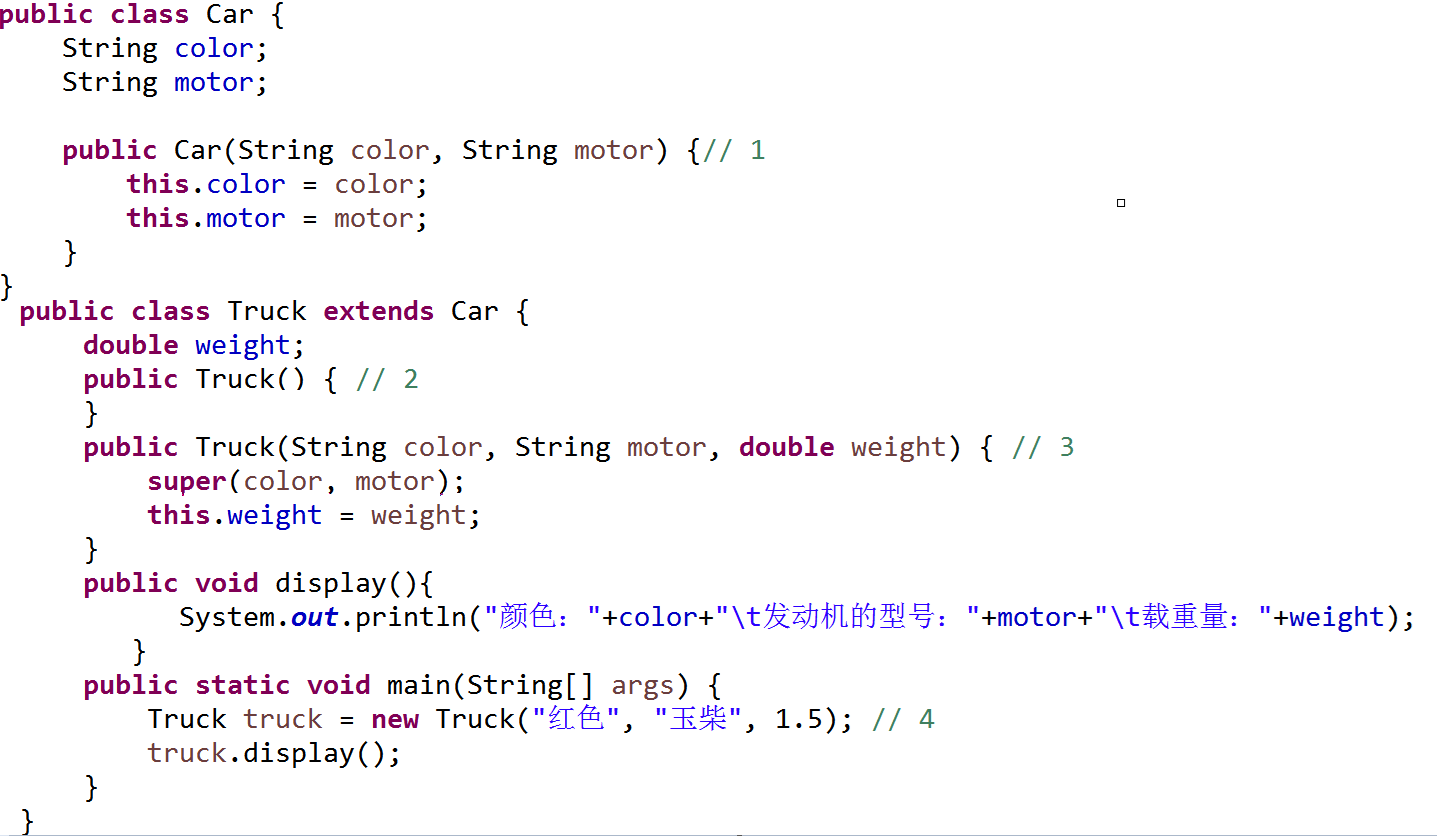
B 是指当前类的对象

C 是指当前父类的对象

D 可以用在main()方法中

##### 2-20. 单选题

阅读下面JAVA代码片段，正确的选项是（）。（选择一项）



A 第1行编译错误，但能输出正确结果

B 第2行编译错误，但能输出正确结果

C 第3行编译错误，不能输出正确结果

D 第4行编译错误，不能输出正确结果

##### 2-22. 多选题

下列关于super和this的说法正确的是（ABD）（选择三项）

A this关键字通常指当前对象

B super关键字则指父类对象

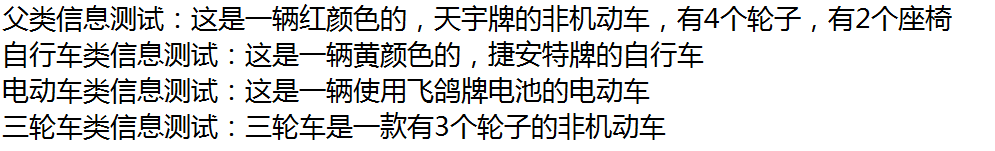
C 在一个类中this只可以调用当前类中公有属性和方法

D 在一个类中super可以调用父类中允许被访问的属性和方法

##### 2-23 编程练习

编程练习：某公司要开发“XX车行管理系统”，请使用面向对象的思想，设计自定义类描述自行车、电动车和三轮车。

程序参考运行效果图如下：



任务

任务分析:

第一步：分析自行车、电动车和三轮车的共性：

都是非机动车，具有非机动车的基本特征

都有运行的方法

第二步：根据共性，定义非机动车

属性：品牌、颜色、轮子(默认2个)、座椅（默认   1个）

方法：

编写无参构造方法、双参构造方法和四参构造方法，其中，在双参构造方法中，完成对品牌和颜色的赋值；在四参构造方法中，完成对所有属性的赋值

编写运行的方法，描述内容为：这是一辆\*\*颜色的，\*\*牌的非机动车，有\*\*个轮子，有\*\*个座椅的非机动车。其中\*\*的数据由属性提供

第三步：定义自行车、电动车和三轮车分别继承自行车类，要求：

自行车类：在构造方法中调用父类多参构造，完成属性赋值

重写运行方法，描述内容为：这是一辆\*\*颜色的，\*\*牌的自行车。其中\*\*的数据由属性提供

电动车： 增加“电池品牌”属性

 重写运行方法，描述内容为：这是一辆使用\*\*牌电池的电动车。其中\*\*的数据由属性提供

 三轮车：在无参构造中实现对轮子属性值进行修改

重写运行方法，描述内容为：三轮车是一款有\*\*个轮子的非机动车。其中\*\*的数据由属性提供

public class Test {

public static void main(String[] args) {

NonMotor f = **new** NonMotor("天宇牌","红颜色",4,2);

System.***out***.println("父类信息测试："+f.work());

Bicycle b = **new** Bicycle("捷安特", "黄颜色");

System.***out***.println("自行车类信息测试："+b.work());

ElectricVehicle e = **new** ElectricVehicle();

e.setDBrand("飞鸽牌");

System.***out***.println("电动车类信息测试："+e.work());

Tricycle t = **new** Tricycle();

System.***out***.println("三轮车类信息测试："+t.work()); }

}

public class NonMotor {

// 私有属性：品牌、颜色、轮子(默认2个)、座椅（默认 1个）

**private** String brand;

**private** String color;

**private** **int** lunzi = 2;

**private** **int** chair = 1;

// 无参构造方法

**public** NonMotor() {

}

// 双参构造方法，完成对品牌和颜色的赋值

**public** NonMotor(String brand, String color) {

**this**.brand = brand;

**this**.color = color;

}

// 四参构造方法，分别对所有属性赋值

**public** NonMotor(String brand, String color, **int** lunzi, **int** chair) {

**this**.brand = brand;

**this**.color = color;

**this**.lunzi = lunzi;

**this**.chair = chair;

}

// 公有的get\*\*\*/set\*\*\*方法完成属性封装

**public** String getBrand() {

**return** brand;

}

**public** **void** setBrand(String brand) {

**this**.brand = brand;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** **int** getLunzi() {

**return** lunzi;

}

**public** **void** setLunzi(**int** lunzi) {

**this**.lunzi = lunzi;

}

**public** **int** getChair() {

**return** chair;

}

**public** **void** setChair(**int** chair) {

**this**.chair = chair;

}

// 方法：运行，描述内容为：这是一辆\*\*颜色的，\*\*牌的非机动车，有\*\*个轮子，有\*\*个座椅的非机动车。其中\*\*的数据由属性提供

**public** String work() {

**return** "这是一辆"+**this**.getColor()+"颜色的，"+**this**.getBrand()+"牌的非机动车，有"+**this**.getLunzi()+"个轮子，有"+**this**.getChair()+"个座椅的非机动车。";

}}

public class Bicycle extends NonMotor {

// 在构造方法中调用父类多参构造，完成属性赋值

**public** Bicycle(String brand,String color) {

**super**(brand, color);

}

// 重写运行方法，描述内容为：这是一辆\*\*颜色的，\*\*牌的自行车。其中\*\*的数据由属性提供

**public** String work() {

**return** "这是一辆"+**this**.getColor()+"颜色的，"+**this**.getBrand()+"牌的自行车。";

}}

public class ElectricVehicle extends NonMotor {

// 私有属性：电池品牌

**private** String DBrand;

// 公有的get\*\*\*/set\*\*\*方法完成属性封装

**public** String getDBrand() {

**return** DBrand;

}

**public** **void** setDBrand(String dbrand) {

DBrand = dbrand;

}

// 重写运行方法，描述内容为：这是一辆使用\*\*牌电池的电动车。其中\*\*的数据由属性提供

**public** String work() {

**return** "这是一辆使用"+**this**.getDBrand()+"牌电池的电动车。";

}

}

public class Tricycle extends NonMotor {

// 在无参构造中实现对轮子属性值进行修改

**public** Tricycle() {

**super**.setLunzi(3);

}

// 重写运行方法，描述内容为：三轮车是一款有\*\*个轮子的非机动车。其中\*\*的数据由属性提供

**public** String work() {

**return** "三轮车是一款有"+**this**.getLunzi()+"个轮子的非机动车。";

}

}

##### 2-3下. 单选题

下列关于Object类的叙述错误的是（）（选择一项）

A Object类是所有类的父类

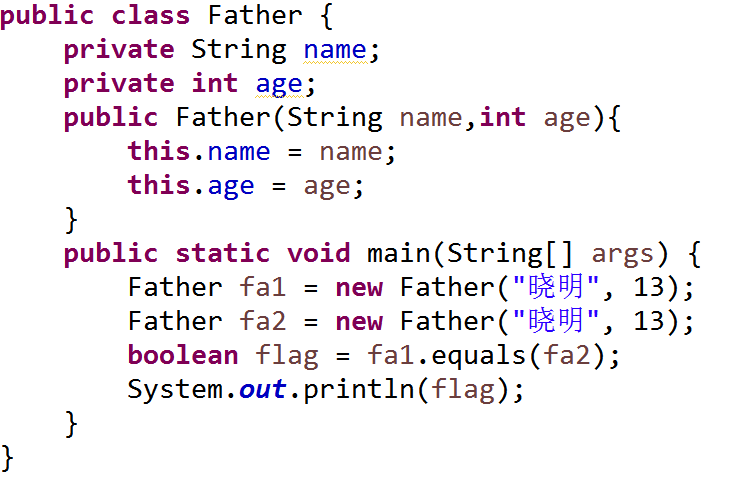
B 所有类都可以继承Object中允许被继承的方法

C 一个类没有使用extends关键字明确标识继承关系，则默认继承Object类

D 要想继承Object类，必须使用extends关键字标识继承关系，否则不会实现继承

##### 2-5下. 单选题

该段代码的运行结果为：（选择一项）



A true

B 相等

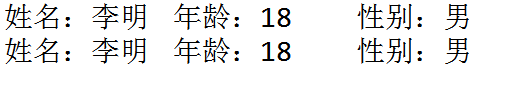
C 不相等

D false

##### 2-6下 编程练习

请使用面向对象的思想，设计自定义类Person继承Object类，重写toString方法实现对象信息输出。

运行效果如下图所示：



任务

思路分析

1、创建一个 Person 类继承自 Object，其中类的结构要求为：

      属性：name（姓名）、age(年龄)、sex（性别）

      方法：

创建带参（name、age、sex为参数）构造方法

重写 toString 方法，输出信息格式为：姓名：\*\* 年龄：\*\* 性别：\*\*（其中，\*\*为对象对应属性值）

2、创建测试类，在测试方法中，实例化 Person对 象，并传入三个属性值。然后，分别通过直接打印Person对象以及利用重写的 toString 方法，打印输出2行对象信息。

//测试类

public class Test {

public static void main(String[] args) {

//实例化对象，传入属性值（李明, 男,18）

Person person = new Person("李明",18,"男");

//打印输出对象信息

System.out.println(person);

System.out.println(person);

}

}

public class Person{

//私有属性：name（姓名）、age(年龄)、sex（性别）

private String name;

private int age;

private String sex;

//带参构造方法（name、age、sex为参数）

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

this.name=name;

this.age=age;

this.sex=sex;

}

//通过封装实现对属性的get/set方法设定

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;

}

//重写toString方法，表示形式为：姓名：+\*\*+ 年龄：+\*\*+ 性别：+\*\*

@Override

public String toString() {

return "姓名："+this.name+",年龄："+this.age+",性别："+this.sex;

}

}

##### 2-7下. 单选题

在Java中，关于继承的说法错误的是（）（选择一项）

A 使用extends关键字实现一个类继承另一个类

B 所有的Java类都直接或间接地继承了java.lang.Object类

C 在子类的构造方法中，必须显式调用父类的构造方法

D 在子类的构造方法中，可以通过super关键字调用父类的构造方法

##### 3-3下. 多选题

下列关于final的说法正确的是（）。（选择三项）

A final修饰的变量值不允许被修改

B final修饰的方法不可被重写

C final可以修饰所有方法

D final不可以修饰构造方法

##### 3-5下. 多选题

MAX\_LENGTH是int型public成员变量，数值保持为常量100，以下定义正确的是（）（选择两项）

A public int MAX\_LENGTH=100;

B final int MAX\_LENGTH=100;

C final public int MAX\_LENGTH=100;

D public final int MAX\_LENGTH=100;

##### 4-2下. 多选题

注解按照运行机制分为（）（选择三项）

A JDK注解

B 源码注解

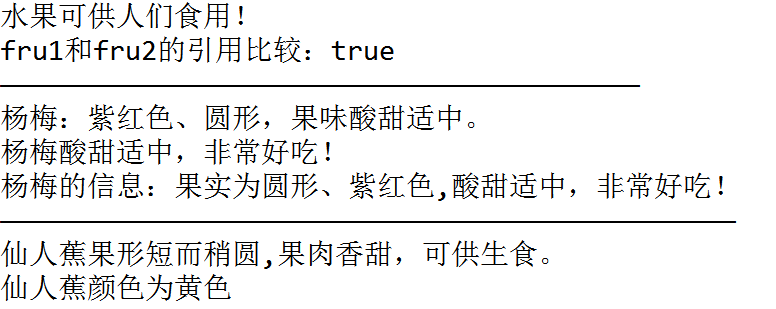
C 编译时注解

D 运行时注解

##### 4-3下 编程练习

请使用面向对象的思想，实现杨梅和仙人蕉的信息描述。

程序参考运行效果图如下：



任务

思路分析：

1、根据杨梅和香蕉的共性，抽取父类水果（Fruits）

私有属性：水果的形状（shape）和口感（taste）

方法：

1) 带参构造函数（参数为shape和taste）

2) 创建无参无返回值得方法eat（描述内容为：水果可供人们食用！）

3) 重写equals方法，比较两个对象是否相等（比较shape，taste）

2、子类Waxberry

私有属性：颜色（color）

方法：

1) 调用父类的构造方法，完成属性赋值

2) 创建不允许重写的face方法，描述为：杨梅：\*\*、\*\*，果味酸甜适中。

3) 重写父类eat方法，描述为：杨梅酸甜适中，非常好吃！

4) 重写toString方法，输出的表现形式不同（输出shape,color,taste）

5) 要求Waxberry类不允许有子类

3、子类：Banana

私有属性：品种（variety）

方法：

1) 带参构造方法为所有属性赋值

2) 创建无参无返回值的advantage方法，描述为：\*\*果形\*\*,果肉香甜，可供生食。

3) 重载要求(2)中的advantage方法（带参数color），描述为：\*\*颜色为\*\*

4、测试，运行效果参照效果图：

1) 实例化2个父类对象，并传入两组相同的参数值

2) 调用父类eat方法

3) 测试重写equals方法，判断两个对象是否相等

4) 实例化子类Wacberry对象，并传入相关参数值

5) 调用子类face方法和重写父类eat方法后的eat方法

6) 测试重写toString方法，输出子类对象的信息

7) 实例化Banana类对象，并传入相关参数值

8) 调用子类的advantage和它的重载方法

public class Test {

public static void main(String[] args) {

// 实例化2个父类对象，传入两组相同的参数值

Fruits fru1 = new Fruits("圆形", "酸甜");

Fruits fru2 = new Fruits("圆形", "酸甜");

// 调用父类eat方法

fru1.eat();

// 测试重写equals方法，判断两个对象是否相等

System.out.println(fru1.equals(fru2));

System.out.println("—————————————————————————");

// 实例化子类对象，并传入相关参数值

Waxberry wb = new Waxberr();

// 调用子类face方法和eat方法

wb.face();

web.eat();

// 测试重写toString方法，输出子类对象的信息

System.out.println(wb);

System.out.println("————————————————————————");

// 实例化Banana类对象，并传入相关参数值

Banana bn1 = new Fruits("长条", "特甜");

Banana bn2 = new Fruits("长条", "特甜");

// 调用子类的advantage和它的重载方法

bn2.advantage("紫红色"){

}

}

}

public class Fruits {

// 私有属性：水果的形状（shape）和口感（taste）

private String shape;

private String taste;

public Fruits() {

}

// 带参构造函数（参数为shape和taste）

public Fruits(String shape, String taste) {

this.shape=shape;

this.taste=taste;

}

//通过封装实现对私有属性的get/set访问

public String getShape() {

return shape;

}

public void setShape(String shape) {

this.shape = shape;

}

public String getTaste() {

return taste;

}

public void setTaste(String taste) {

this.taste = taste;

}

// 创建无参无返回值得方法eat（描述内容为：水果可供人们食用！）

public void eat(){

System.out.println("水果可供人们食用！");

}

// 重写equals方法，比较两个对象是否相等（比较shape，taste）

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (getClass() != obj.getClass())

return false;

Fruits other = (Fruits) obj;

if (shape == null) {

if (other.shape != null)

return false;

} else if (!shape.equals(other.shape))

return false;

if (taste == null) {

if (other.taste != null)

return false;

} else if (!taste.equals(other.taste))

return false;

return true;

}

}

//要求Waxberry类不允许有子类

public final class Waxberry {

// 私有属性：颜色（color）

private String color;

//创建构造方法，完成调用父类的构造方法，完成属性赋值

public Waxberry(){

super(String shape, String taste);

}

//创建不允许重写的face方法，描述为：杨梅：\*\*、\*\*，果味酸甜适中

public final void face(){

System.out.println("杨梅："+this.getShape()

+"、"+this.getTaste()+"，果味酸甜适中");

}

//重写父类eat方法，描述为：杨梅酸甜适中，非常好吃！

@override

public void eat(){

System.out.println("杨梅酸甜适中，非常好吃！");

}

//重写toString方法，输出的表现形式不同（输出shape,color,taste）

@Override

public String toString() {

return "shape="+(this.color)+",color="+this.taste+"taste="+this.taste;

}

}

public class Banana {

// 私有属性：品种（variety）

private String variety;

//创建带参构造方法为所有属性赋值

public Banana(String shape, String taste) {

this.shape=shape;

this.taste=taste;

}

//创建无参无返回值的advantage方法，描述为：\*\*果形\*\*,果肉香甜，可供生食。

public void advantage() {

System.out.println(this.shape+"果形"+this.taste+",果肉香甜，可供生食。");

}

//创建重载advantage方法（带参数color），描述为：\*\*颜色为\*\*

public void advantage(String color) {

System.out.println(this.variety+"颜色为"+color+"。");

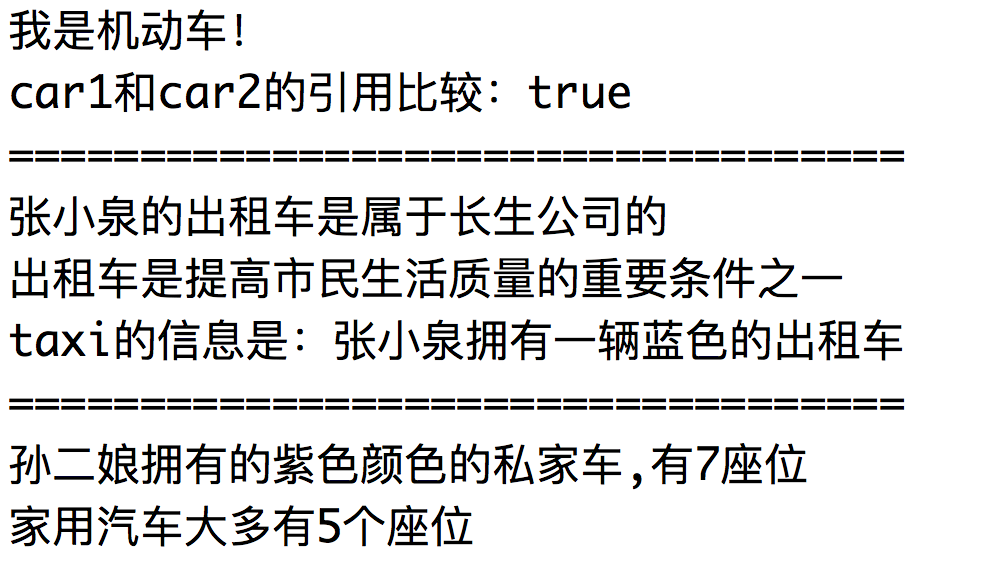
}

}

##### 5-2下 编程练习

请使用面向对象的思想，实现出租车和家用轿车的信息描述。

程序运行参考效果图如下：



任务

要求：

1、 根据出租车和家用轿车的共性，抽取父类Car

属性：车的颜色color、车主姓名userName

方法：

1) 带参构造函数（参数为color和useName）

2) 创建无参无返回值得方法use（描述内容为：我是机动车！）

3) 重写equals方法，比较两个对象是否相等（比较color，userName）

2、子类taxi

属性：所属公司( company)

方法：

1) 调用父类的构造方法，完成属性赋值

2) 创建不允许重写的ride方法，描述为：\*\*出租车是所属于在\*\*\*公司的。

3) 重写父类use方法，描述为：出租车是提高市民生活质量的重要条件之一

4) 重写toString方法，输出的表现形式不同（输出color,userName）

3、子类：私家车HomeCar

属性：载客数（num）

方法：

1) 带参构造方法为所有属性赋值

2) 创建无参无返回值的display方法，表现为：\*\*拥有的\*\*颜色的私家车有\*\*座位

3) 重载要求（2）中的display方法（带参数num），描述为：家用汽车大多有\*\*个座位

4) 要求私家车类不允许有子类

4、测试类，运行效果参照效果图：

1) 实例化2个父类对象，并传入两组相同的参数值

2) 调用父类use方法

3) 测试重写equals方法，判断两个对象是否相等

4) 实例化子类对象，并传入相关参数值

5) 调用子类ride方法和use方法

6) 测试重写toString方法，输出子类对象的信息

7) 实例化HomeCar类对象，并传入相关参数值

8) 调用子类的display和他的重载方法

public class Test {

public static void main(String[] args){

//实例化父类对象，并传入两组相同的属性值

//调用父类use方法

//通过重写equals方法，比较两个对象是否相等

System.out.println("==================================");

//实例化子类Taxi对象，并传入属性值

//调用ride和use方法

//输出重写toString后的taxi信息

System.out.println("==================================");

//实例化子类HomeCar对象，并传入属性值

//调用display方法及它的重载方法

}

}

public class Car {

//属性：车的颜色color、车主姓名userName

public Car(){

}

//带参构造函数（参数为color和useName）

//通过封装实现对私有属性的get/set操作

//创建无参无返回值得方法use（描述内容为：我是机动车！）

//重写equals方法，比较两个对象是否相等（比较color，userName）

}

}

public class Taxi extends Car{

//私有属性：所属公司( company)

public Taxi(){

}

//调用父类的构造方法，完成属性赋值

//通过封装实现对私有属性的get/set操作

//创建不允许重写的ride方法，描述为：\*\*出租车是所属于\*\*\*公司的。

//重写父类use方法，描述为：出租车是提高市民生活质量的重要条件之一

//重写toString方法，输出的表现形式不同（输出color,userName）

}

public final class HomeCar extends Car{

//私有属性：载客数（num）

public HomeCar(){

}

//带参构造方法为所有属性赋值

//通过封装实现对私有属性的get/set操作

//创建无参无返回值的display方法，表现为：\*\*拥有的\*\*颜色的私家车,有\*\*座位

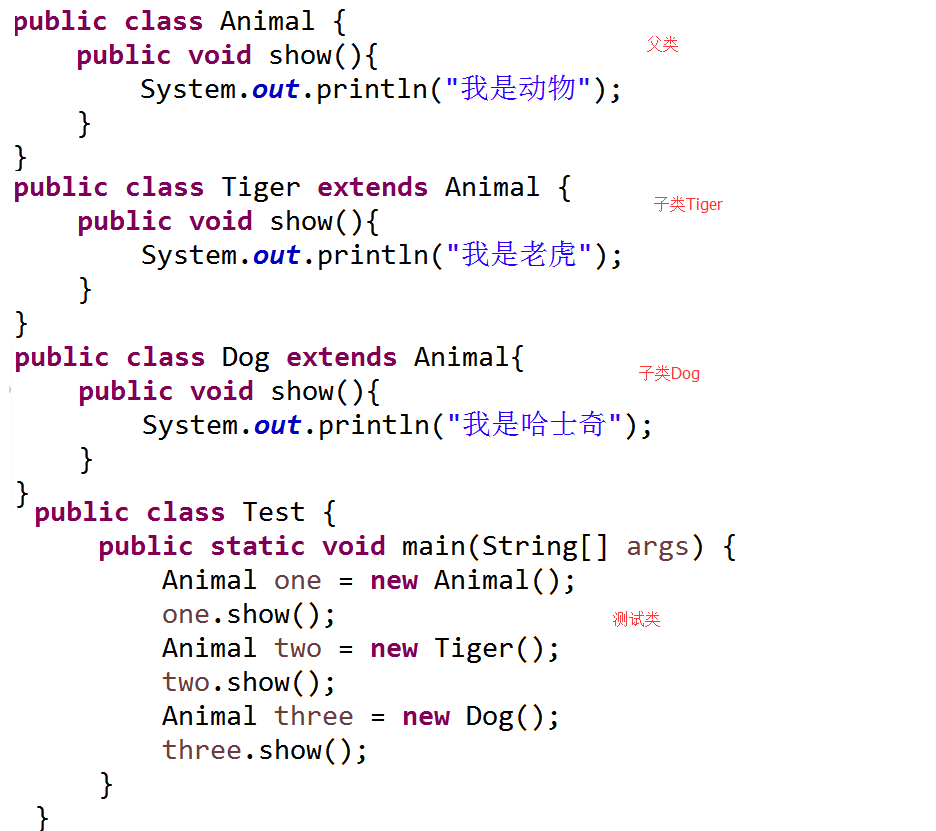
//重载display方法（带参数num），描述为：家用汽车大多有\*\*个座位

}

# JAVA多态

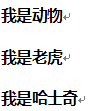
##### 3-3. 单选题

下列代码的运行结果为：（）（选择一项）。



A 我是动物

B 编译错误

C 

D 我是动物 我是老虎 我是哈士奇

答案：

##### 3-4. 单选题

创建一个父类Animal，一个子类Cat，Animal three = new Cat();不是（）（选择一项）

A 向上转型

B 自动转型

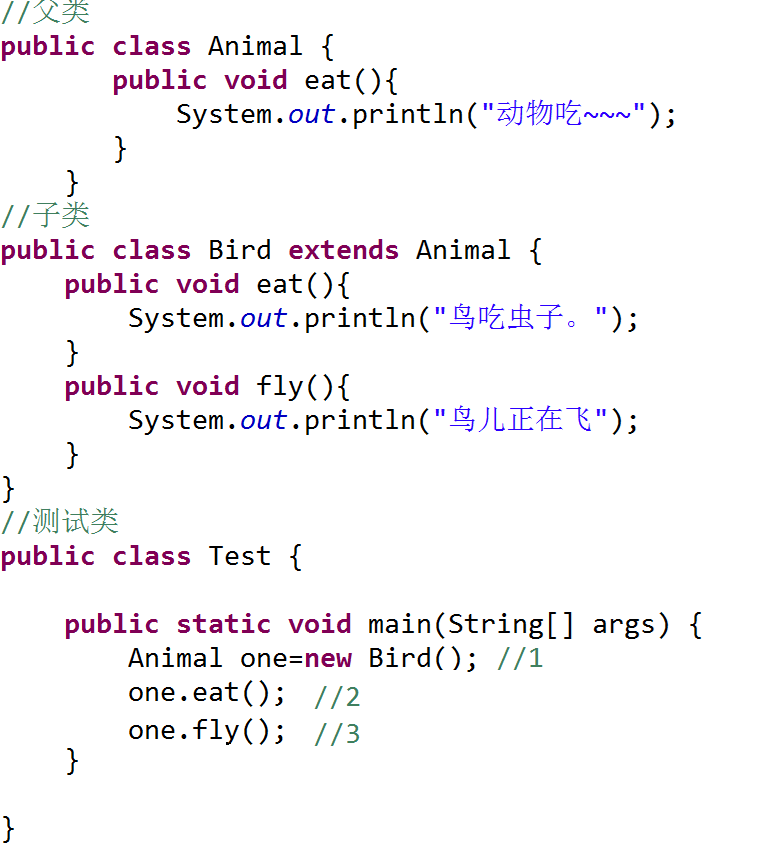
C 向下转型

D隐式转型

答案：

##### 3-5. 单选题

下列代码怎么修改可以使其成功运行：（）（选择一项）



A 删除掉标注3位置的one.fly( )

B 标注1的Animal one=new Bird()修改为Animal one=new Animal()

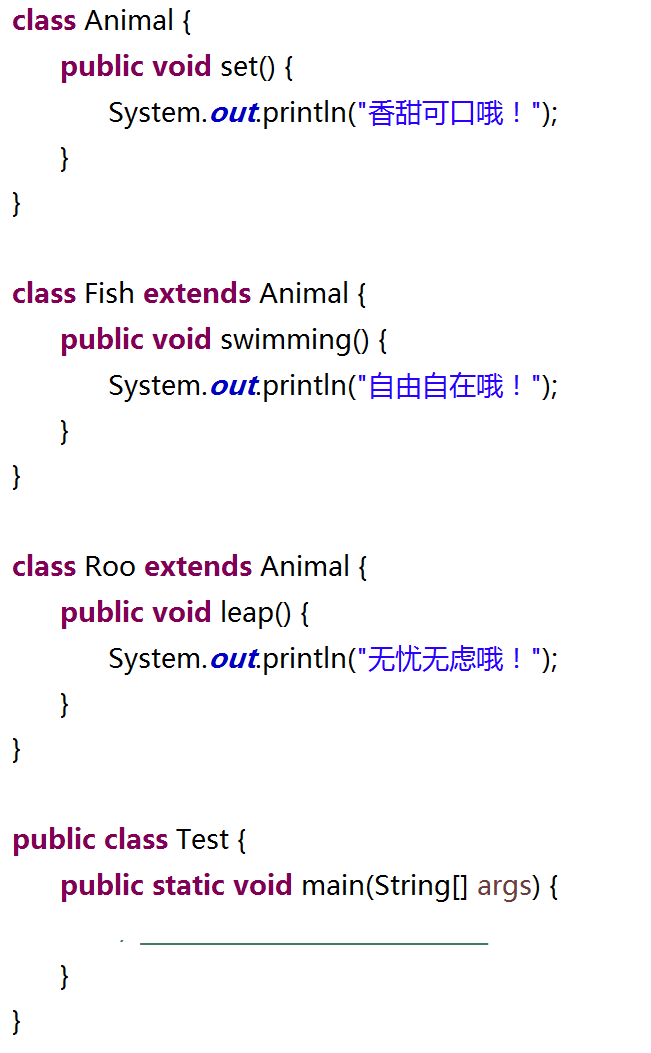
C 删除掉标注2位置的one.eat( )

D 标注1的Animal one=new Bird()修改为Bird one=new Animal()

答案：

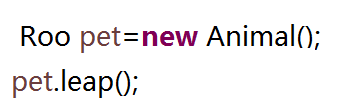
##### 3-7. 多选题

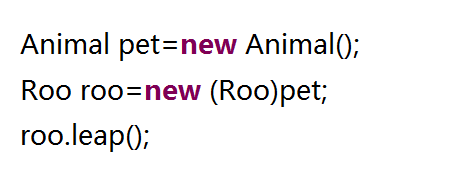
阅读以下代码，横线处填写（）无法在控制台输出‘无忧无虑哦！’。（选择两项）



A //img.mukewang.com/climg/59f69d9700012ce102670055.jpg

B //img.mukewang.com/climg/59f69d550001fba602740056.jpg

C 

D 

答案：

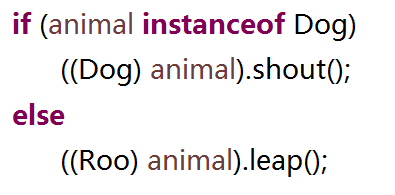
##### 3-9. 单选题

阅读下列Java代码，横线处应该填写（）能在控制台正确输出“汪汪汪！”。(选择一项)

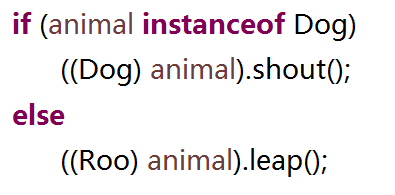


A 1处：(Dog)animal.shout();

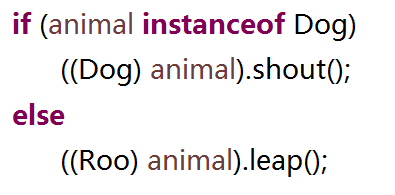
2处：Animal animal = new Dog();

B 1处：

2处：Animal animal = new Animal();

C 1处：

2处：Animal animal = new Dog();

D 1处：

2处：Animal animal = new Roo();

答案：

##### 3-10. 单选题

下列关于instanceof说法不正确的是（）。（选择一项）

A instanceof 的返回值为true和false

B instanceof可以用来判断对象是否可满足某个特定类型

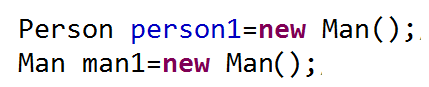
C 可以通过“A instanceof B"表示 A 类可以转型为B类

D instanceof可放在if语句的条件表达式中

答案：

##### 3-11. 单选题

已知父类Person,子类Man。判断类Person的对象person1是否满足类Man的实例特征，正确的语句为（）（选择一项）



A if (person1  instanceof  Man)

B if (man1  instanceof  Person)

C if (Person  instanceof  man1)

D if (Man  instanceof  person1)

答案：

##### 3-15. 单选题

在Java中，多态的实现不仅能减少编码的工作量，还能大大提高程序的可维护性及可扩展性，那么实现多态的步骤包括以下几个方面除了（ ）。（选择一项）

A 子类重写父类的方法

B 子类方法设置为final类型

C 定义方法时，把父类类型作为参数类型；调用方法时，把父类或子类的对象作为参数传入方法

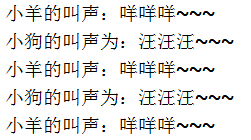
D 运行时，根据实际创建的对象类型动态决定使用哪个方法

答案：

##### 3-16 编程练习

应用继承和多态的思想，编写动物类，成员方法是动物叫声。写三个具体的类（猫、狗、羊），它们都是动物类的子类，并重写父类的成员方法。编写测试类，随机产生三种具体动物，调用叫声这个方法。

程序参考运行效果如图所示：



任务

任务分析：

1.定义一个父类Animal类

属性：kind(种类)

方法：创建带参（kind为参数）构造方法

            创建cry():void方法

2.编写三个具体的子类Cat类、Dog类、Sheep类,分别重写父类中的 cry() 方法，输出信息分别为:

Cat类：小猫的叫声：喵喵喵~~~

Dog类：小狗的叫声：汪汪汪~~~

Sheep类：小羊的叫声：咩咩咩~~~

3.编写测试类，首先生成长度为5的父类对象数组，然后通过循环依次向数组中存入数据，现设定存储规则为：

a)      每次随机产生一个0~2的正整数

b)     若数值为 0，则生成一个 Cat 类的对象，存入数组

c)      若数值为 1，则生成一个 Dog 类的对象，存入数组

d)     若数值为 2，则生成一个 Sheep 类的对象，存入数组

最后循环输出数组成员，并分别调用 cry() 方法。

public class Test {

public static void main(String[] args) {

//生成父类对象数组,数组长度为5

//产生随机数，随机产生三种具体子类的实例

//循环输出，循环体中每个对象分别调用cry()方法。

}

}

public class Animal {

//属性：kind(种类)

public Animal(){

}

//创建带参（king为参数）构造函数

//应用封装实现对私有属性的get/set操作

//创建成员方法cry():void

}

public class Cat extends Animal {

//重写父类cry()方法，输出信息为“小猫的叫声：喵喵喵~~~”

}

public class Dog extends Animal {

//重写父类cry()方法：输出信息为“小狗的叫声：汪汪汪~~~”

}

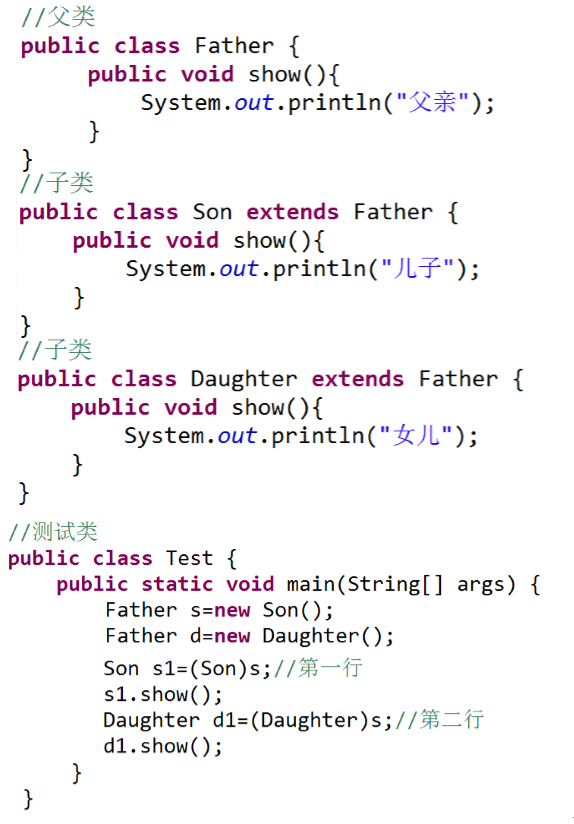
public class Sheep extends Animal {

//重写父类的方法cry(),要求输出信息“小羊的叫声：咩咩咩~~~”

}

##### 3-17. 单选题

下面代码运行测试后，出现的结果是（）(选择一项)



A 编译错误，错误位置在第一行

B 编译错误，错误位置在第二行

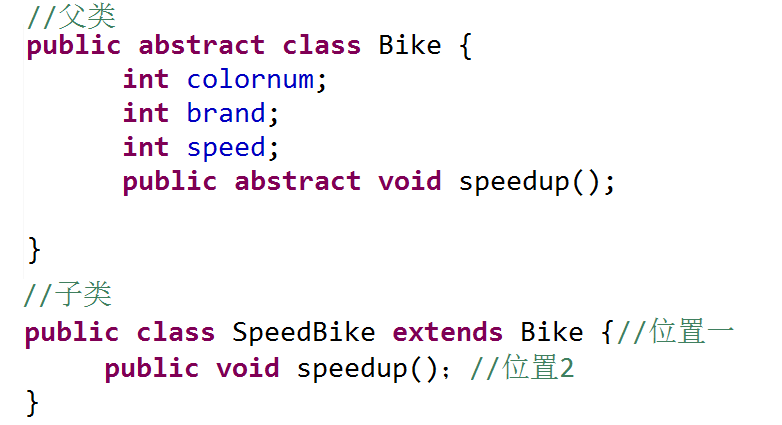
C 第一行和第二行都运行成功，输出结果为:

D 编译成功，但运行报错，错误位置在第二行

答案：

##### 4-3. 多选题

下面代码怎么修改可以编译时不报错（）（选择两项）



A 在位置一处将SpeedBike类设为抽象类，同时将位置2处的speedup也设为抽象方法

B 将位置一中的public改为final

C 将位置二中speedup()方法改为抽象方法

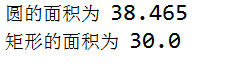
D 将位置二中speedup()方法中加入方法的实现

答案：

##### 4-4 编程练习

定义一个抽象类图形Shape类，由该派生出两个子类圆Circle类和矩形Rectangle类。Shape里声明了抽象方法area()，该方法分别在两个子类里得到实现。

程序参考运行效果图如下：



任务

任务分析：

1、定义一个抽象类Shape，在它里面创建抽象方法area（）方法

2、它的子类Circle类和Rectangle类

子类Circle类：

       属性： r（圆的半径）

       方法：

创建带参构造函数（参数为r）

重写area()方法，要求在里面返回圆的面积

Rectangle类：

       属性：length（长）和wide(宽)

      方法：

创建带参（length和wide为参数）构造方法

重写area()方法，要求在里面返回矩形的面积

3、编写测试类，分别创建Circle类的对象和Rectangle类的对象，设置圆的半径以及矩形的长宽，输出圆的面积和矩形的面积。（具体输出结果以具体设定参数为准）

//测试类

public class Test {

public static void main(String[] args) {

//创建类的实例，并分别对圆的半径和矩形的长宽进行赋值

//调用area（）方法，输出结果

}

}

public class Rectangle extends Shape {

//属性：矩形的长lenghth、宽wide

//创建带参构造方法以及无参构造方法

//创建针对长、宽的赋值和取值方法

//重写父类的area（）方法

}

//子类圆Circle类

public class Circle extends Shape {

//属性：圆的半径r

//创建带参构造方法以及无参构造方法

//创建针对半径的赋值和取值方法

//重写父类中area()方法

}

//抽象类Shape，里面含抽象方法area( )

class Shape{

}

##### 4-5. 单选题

下列选项中，关于Java的抽象类和抽象方法说法不正确的是（）。（选择一项）

A 抽象类和抽象方法都通过abstract关键字来修饰

B 抽象类中必须包含抽象方法

C 抽象方法只有方法声明，没有方法实现

D 子类如果不重写父类所有的抽象方法，则必须设置为抽象类

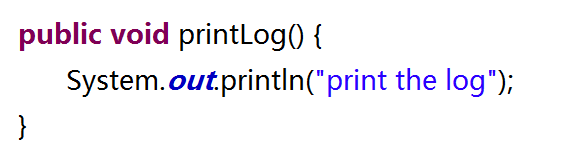
答案：

##### 5-4. 单选题

阅读下面的Java代码，能够填写在横线处的语句是（）。（选择一项）



A private int MAX\_LOG\_SIZE = 1000;

B 

C private Boolean saveToFile(String fileNmae);

D int getSize();

答案：

##### 5-5. 单选题

下列关于Java中接口的说法不正确的是（） (选择一项)

A 接口中方法的访问修饰符默认为public

B 接口中的方法如果写成void test();的形式，默认是抽象方法

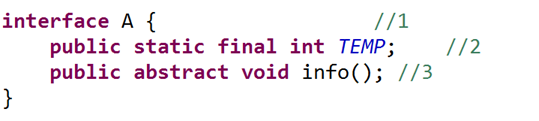
C 实现接口的类中在重写接口中方法时访问修饰符可以为protected

D 当类实现接口时，需要实现接口中所有的抽象方法，否则需要将该类设置为抽象类

答案：

##### 5-6. 单选题

运行下列代码时，哪个位置会发生编译报错（）(选择一项)



A 位置1

B 位置2

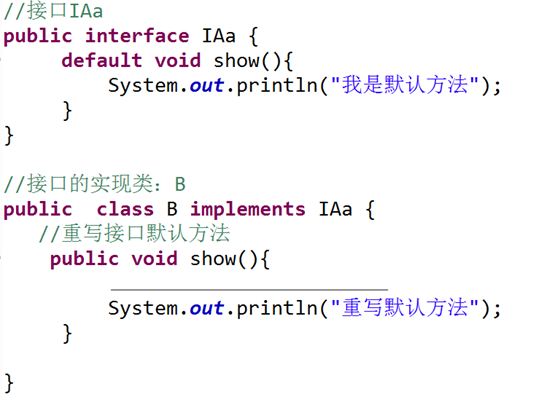
C 位置3

D 不存在错误

答案：

##### 5-8. 单选题

在实现类中的划线处加入下列哪条代码可以调用 IAa 接口中默认的方法（）（选择一项）



A //climg.mukewang.com/583e376a0001692202110038.jpg

B //climg.mukewang.com/583e37740001a86c02130034.jpg

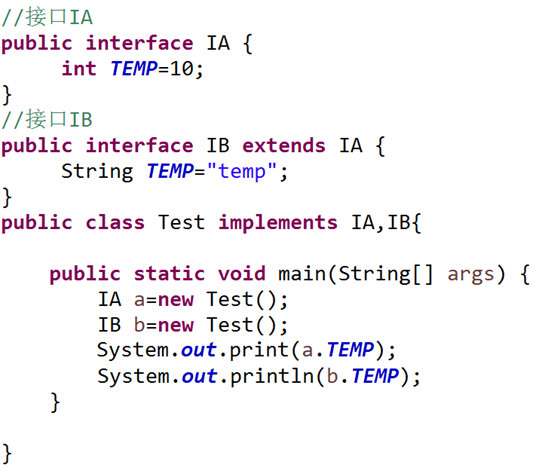
C //climg.mukewang.com/583e37770001b3c803160033.jpg

D //climg.mukewang.com/583e377f0001d05f02750037.jpg

答案：

##### 5-11. 单选题

下列代码的运行结果是（）。(选择一项)



A 10temp

B temp10

C 1010

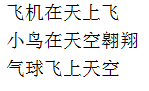
D temptemp

答案：

##### 5-13 编程练习

使用接口的知识， 定义接口IFly,创建三个类Plane类、Bird类、Balloon类，分别重写接口中的fly( )方法，然后再测试类中进行调用。

程序运行参考效果如图所示：



任务

任务分析：

1、创建接口IFly( )

方法：创建抽象方法 fly() 方法

2、创建子类：Plane

方法：实现接口中的方法fly( )，输出信息“飞机在天上飞”

创建子类：Bird

方法：实现接口中的方法fly( )，输出信息“小鸟在天空翱翔"”

创建子类：Balloon(气球)

方法：实现接口中的方法fly( )，输出信息“气球飞上天空”

3、创建测试类，分别创建Plane、Bird、Balloon类的对象，调用 fly( ) 方法，输出效果参考效果图

//测试类

public class Test{

public static void main(String[] args){

//对象实例化

//分别调用fly()方法

}

}

//创建类Balloon并实现接口IFly

public class Balloon{

//重写方法fly()

}

//创建类Bird并实现接口IFly

public class Bird{

//重写fly()方法

}

//创建类Plane并实现接口IFly

public class Plane{

//重写fly方法

}

//定义接口IFly

public IFly{

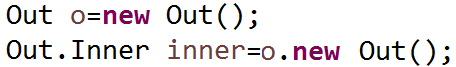
//创建方法fly()

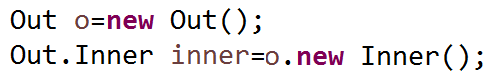
}

# JAVA内部类

##### 6-3. 多选题

已知外部类Out中含有成员内部类Inner，在主方法中怎么获取内部类Inner的实例化对象inner（）（选择两项）

A 

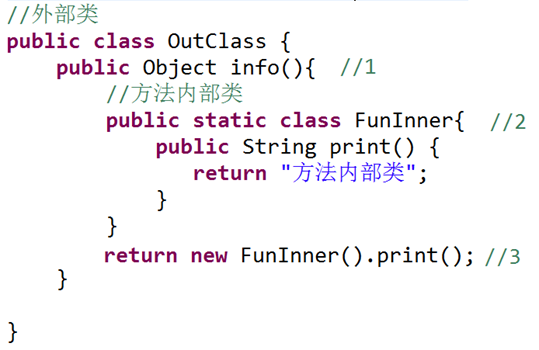
B 

C //climg.mukewang.com/5836504d0001214f04600035.jpg

D //climg.mukewang.com/5836505f0001474804870028.jpg

##### 6-6. 单选题

关于下列代码，说法正确的是（）（选择一项）



A 代码编译成功

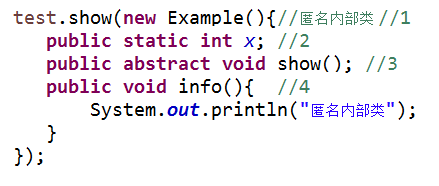
B 代码编译错误，错误发生在位置1

C 代码编译错误，错误发生在位置2

D 代码编译错误，错误发生在位置3

##### 6-9. 多选题

下面截取了一段匿名内部类的代码，关于这段代码，下面说法正确的是（）（选择两项）



A 位置1有错误

B 位置2有错误

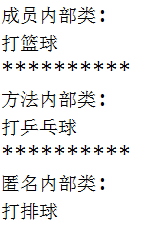
C 位置3有错误

D 位置4有错误

##### 6-10 编程练习

分别通过成员内部类、方法内部类、匿名内部类完成接口Ball，在测试类BallTest中的调用。

程序参考运行效果图如下：



任务

任务分析：

1.    创建接口Ball , 包含抽象方法play( )

2.    创建外部类BallTest, 定义方法 void playBall(Ball ball)，调用接口中的 play( ) 方法

3.    分别通过成员内部类、方法内部类以及匿名内部类完成接口实现

> 在BallTest中，创建成员内部类 Inner\_m 实现接口Ball，在类中重写 play( ) 方法，输出两条信息，分别为"成员内部类:"和"打篮球"

> 在BallTest中， 创建成员方法 info() ，在方法中创建方法内部类 Inner\_f 实现Ball接口，重写 play( ) 方法，输出三条信息，分别为"\*\*\*\*\*\*\*\*\*\*"，"方法内部类:" 和 "打乒乓球"

> 在测试类Test中通过匿名内部类实现接口Ball，并重写 play( )方法，输出两条信息，分别为 "\*\*\*\*\*\*\*\*\*\*"，"匿名内部类:" 和 "打排球"

4.    创建测试类Test，通过创建BallTest类及相关内部类实例，调用相关方法，依次完成相关内部类测试。运行效果参见效果图

//测试类

public class Test {

public static void main(String[] args) {

//完成成员内部类内部测试

//完成方法内部类测试

//完成匿名内部类测试

}

//创建类BallTest

public class BallTest{

// 创建成员内部类Inner\_m

// 创建方法内部类

// 定义一个方法void playBall(Ball ball)，调用play()方法

}

//接口Ball

public interface Ball {

//创建抽象方法play()：void

}

# JAVA单例模式

##### 2-2. 多选题

单例模式的实现必须满足（）个条件。（选择三项）

A 类中的构造方法的访问权限必须设置为私有的

B 类中的构造方法必须用protected修饰

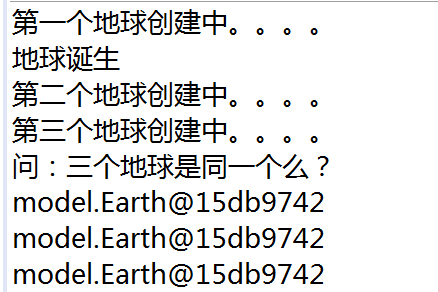
C 必须在类中创建该类的静态私有对象

D 在类中提供一个公有的静态方法用于创建、获取静态私有对象

##### 3-2 编程练习

某公司研发星球维护系统，请使用饿汉式单例模式的实现思想，设计编写地球类。

程序运行参考效果图如下：



public class Test {

public static void main(String[] args) {

System.out.println("第一个地球创建中。。。。");

System.out.println("第二个地球创建中。。。。");

System.out.println("第三个地球创建中。。。。");

System.out.println("问：三个地球是同一个么？");

}

}

public class Earth {

//定义私有构造方法，并在构造方法中打印输出“地球诞生”

//定义私有静态类对象并完成实例化

//定义公有静态方法返回类内的私有静态对象

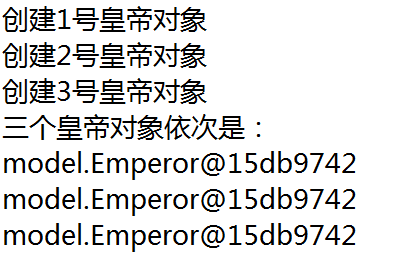
}

}

##### 3-4 编程练习

请使用懒汉式单例模式的实现思想，设计编写皇帝类。

程序运行参考效果图如下：



任务

实现步骤：

1、定义私有构造方法

2、定义私有静态类对象

3、定义公有静态方法返回类内的私有静态对象，结合判断实现对象实例化

4、编写测试类，结合注释完成测试单例的皇帝类信息，程序效果参考运行效果图（具体对象引用信息不限）

public class Test {

public static void main(String[] args) {

System.out.println("创建1号皇帝对象");

System.out.println("创建2号皇帝对象");

System.out.println("创建3号皇帝对象");

System.out.println("三个皇帝对象依次是：");

}

}

public class Emperor {

//定义私有构造方法

//定义私有静态类对象

//定义公有静态方法返回类内的私有静态对象

}

##### 3-6. 多选题

下列关于懒汉式和饿汉式的说法错误的是（）（选择两项）

A 饿汉式在第一次使用时进行实例化

B 懒汉式在类加载时就创建实例

C 饿汉式的线程是安全的

D 懒汉式存在线程风险