java面向对象考题和答案

1.下面代码的输出结果是:

```
public class Main {
public static void main(String[] args) { int n = 100;
int m = 200;
System.out.println(f(n,m));
System.out.println(n);
}
public static int f(int m, int n) {
n = m+n;
return n;
}
}
```

A.300

300

B.100

100

C.300

100

D.100

300

2.程序执行的结果是:

```
public class Test {
public static void main(String[] args) {
int x = 6;
Test p = new Test();
p.doStuff(x);
System.out.println(" main:  x = " + x); }
void doStuff(int x) {
System.out.println(" doStuff:  x =" + x++); }
}
```

A.doStuff: x =7

main: x = 7

B.doStuff: x =7

main: x = 6

C.doStuff: x =6

main: x = 7

D.doStuff: x =6

main: x = 6

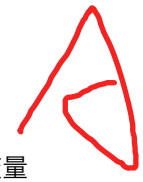
3.下列关于JVM的内存结构描述正确的是:

A.类的各种信息在方法区中保存

B.堆用于存放程序运行过程当中所有的局部变量

C.栈用于存储使用new关键字创建的对象

D.数组属于JVM的内存结构



JVM的内存结构大致分为五个部分,分别是程序计数器、虚拟机栈、本地方法栈、堆和方法区。除此之外,还有由堆中引用的JVM外的直接内存

4.下列代码的输出结果是:

```
public class Test {
public void print(char c) {
System.out.println("c");
}
public void print(int i) {
System.out.println("i");
}
public void print(double d) {
System.out.println("d");
}
public void print(String s) {
System.out.println("s");
}
public static void main(String[] args) { Test test=new Test();
test.print('5');
}
}
```

A.c

B.i

C.d

D.s

5.程序执行的结果是:

```
public class Test {
String name="Tom";
public Test(String name){
```

name=name;

}

public static void main(String [] args){ Test t = new Test("Jack");

System.out.println(http://www.doczj.com/doc/defffa8b094e767f5acfa1c7aa00b52acec79c01.html);

}

}

A.null

B.Tom

C.Jack

D." "

6.关于构造方法，下列说法错误的是:

B

A.构造方法不可以进行方法重写

B.构造方法用来实例化一个新的对象

应该是初始化不是实例化，实例化由new运算符完成，之后自动调用构造方法

C.构造方法具有和类名相同的名称

D.构造方法不返回任何数据类型

7.关于 Java 中继承的特点，下列说法正确的是:

A.使类的定义复杂化

B.Java 只支持单继承，不可多继承，但可以通过实现接口来达到多继承的目的

C.子类继承父类的所有成员变量和方法，包括父类的构造方法

D.不可以多层继承，即一个类不可以继承另一个类的子类

8.下列代码运行的结果是:

class Foo {

public int a;

public Foo() {

a = 3;

}

public void addFive() {

a += 5;

}

}

class Bar extends Foo {

public int a;

public Bar() {

a = 8;

}

```
public void addFive() {
this.a += 5;
}
}
public class TestFoo {
public static void main(String[] args) {
Foo foo = new Bar();
foo.addFive();
System.out.println("Value: " + foo.a);
}
}
```

new Bar()时，创建了一个Bar类对象，同时会默认调用父类的空参构造器，从而创建一个父类对象。
子类和父类定义相同的方法会产生覆盖，定义相同的属性则不会
这里foo.addFive()调用的是子类的方法(验证可知，Bar中的a=13)，这里foo.a是父类的a=3

A.Value: 3

B.Value: 8

C.Value: 13

D.Value: 18

9.下列代码编译和运行的结果是:

```
class Person {
String name = "No name";
public Person(String nm) {
name = nm;
}
}
class Employee extends Person {
String empID = "0000";
public Employee(String id) {
empID = id;
}
}
public class EmployeeTest {
public static void main(String[] args) {
Employee e = new Employee("4321");
System.out.println(e.empID);
}
}
```

子类在实例化时会默认调用父类的空参构造器，
这里父类声明了带参构造器，覆盖了空参构造器，
所以子类调不到，出错

A.输出：0000

B.输出：4321

C.代码public Employee(String id) {行，出现编译错误

D.抛出运行时异常

10.下列代码的运行结果是：

```
public class Animal {
public String noise() {
return "peep";
}
public static void main(String[] args) {
Animal animal = new Dog();
Cat cat = (Cat)animal;
System.out.println(cat.noise());
}
}
class Dog extends Animal {
public String noise() {
return "bark";
}
}
class Cat extends Animal {
public String noise() {
return "meow"; }
}
```

animal原本是dog，无法强转为cat

A.peep

B.bark

C.meow

D.抛出运行时异常

11.下列代码编译和运行的结果是：

```
public class A {
public void start() {
System.out.println("TestA");
}
}
public class B extends A {
public void start() {
```

System.out.println("TestB");

}

public static void main(String[] args) {

((A) new B( )).start();

}

}

A.输出：TestA

B.输出：TestB

C.输出：TestA TestB

D.编译错误

12.请看下列代码：

class One {

void foo() {

}

}

class Two extends One {

// insert method here

}

下列选项中的代码，放置在<插入代码>处无编译错误的是： A.int foo() { /* more code here */ }

B. protected void foo() { /* more code here */ }

C. public void foo() { /* more code here */ }

D. private void foo() { /* more code here */ }

BC

子类不能声明与父类方法同名且返回值类型不同的方法，因为在多态时会产生矛盾，不知道调用的是哪个

如果子类要覆盖了父类中定义的方法，那么不能降低其可见性

13.下列选项中，不属于Java 的访问修饰符的是：

A.private

B.protected

C.friendly

D.public

14.下列代码的输出结果是：

B

class Foo {

private int x;

public Foo(int x) {

this.x = x;

}

public void setX(int x) {

this.x = x;

```
}
public int getX() {
return x;
}
}
public class Gamma {
static Foo fooBar(Foo foo) {
foo = new Foo(100);
return foo;
}
public static void main(String[] args) {
Foo foo = new Foo(300);
System.out.print(foo.getX() + "-");
Foo fooFoo = fooBar(foo);
System.out.print(foo.getX() + "-");
System.out.print(fooFoo.getX() + "-");
foo = fooBar(fooFoo);
System.out.print(foo.getX() + "-");
System.out.print(fooFoo.getX());
}
}
```

传过去的引用的副本被覆盖了，而原来的引用没变

A.300-100-100-100-100

B.300-300-100-100-100

C.300-300-300-100-100

D.300-300-300-300-100

15.下列代码运行的结果是:

```
public class Base {
public static final String FOO = "foo";
16.class Sub extends Base {
public static final String FOO = "bar";
}
public static void main(String[] args) {
Base b = new Base();
Sub s = new Sub();
System.out.print(Base.FOO);
```

System.out.print(Sub.FOO);

System.out.print(b.FOO);

System.out.print(s.FOO);

System.out.print(((Base) s).FOO);

}

}

A.foofoofoofoofoo

B.foobarfoobarbar

C.foobarfoofoofoo

D.foobarfoobarfoo

子类与父类的静态变量名相同时，静态变量是不同的，父类强转为子类后，静态变量就是子类的静态变量

16.关于下列代码说法正确的是：

public class ItemTest {

private final int id;

public ItemTest(int id) {

this.id = id;

}

public void updateId(int newId) {

id = newId;

}

public static void main(String[] args) {

ItemTest fa = new ItemTest(42);

fa.updateId(69);

System.out.println(fa.id);

}

}

A.编译错误

B.运行时抛出异常

C.运行后，fa对象属性id的值没有改变，应然是42