第六章 类的特殊成员

# 选择题

1. 类中存在一个方法int getValue(int x){…}，以下能与这个方法重载的是( C )。
2. public getValue(int x){…};
3. void getValue(int x){…};
4. int getValue(int x, int y){…};
5. int get(int x){…};
6. 类中存在一个方法int getValue(double x){…}，以下不能与这个方法重载的是( C )。
7. int getValue(int x){…};
8. int getValue(float x){…};
9. int getValue(double t){…};
10. int getValue(double x, double y){…};
11. 以下关于重载的说法错误的是( A )
12. 重载的方法只要定义正确，调用时就不会有歧义性。
13. 重载的方法通常实现相似的功能
14. 重载的方法可以互相调用，提高代码复用
15. 重载是多态的一种表现形式
16. 已知A类具有以下四个成员方法，a是A类的对象，则a.fun(1, 3.14F)调用的是( D )。
17. public void fun(int a, double b){…}
18. public int fun(double a, double b){…}
19. public void fun(float a, float b){…}
20. public float fun(int a, float b){…}
21. 下列不是重载方法的特征的是( B )。
22. 参数个数不同
23. 返回值类型不同
24. 参数类型不同
25. 方法名相同
26. 一个类可同时定义许多同名的方法，这些方法的形式参数个数、类型或顺序各不相同，传回的值也可以不相同，这种特性称为( C )。
27. 继承
28. 重写
29. 重载
30. 封装
31. 定义类属性或类方法使用修饰符( A )。
32. static
33. package
34. private
35. public
36. 对以下类说法正确的是( D )。

class Demon {

double x;

static int y;

}

1. x称为类变量，y为实例变量
2. x、y均为类变量
3. x、y均为实例变量
4. x为实例变量，y为类变量
5. 下列关于静态属性的描述中错误的是( A )。
6. 静态属性只能通过类名访问，不能通过对象名访问
7. 静态属性可以通过类名访问，也可以通过对象名访问
8. 无论创建类的多少个实例，一个静态属性都被所有实例共享
9. 用关键字static修饰的成员变量为静态属性
10. 下列关于实例属性的描述中错误的是( B )。
11. 实例属性只能通过对象名访问，不能通过类名访问
12. 类的不同实例可以共享相同的实例属性空间
13. 不使用static修饰符的成员变量为实例属性
14. 实例变量取决于类的实例，每创建一个实例就会为其实例变量分配一次内存
15. 以下程序的结果是( A )。

public class Example {

public static void main(String args[]) {

static int x[] = new int[15];

System.out.println (x[5]);

}

}

1. 编译错误
2. 运行时异常
3. 输出0
4. 输出null
5. 下列关于静态方法的描述中错误的是( D )。
6. 可以使用"类名.方法名"来调用静态方法
7. 可以使用"对象.方法名"来调用静态方法
8. 程序的入口点main()方法必须被定义为静态方法
9. 构造方法不可以定义为静态方法
10. 下列关于静态方法和实例方法的描述中错误的是( B )。
11. 静态方法可以通过成员名直接访问同类静态成员
12. 静态方法可以通过成员名直接访问同类实例成员
13. 实例方法可以通过成员名直接访问同类静态成员
14. 实例方法可以通过成员名直接访问同类实例成员
15. 以下程序的结果是( D )。

class A {

private int counter = 0;

public static int getInstanceCount() {

return counter;

}

public A() {

counter++;

}

}

public class Test {

public static void main(String args[]) {

A al = new A();

A a2 = new A();

System.out.println(A.getInstanceCount());

}

}

1. 输出1
2. 输出2
3. 运行时异常
4. 编译错误
5. 一个静态方法不可能具有的特点是( C )。
6. 方法体只访问方法的参数
7. 方法体只访问同类的静态成员
8. 方法体只访问同类的实例成员
9. 方法体不访问同类的任何成员
10. 类A具有一个无形式参数无返回值的方法method，为了使用A.method()就可以调用它，该方法的形式可以为( A )。
11. static void method(){}
12. public void method(){}
13. final void method(){}
14. abstract void method(){}
15. 下列关于this的描述中，错误的是( D )。
16. 每个对象都有一个名为this的引用，它指向当前对象本身
17. this可以用于访问本类的成员变量
18. this可以用于访问本类的成员方法
19. this可以出现在static方法中
20. this关键字的含义是表示( B )。
21. 当前变量
22. 当前对象
23. 当前类
24. 当前方法
25. 成员的访问限定符不包括( D )。
26. public
27. private
28. default
29. final
30. 以下关于类的final成员变量说法正确的是( A )。
31. 可以在定义时或在构造方法中初始化
32. 可以在任意成员方法中赋值
33. 对于一个类的所有对象，同名final成员变量的值都相同
34. 如果没有在定义时或构造方法里初始化，则被初始化为默认值
35. 以下关于final和static关键字的说法正确的是( C )。
36. 都可以修饰成员方法、成员变量和构造方法
37. 都可以修饰局部变量
38. 用static final同时修饰的成员变量不能在构造方法中初始化
39. 不可以使用static final同时修饰成员方法或成员变量
40. 以下程序的运行结果是( A )。

public class Test {

private final int id;

public Test(int id) {

this.id = id;

}

public void updateId(int newId) {

id = newId;

}

public static void main(String args[]) {

Test fa = new Test(42);

fa.updateId(69);

System.out.println(fa.id);

}

}

1. 编译错误
2. 运行时异常
3. 42
4. 69
5. 以下关于toString()方法的说法不正确的是( D )。
6. 所有类都有隐含的toString()方法，默认返回对象地址相关的字符串
7. 可以按照需要重新定义toString()方法返回自定义的字符串
8. 当对象需要作为字符串使用时toString()通常将被自动调用
9. 若类没有定义toString()方法，则不能调用该方法
10. 以下关于类的equals()说法正确的是( B )。
11. 所有的类都有隐含的equals()方法，默认对比当前对象和形式参数引用的对象的属性是否相同
12. 定义一个类时可以按照需要重新定义equals()方法
13. 比较对象时，equals()方法和==的作用完全相同，只能比较两个变量是否引用同一对象
14. 若一个类没有定义equals()方法，则不能调用该方法
15. 以下程序的输出结果是( B )。

public class Test {

protected int a, b;

public Test(int a, int b) {

this.a = a;

this.b = b;

}

public boolean equals(Test te) {

return te.a == this.a && te.b == this.b;

}

public static void main(String[] args) {

Test a = new Test(3, 4);

Test b = new Test(1, 4);

System.out.println(a.equals(b));

}

}

1. true
2. false
3. 不确定
4. 编译错误
5. 以下程序的输出结果是( B )。

public class Test {

protected int a, b;

public Test(int a, int b) {

this.a = a;

this.b = b;

}

public static void main(String[] args) {

Test a = new Test(1, 4);

Test b = new Test(1, 4);

System.out.println(a.equals(b));

}

}

1. true
2. false
3. 不确定
4. 编译错误
5. 以下程序的输出结果是( B )。

public class Test {

protected int a, b;

public Test(int a, int b) {

this.a = a;

this.b = b;

}

public static void main(String[] args) {

Test a = new Test(1, 4);

Test b = new Test(1, 4);

System.out.println(a == b);

}

}

1. true
2. false
3. 不确定
4. 编译错误
5. 以下关于递归的说法不正确的是( D )。
6. 递归可以分为直接递归和间接递归
7. 递归方法内一定要有递归出口
8. 递归出口可以是在某种条件下才继续递归，也可以是在某种条件下结束递归
9. 一个问题的递归算法一定比非递归算法效率低