一、填空题

1. 使用___static____修饰符定义的类成员,可以通过类直接访问而不需要创建对象后再访问。
2. 用_abstract_修饰符定义的方法,没有方法体,使用_abstract_修饰符定义的类不能实例化。
3. 类的中一个成员是一个类的对象时,如果该成员没有被初始化,则该对象的初始值是___null____。
4. 在子类构造函数中使用_____super_____关键字来调用父类的构造函数。
5. Java 接口中可以声明_____公共静态常量_____和_____公共抽象实例方法_____。
6. 用 final 关键字修饰一个类表示___该类不能被继承_____。
7. 在子类的实例方法 m 中要调用父类被覆盖的实例方法 m(方法 m 不带参数且没有返回值)的语句是 ____super.m()_____。
8. 如有以下类的定义:

```
abstract class A {
        public void fa () {};
        public abstract void fb();
        public abstract void fc();
}
interface I {
        void fx();
}
abstract class B extends A    {
        public void fb() {};
        public abstract void fd();
}
public class C extends B implements I {
        …
}
```

则在在 class C 中必须要实现的方法为____ fc(),fd(),fx()_____。

9. 如有下列接口和类的定义:

```
interface I1{      }
interface I2 extends I1{    }
```

class A implements I2{ }

class B extends A{ }

则 B 类的一个实例对象 o 的类型可以是___B、A、I2、I1_____。

B, A, I2, I1, Object

10. 下列程序的输出结果是_____ three two one 1 2_____。

```
class C {
    int x;
    String y;
    public C() {
        this(“1”);
        System.out.print(“one ”);
    }
    public C(String y) {
        this(1, “2”);
        System.out.print(“two ”);
    }
    public C(int x, String y) {
        this.x = x;
        this.y = y;
        System.out.print(“three ”);
    }
    public static void main(String[] args) {
        C c = new C();
        System.out.println(c.x + “ ” + c.y);
    }
}
```

11. 在 Java 中对于程序可能出现的必检异常，要么用 try…catch 语句捕获并处理它，要么使用____throw_____语句抛出它，由上一级调用者来处理。

12. 在 Java 中异常分为必检异常和非必检异常二种类型，其中表达式 10/0 会抛出_ArithmeticException__ 非必检异常类型异常，打开一个不存在的文件会抛出_FileNotFoundException____必检异常类型异常，通过空引用调用实例方法会抛出__NullPointerException_____ 非必检异常类型异常，数组越界访问会抛出___ArrayIndexOutOfBoundsException_____非必检异常类型异常，用 throw 语句抛出一个自定义的 Exception 子类异常是__编译（必检）_类型异常。

二、单项选择题

1. 接口中的成员变量被隐含地声明为___A_____。

A. public static final　　B. public final

C. public static　　D. public abstract

2. 下列叙述中正确的是____B_____。

A. Java 中一个类可以有多个直接父类，可以实现多个接口

B. Java 中一个类只可以有一个直接父类，可以实现多个接口

C. Java 中一个类只可以有一个直接父类，只可以实现一个接口

D. Java 中一个类可以有多个直接父类，只可以实现一个接口

3. 关于子类覆盖（Override）父类的方法，下列说法正确的是__D____。B

A．子类方法与父类方法形式参数个数或者类型不同

B．子类方法与父类方法的形式参数个数、形参类型、返回类型相容

C．子类方法与父类方法的访问权限必须相同

D．子类方法与父类方法形式参数名称必须相同

4. 定义类时不能使用的修饰符是_____D____。

A． abstract　　B．final　　C．public　　D．abstract final

5. 下面程序运行后的输出结果为__B___。

```
class B{
    int y=3;
    static void showy( ){System.out.println("y="+y); }
}
class TestB{
    public static void main(String aaa []){
        B a1=new B( );
        a1.y++;
        a1.showy( );
    }
}
```

A.　y=3;　　B.　y=4;

C.　y=5;　　D.　程序编译出错

6. 给定以下类的定义

```
public class A{
    public A(){
        System.out.println(“Constructor”);
    }
    public static int i = 0;
}
```

则下列语句中会在控制台中打印出字符串 Constructor 的是____C_____。

A. A a = null;　　B. int k = A.i;

C. int k = new A().i;　　D. A[] array = new A[1];

7. class A extends B implements C，假定 A 和 B 有缺省构造方法，则下面的语句编译和运行正确的是___C___。

A．A a = new A( );　B b = a;　C c = b;

B．B b = new B( );　A a = (A) b;

C．A a = new A( );　B b = a; C c1 = a ,c2 = new A( );

D．A a = new A( );　C c = new A( );　B b = new C( );

8. 给定下列程序，程序的输出结果为___c___。 A

```
class Base {
    public Base(String s) {
        System.out.print("B");
    }
}
public class Derived extends Base {
    public Derived (String s) {
        System.out.print("D");
    }
    public static void main(String [] args) {
        new Derived ("C");
    }
}
```

那么结果为？

A. 编译错误　　B. DB　　C. BD　　D. BDC

9. 已知如下目录结构（dira和dirb为目录）

```
dira
|---A.class
```

```
    |---dirb
        |---B.class
```

和如下源代码：

```
    import dira.*;
    class C {
        A a;
        B b;
    }
```

那么要使源代码通过编译，需要在源代码中添加___A___。 C

A. package dira;
B. package dirb;
C. package dira.dirb;
D. package dirb.dira;

10. 给定下列程序

```
interface I { }
class A implements I { }
class B extends A { }
class C extends B {
    public static void main(String[] args) {
        B b = new B();
        _____________
    }
}
```

在横线处添加下面哪条语句运行时会产生异常 C 。

A. A a = b;
B. I i = b;
C. C c = (C) b;
D. B d = (B) (A) b;

11. 下面程序的输出结果是___C___。

```
class C {
    public static void main(String[] args) {
        try {
            System.out.print(10 + 10 / 0);
        } catch(NullPointerException e1) {
            System.out.print(“a”);
        } catch(RuntimeException e2) {
            System.out.print(“b”);
        } finally {
            System.out.print(“c”);
        }
    }
```

}

A. a　　B. ac　　C. bc　　D. abc

12．下面程序的执行结果是___D_____。

```
public class MyProgram{
    public static void main (String args[]){
        try{
            System.out.print(“Hello Java.”);
        }
        finally{
            System.out.print(“Finally Java.”);
        }
    }
}
```

A． 无法编译，因为没有 catch 子句　　B． Hello Java.

C． Finally Java.　　D. Hello Java. Finally Java.

13. 下面程序的执行结果是____A_____。C

```
public class Test7 {
    public static void main(String[] args){
        new B().display();
    }
}
class A{
    public void draw() {
        System.out.print("Draw A.");
    }
    public void display() {
        draw();
        System.out.print("Display A.");
    }
}
class B extends A{
    public void draw() {
        System.out.print("Draw B.");
    }
    public void display() {
        super.display();
        System.out.print("Display B.");
    }
}
```

A. Draw A.Display A.Display B.　　B. Draw A.Display B.Display A.

C. Draw B.Display A.Display B.　　D. Draw B.Display B.Display A.

14. 语句 int[] m = new int[5];则 m[5]=10;会有___C_____。

A. 编译运行都正确；

B. 编译不正确

C. 会引发 ArrayIndexOutOfBoundsException 异常

D. 会引发 NullPointerException 异常

15. 下面程序执行结果是___D_____。

```
public class Test {
    public static void main(String args[]) {
        try {
            System.out.print("try.");
            return;
        } catch(Exception e){
            System.out.print("catch.");
        }finally {
            System.out.print("finally.");
        }
    }
}
```

A. try.catch.finally.　　B. try.

C. try.catch.　　D. try.finally

16. 给定下列程序，下面说法正确的是____B_____。

```
public class Test2_16 {
    public void m1() throws IOException{
        try {
            throw new IOException();
        }
        catch (IOException e){

        }
    }

    public void m2(){
        m1();
    }
}
```

A. 因 m1 方法里已经捕获了异常，因此 m2 里调用 m1()时不用处理异常，程序编译通过

B. m2 或者用 throws 声明异常，或者在方法体里面用 try/catch 块去调用 m1 并捕获异常，否则编译报错

C. m2 方法体里面必须用 try/catch 块去调用 m1 并捕获异常，否则编译报错

D. m2 方法必须用 throws 声明异常，否则编译报错

17. 给定下列程序，下面说法正确的是____B_____。A

```
public class Test2_17 {
    public void m1() throws RuntimeException{
        throw new RuntimeException();
    }

    public void m2(){
        m1();
    }
}
```

A. 程序编译通过

B. m2 或者用 throws 声明异常，或者在方法体里面用 try/catch 块去调用 m1 并捕获异常，否则编译报错

C. m2 方法体里面必须用 try/catch 块去调用 m1 并捕获异常，否则编译报错

D. m2 方法必须用 throws 声明异常，否则编译报错

三、判断对错题

1. 包含有抽象方法的类必须是抽象类，抽象类也必须包含有抽象方法。(false)
2. 一个接口只能继承一个直接父接口。（ false ）
3. 非抽象类的子类不能是抽象类。（ false ）
4. 接口类型的引用变量能直接指向一个实现了该接口的类的实例而无需强制类型转换。(true)
5. import 语句通常出现在 package 语句之前。(false)
6. 抽象类中不能定义构造方法。(true)
7. this 关键字可以在类的所有方法里使用。(false)
8. 类 A 的所有实例方法都可以在 A 的子类中进行覆盖(Override)。(false)
9. 在类的静态初始化块里可以初始化类的静态数据成员和实例数据成员。(false)

10. 由于抽象类不能被实例化，因此方法的参数类型和返回类型都不能是抽象类类型。(true)

false

四、阅读题

阅读下列程序，写出输出结果：

```
class SuperClass{
    static int i = 10;
    static{
        System.out.println(" static in SuperClass");
    }
    {
        System.out.println("SupuerClass is called");
    }
}
class SubClass extends SuperClass{
    static int i = 15;
    static{
        System.out.println(" static in SubClass");
    }
    SubClass( ){
        System.out.println("SubClass is called");
    }
    public static void main(String[] args){
        int i = SubClass.i;
        new SubClass( );
        new SuperClass( );
    }
}
```

运行结果：

____ static in SuperClass _____

____ static in SubClass _____

____ SupuerClass is called _____

____ SubClass is called _____

____ SupuerClass is called _____

五、阅读题

下面程序，写出指定语句的输出结果，并解释原因。

```
public class Test5 {
    public static void main(String... args){
        C o1 = new D();
```

```
        o1.m(1,1);              //①
        o1.m(1.0,1.0);          //②
        o1.m(1.0f, 1.0f);       //③

        D o2 = new D();
        o2.m(1,1);              //④
        o2.m(1.0,1.0);          //⑤
        o2.m(1.0f, 1.0f);       //⑥
    }
}

class C{
    public void m(int x, int y) {
        System.out.println("C's m(int,int)");
    }
    public void m(double x, double y) {
        System.out.println("C' m(double,double)");
    }
}

class D extends C{
    public void m(float x, float y) {
        System.out.println("D's m(float,float)");
    }
    public void m(int x, int y)   {
        System.out.println("D's m(int,int)");
    }
}
```

语句①的输出结果为_____D's m(int,int)_____,原因是 D 拥有 m(int x, int y)的方法，覆盖了 C 中的相应方法，调用 D 中的函数_____。

语句②的输出结果为_C' m(double,double) ,原因是 o1 的声明类型为 C，而且传进的参数 1.0 为 Double 类型，直接调用 C 中的该函数_____。

语句③的输出结果为____C' m(double,double)___ ,原因是_____o1 的声明类型为 C，而且传进的参数 1.0 为 double 类型，直接调用 C 中的该函数_____。

语句④的输出结果为 D's m(int,int) ,原因是 o2 的声明和运行类型都为 D，所以直接调用 D 中相应的方法 。

语句⑤的输出结果为 C' m(double,double) ,原因是 传入的参数为 double 类型，在 D 中没有找到相应的方法 ,于是想 D 的父类 C 中寻找调用 。

语句⑥的输出结果为 D's m(float,float) ,原因是 传入 float 类型的参数，对应 D 中的相应函数，直接调用 。

六、阅读题

阅读下面程序，写出指定语句的输出结果，并解释原因。

```
public class Test_Hide_Override {
    public static void main(String... args){
        A o = new C();
        o.m1();             //①
        o.m2();             //②
        ((B)o).m1();        //③
        ((A)(B)o).m1();     //④
        ((A)(B)o).m2();     //⑤
    }
}

class A{
    public static void m1(){    System.out.println("A's m1"); }
    public void m2(){ System.out.println("A's m2"); }
}

class B extends A{
    public static void m1(){    System.out.println("B's m1"); }
    public void m2(){ System.out.println("B's m2"); }
}

class C extends B{
    public static void m1(){    System.out.println("C's m1"); }
    public void m2(){ System.out.println("C's m2"); }
}
```

语句①的输出结果为 A's m1 ,原因是静态方法不具有多态性，o 的声明类型为 A，就调用 A 中的方法。

语句②的输出结果为 C's m2 ,原因是实例方法看运行时对象的类型，o 在运行时是 C 类型，就调用 C 中的实例方法。

语句③的输出结果为 B's m1 ,原因是 通过强制类型转换把 o 当前类型解释为 B，所以调用了 B 中的静态 m1 方法。

语句④的输出结果为 A's m1 ,原因是o最终被转换成了A类型,调用A中的m1方法。

语句⑤的输出结果为 C's m2 ,原因是 实例方法看运行时类型,所以还是调用C中的m2方法 。

七、编程题第二题

```
public boolean equals(Object obj){
    if(obj instanceof Course){
        Course o = (Course) obj;
        if (o.getCourseName().equals(this.courseName)==false){
            return false;
        }
        if (o.getTeacher().equals(this.teacher)==false){
            return false;
        }
        if (o.getStudents().size()==this.getStudents().size()){
            return o.getStudents().containsAll(this.students);
        }else{
            return false;
        }
    }else {
        return false;
    }
}
```

```
@Override
public Object clone()throws CloneNotSupportedException{
    Course newObj = (Course) super.clone();
    newObj.teacher = (Person) this.teacher.clone();
    newObj.courseName = new String(this.courseName);
    newObj.students = new ArrayList<>();
    for (Person current:this.students){
        newObj.students.add((Student) current.clone());
    }
    return newObj;
}
```

八、编程题第三题

```
public class CompositeIterator implements Iterator{
    protected List<Iterator> iterators = new ArrayList<Iterator>();
    public CompositeIterator(Iterator iterator){
        iterators.add(iterator);
    }
    @Override
    public boolean hasNext(){
        if(iterators.size()>0){
            Iterator it = iterators.get(0);
            if(!it.hasNext()){
                iterators.remove(0);
                return hasNext();
            }else {
                return true;
            }
        }else{
            return false;
        }
    }
    @Override
    public Component next(){
        if(hasNext()){
            Iterator it = iterators.get(0);
            Component c = it.next();
            iterators.add(c.iterator());
            return c;
        }else return null;
    }
}
public Iterator iterator(){
    Iterator it = new CompositeIterator(this.childs);
    return it;}
```