Java面向对象综合测试

不定项选择题（共40道题，每题3分，共计120分）

1.下面关于Java语言中方法的说法错误的是:【BD】

A.方法调用时参数类型必须符合方法的定义

B.参数传递是值传递的方式

C.如果方法没有返回值必须声明返回为void

D.如果方法定义为返回void，则方法中不能出现return语句

2.指出下列程序运行的结果【B】

```
public class Example{
String str=new String("welcome");
char[]ch={ ‘a’,’b’,’c’};
public static void main(String args[]){
Example ex=new Example();
ex.change(ex.str,ex.ch);
System.out.print(ex.str+" and ");
System.out.print(Arrays.toString(ex.ch));
}
public void change(String str,char ch[] ){
str="test ok";
ch[0]=’g’;
}
}
```

A.welcome and abc

B.welcome and gbc

C.test ok and abc

D.test ok and gbc

3.关于Java对象的删除，下列说法正确的是: 【B】

A.必须由程序员完成对象的清除

B.Java把没有引用的对象作为垃圾收集起来并释放

C.只有当程序中调用System.gc()方法时才能进行垃圾收集

D.Java中的对象都很小，一般不进行删除操作

4.下列代码编译或运行的结果是:【C】

```
public class Foo {
 public String doit(int x, int y) {
    return "a";
 }
 public String doit(int[] vals) {
    return "b";
 }
 public static void main(String[] args) {
    Foo f=new Foo();
    System.out.println(f.doit(4, 5));
 }
}
```

A.代码public String doit(int[] vals) {行，编译出错

B.代码System.out.println(f.doit(4, 5));行，抛出运行时异常

C.输出：a

D.输出：b

5.请看下列代码:

```
class Inner {
 private int x;
 public void setX(int x) {
  this.x = x;
 }
 public int getX() {
  return x;
 }
}
```

```
class Outer {
 private Inner y;
 public void setY(Inner y) {
  this.y = y;
 }
 public Inner getY() {
  return y;
 }
}
public class Gamma {
  public static void main(String[] args) {
    Outer o = new Outer();
    Inner i = new Inner();
    int n = 10;
    i.setX(n);
    o.setY(i);
    <插入代码> i = new Inner(); i.setX( 100); o.setY( i);
    System.out.println(o.getY().getX());
  }
}
```

运行上述代码，要求输出“100”，那么<插入代码>处应填入的代码是：【BCD】

A.n = 100;

B.i.setX(100);

C.o.getY().setX(100);

D.i = new Inner(); i.setX(100); o.setY(i);

6.关于下列代码说法正确的是：【D】

```
public class Foo {
 public int add(int a, int b) {
  return a + b;
 }
```

```
 public static void main(String[] args) {
  Foo foo = null;
  System.out.println(foo.add(10, 20));
 }
}
```

A.编译错误

B.正常运行，但无结果输出

C.运行输出：30

D.运行时抛出NullPointerException异常

7.下列说法正确的有: 【C】

A.class中的constructor不可省略

B.constructor必须与class同名，但方法不能与class同名

C.constructor在一个对象被实例化时执行

D.一个class只能定义一个constructor

8.关于 Java 中继承的特点，下列说法正确的是:【B】

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

9.请看下列代码：

```
class ClassA {}
class ClassB extends ClassA {}
class ClassC extends ClassA {}
public class Test{
 public static void main(String[] args) {
  ClassA p0 = new ClassA();
  ClassB p1 = new ClassB();
  ClassC p2 = new ClassC();
  ClassA p3 = new ClassB();
```

```
  ClassA p4 = new ClassC();
  <插入代码>
 }
}
```

下列选项中放置在<插入代码>处，使程序编译正确的是：【A】

A.p0 = p1;

B.p1 =p2;

C.p2 = p4;

D.p2 = (ClassC)p1;

10.请看下列代码，程序的输出结果是：【C】

```
class One {
 public One() {
  System.out.print(1);
 }
}
class Two extends One {
 public Two() {

  System.out.print(2);
 }
}
class Three extends Two {
 public Three() {
  System.out.print(3);
 }
}
public class Numbers {
 public static void main(String[] argv) {
  new Three();
 }
}
```

A.1

B.3

C.123

D.321

11.下列代码的运行结果是：【D】

```
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";
 }
}
```

A.peep

B.bark

C.meow

D.抛出运行时异常

12.下列代码的运行结果是：【A】

```
class SimpleCalc {
 public int value;
 public void calculate() {
  value += 7;
 }
}
public class MultiCalc extends SimpleCalc {
 public void calculate() {
  value -= 3;
 }
 public void calculate(int multiplier) {
  calculate();
  super.calculate();
  value *= multiplier;
 }
 public static void main(String[] args) {
  MultiCalc calculator = new MultiCalc();
  calculator.calculate(2);
  System.out.println("Value is: " + calculator.value);
 }
}
```

A.Value is: 8

B.Value is: -8

C.Value is: 12

D.Value is: -12

13.下列选项不属于属性的可见性有: 【C】

A.公有的

B.私有的

C.私有保护的

D.保护的

14.在 Java 中，关于 static 关键字的说法错误的是: 【D】

A.static 可以修饰方法

B.static 可以修饰变量

C.static 可以修饰代码块

D.static 修饰的方法，在该方法内部可以直接访问非静态的成员变量

15.下列类的定义，错误的是: 【D】

A.public class Test extends Object{……}

B.final class Operators{……}

C.class Point{……}

D.void class Point{……}

16.在 Java 中，关于 final 关键字的说法正确的是:【C】

A.如果修饰局部变量，必须初始化

B.如果修饰类，则该类只能被一个子类继承

C.如果修饰方法，则该方法不能在子类中被覆盖(override)

D.如果修饰方法，则该方法所在的类不能被继承

17.下列数组创建和初始化的方式不正确的是： 【D】

```
A. public class Test02 {
  static final int[] a = { 100,200 };
}
B. public class Test02 {
static final int[] a;
 static {
  a=new int[2];
  a[0]=100;
  a[1]=200;
 }
}
C.public class Test02 {
   final int[] a;
```

```
  public Test02(){
   a=new int[2];
   a[0]=100; a[1]=200;
  }
}
```

D.
```
public class Test02 {
 static final int[] a;
 static void init() {
  a = new int[3];
  a[0]=100;
  a[1]=200;
 }
}
```

18.关于抽象类的说法正确的是 【B】

A.抽象类中一定包含抽象方法，否则是出现编译错误

B.包含抽象方法的类一定是抽象类

C.抽象方法可以没有方法体，也可以有方法体

D.抽象类的子类一定不是抽象类

19.关于接口的说法错误的是：【D】

A.接口是特殊的抽象类

B.接口是抽象方法和常量值的定义的集合

C.当一个非抽象类实现一个接口时，需要实现接口中的所有方法

D.多个类可以实现一个接口，一个类只能实现一个接口

20.请看下列代码：

```
public class UserRequest {
 public void request(ServletAction action) {
  action.doService();
 }
 public static void main(String[] args) {
```

```
  UserRequest user = new UserRequest();
  user.request(new ServletAction() {
   public void doService() {
    System.out.println("处理请求");
   }
  });
 }
}
```

如果上述代码采用回调模式编写，下列关于 ServletAction 的定义正确的是：【D】

```
A.public static class ServletAction {
      public void doService();
}

B.public final class ServletAction {
      public void doService();
}

C.public class ServletAction {
      public void doService();
}

D. public interface ServletAction {
      public void doService();
}
```

21.下列不属于 Swing 提供的 Listener 的是:【D】

A.ActionListener

B.MouseListener

C.KeyListener

D.MemeryListener

22.请看下列代码：

```
public class Line {
 public static class Point { }
}
class Triangle {
  <插入代码>
}
```

在<插入代码>处，需要创建Point类的对象，下列选项正确的是：【B】

A.Point p = new Point();

B.Line.Point p = new Line.Point();

C.Line line = new Line() ; line.Point p = new line.Point();

D.Line.Point p = new Line().new Point();

23.请看下列代码：

```
public abstract class A {
 abstract void a1();
 void a2() {
 }
}
class B extends A {
 void a1() {
 }
 void a2() {
 }
}
class C extends B {
 void c1() {
 }
}
```

和

```
A x = new B(); C y = new C(); A z = new C();
```

下列选项中属于多态形式调用方法的是：【AB】

A.x.a2();

B.z.a2();

C.z.c1();

D.y.c1();

24.程序执行的结果是：【B】

```
public class Test{
 int x = 12;
 public void method(int x) {
   x += x;
   System.out.println(x);
 }
 public static void main(String[] args) {
   Test t = new Test();
   t.method(5);
 }
}
```

A.5

B.10

C.12

D.17

25.程序的执行结果是:【B】

```
public class Test {
        int x;
    public static void main(String [] args){
         Test t = new Test();
         t.x=5;
         change(t);
         System.out.println(t.x);
    }
public static void change(Test t){
         t.x=3;
```

```
    }
}
```

A.5

B.3

C.0

D.4

26.关于下列代码说法正确的是：【B】

```
public class  CreditCard {
 private String cardlD;
 private Integer limit;
 public String ownerName;
 public void setCardlnformation(String cardlD, String ownerName,
   Integer limit) {
  this.cardlD = cardlD;
  this.ownerName = ownerName;
  this.limit = limit;
 }
}
```

A.类CreditCard是完全封装的

B.属性ownerName打破了封装

C.属性cardlD 和 limit打破了封装

D.方法setCardlnformation打破了封装

27.关于下列代码说法正确的是：【B】

```
public class A {
 public void doit() {
 }
 public String doit() {
  return "a";
```

```
 }
 public double doit(int x) {
  return 1.0;
 }
}
```

A.无编译错误

B.代码 public String doit() { 行，出现编译错误

C.代码 public double doit(int x) {行，出现编译错误

D.代码 return "a";行处出现编译错误

28.关于下列代码说法正确的是：【D】

```
public class ItemTest {
  private  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.运行时异常抛出

C.运行后，fa对象属性id的值没有改变，应然是42

D.运行后，fa对象属性id的值改变成新的值69

29.请看下列代码：

```
public class Operator {
```

```
 public String find(){
  return "jessica";
 }
    public static void main(String[] args) {
     <插入代码>
     System.out.println(op.find());
 }
```

}如果运行上述代码，抛出NullPointerException异常，那么<插入代码>处应填入的代码是：【C】

A.Operator op=new Operator();

B.Operator op;

C.Operator op=null;

D.new Operator();

30.给出下面的代码段：【D】

```
public class Base{
  int w, x, y, z;
  public Base(int a, int b){
    x=a; y=b;
  }
  public Base(int a, int b, int c, int d)
  {
    <插入代码>
   w=d;z=c;
  }}
```

在<插入代码>处写下如下代码，正确的是:

A.Base(a, b);

B.super(a, b);

C.x=a, y=b;

D.this(a, b);

31.在Java中，所有类的基类是：【A】

A.java.lang.Object

B.java.lang.Class

C.java.applet.Applet

D.java.awt.Frame

32.下列代码运行的结果是：【A】

```
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);
 }
}
```

A.Value: 3

B.Value: 8

C.Value: 13

D.Value: 18

33.程序的执行结果是:【A】

```
public class Test {
    public static void main(String [] args){
        Child c = new Child();
    }
}
class Father{
    public Father(){
        System.out.println("父类无参构造函数");
    }
    public Father(String name){
        System.out.println("父类有参构造函数");
    }
}
class Child extends Father{
      public Child(){
        this("dd");
        System.out.println("子类无参构造函数");
    }
    public Child(String name){
        super("dd");
        System.out.println("子类有参构造函数");
    }
}
```

A.父类有参构造函数

子类有参构造函数

子类无参构造函数

B.父类无参构造函数

子类有参构造函数

子类无参构造函数

C.子类有参构造函数

子类无参构造函数

父类无参构造函数

D.子类无参构造函数

子类有参构造函数

父类无参构造函数

34.下列代码的运行结果是：【D】

```
public class Animal {
 public String noise() {
  return "peep";
 }
 public static void main(String[] args) {
  Cat cat =null;
  Animal animal = new Dog();
  if (animal instanceof Cat) {
   cat = (Cat) animal;
   System.out.println(cat.noise());
  }else{
   System.out.println("animal is not Cat's instance");
  }
 }
}
class Dog extends Animal {
 public String noise() {
  return "bark";
 }
}
class Cat extends Animal {
```

```
 public String noise() {
  return "meow";
 }
}
```

A.peep

B.bark

C.meow

D.animal is not Cat's instance

35.请看下列代码：

```
public class Blip {
 protected int blipvert(int x) {
  return 0;
 }
}
class Vert extends Blip {
 <插入代码> protected long blipvert(int x) { return 0; }  }
```

在<插入代码>处填入选项中的代码，使Vert类没有编译错误的是：【AC】

A.public int blipvert(int x) { return 0; }

B.private int blipvert(int x) { return 0; }

C.private int blipvert(long x) { return 0; }

D.protected long blipvert(int x) { return 0; }

36.下面关于import, class和package的声明顺序正确的是: 【A】

A.package, import, class

B.class, import, package

C.import, package, class

D.package, class, import

37.请看下列代码：【BC】

```
public class Foo {
```

```
  static void alpha() { /* more code here */}
  void beta() { /* more code here */}
}
```

下列说法正确的是：

A.Foo.beta()是调用beta方法的正确方式

B.Foo.alpha()是调用alpha方法的正确方式

C.beta方法可以直接调用alpha方法

D.alpha方法可以直接调用beta方法

38.关于下列代码说法正确的是：【A】

```
public interface DoStuff2 {
 float getRange(int low, int high);
}
interface DoMore {
 float getAvg(int a, int b, int c);
}
abstract class DoAbstract implements DoStuff2, DoMore {
}
class DoStuff implements DoStuff2 {
 public float getRange(int x, int y) {
  return 3.14f;
 }
}
interface DoAll extends DoMore {
 float getAvg(int a, int b, int c, int d);
}
```

A.无编译错误

B.代码abstract class DoAbstract implements DoStuff2, DoMore {行，编译错误

C.代码interface DoAll extends DoMore {行，编译错误

D.代码float getAvg(int a, int b, int c, int d);行，编译错误

39.请看下列代码编译和运行的结果是：【D】

```
public class Student {
 private String name="sun";
 public static void main(String[] args) {
  Student[] students=new Student[2];
  System.out.println(students[0].name);
  System.out.println(students.length);
 }
}
```

A.sun 2

B.null 2

C.null 1

D.运行时抛出NullPointerException异常

40.下面程序定义了一个类，关于该类说法正确的是:【D】

```
abstract class AbstractClass{ … }
```

A.可以使用new AbstractClass()；来实例化一个AbstractClass类的对象

B.该类不能被继承

C.该类的方法都不能被重载

D.该类的方法可以在子类中重写