**四、程序阅读题**

1、阅读以下程序，写出输出结果。

public class Abc {

public static void main(String args[]) {

Ab s = new Ab("Hello!", "I love JAVA.");

System.*out*.println(s);

}

}

class Ab {

String s1;

String s2;

Ab(String str1, String str2) {

s1 = str1;

s2 = str2;

}

public String toString() {

return s1 + s2 + "You?";

}

}

2、阅读以下程序，写出输出结果。

**public** **class** Compare {

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

String str1 = "abc";

String str2 = **new** String("abc");

String str3 = "ab" + "c";

String str4 = **new** String(str2);

String str5 = str1;

System.*out*.println(str1 == str2);

System.*out*.println(str2 == str3);

System.*out*.println(str2 == str4);

System.*out*.println(str5 == str1);

}

}

3、阅读以下程序，写出输出结果。

public class GroupTwo {

private int count;

public class Student {

String name;

public Student(String n1) {

name = n1;

count++;

}

public void Output() {

System.out.println(this.name);

}

}

public void output() {

Student s1 = new Student("Johnson");

s1.Output();

System.out.println("count=" + this.count);

}

public static void main(String args[]) {

GroupTwo g2 = new GroupTwo();

g2.output();

}

}

4、阅读以下程序，写出输出结果。

class superClass {

int y;

superClass() {

y = 30;

System.out.println("in superClass:y=" + y);

}

void doPrint() {

System.out.println("In superClass.doPrint()");

}

}

class subClass extends superClass {

int y;

subClass() {

super();

y = 50;

System.out.println("in subClass:y=" + y);

}

void doPrint() {

super.doPrint();

System.out.println("in subClass.doPrint()");

System.out.println("super.y=" + super.y + " sub.y=" + y);

}

}

public class inviteSuper {

public static void main(String args[]) {

subClass subSC = new subClass();

subSC.doPrint();

}

}

5、阅读以下程序，写出输出结果。

public class GroupThree {

private static int count;

private String name;

public class Student {

private int count;

private String name;

public void Output(int count) {

count++;

this.count++;

GroupThree.count++;

GroupThree.this.count++;

System.out.println(count + " " + this.count + " " + GroupThree.count + " " + GroupThree.this.count++);

}

}

public Student aStu()

{

return new Student();

}

public static void main(String args[]) {

GroupThree g3 = new GroupThree();

g3.count = 10;

GroupThree.Student s1 = g3.aStu();

GroupThree.Student s1.Output(5);

}

}

6、阅读以下程序，写出输出结果。

class Mammal {

private int n = 40;

void crySpeak(String s) {

System.out.println(s);

}

}

public class Monkey extends Mammal {

void computer(int aa, int bb) {

int cc = aa \* bb;

System.out.println(cc);

}

void crySpeak(String s) {

System.out.println("\*\*" + s + "\*\*");

}

public static void main(String args[]) {

Mammal mammal = new Monkey();

mammal.crySpeak("I love this game");

Monkey monkey = (Monkey) mammal;

monkey.computer(10, 10);

}

}

7、阅读以下程序，写出输出结果。

**public** **class** Flower {

**int** petalCount = 0;

String s = "initial value";

Flower(**int** petals) {

petalCount = petals;

print("Constructor w/ int arg only, petalCount= " + petalCount);

}

Flower(String ss) {

print("Constructor w/ String arg only, s = " + ss);

s = ss;

}

Flower(String s, **int** petals) {

**this**(petals);

**this**.s = s; // Another use of "this"

print("String & int args");

}

Flower() {

**this**("hi", 47);

print("default constructor (no args)");

}

**void** printPetalCount() {

print("petalCount = " + petalCount + " s = " + s);

}

**void** print(String s) {

System.*out*.println(s);

}

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

Flower x = **new** Flower();

x.printPetalCount();

}

}

8、阅读以下程序，写出输出结果。

class Cup {

Cup(int marker) {

System.out.println("Cup(" + marker + ")");

}

void f(int marker) {

System.out.println("f(" + marker + ")");

}

}

class Cups {

static Cup cup1;

static Cup cup2;

static {

cup1 = new Cup(1);

cup2 = new Cup(2);

}

Cups() {

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

}

}

public class ExplicitStatic {

public static void main(String[] args) {

System.out.println("Inside main()");

Cups.cup1.f(99);

}

}

9、阅读以下程序，写出输出结果。

class Tree {

int height;

Tree() {

System.out.println("Planting a seedling");

height = 0;

}

Tree(int initialHeight) {

height = initialHeight;

System.out.println("Creating new Tree that is " +

height + " feet tall");

}

void info() {

System.out.println ("Tree is " + height + " feet tall");

}

void info(String s) {

System.out.println (s + ": Tree is " + height + " feet tall");

}

}

public class Overloading {

public static void main(String[] args) {

for(int i = 0; i < 5; i++) {

Tree t = new Tree(i);

t.info();

t.info("overloaded method");

}

// Overloaded constructor:

new Tree();

}

}

10、阅读以下程序，写出输出结果。

**class** A {

**double** f(**double** x, **double** y) {

**return** x \* y;

}

}

**class** B **extends** A {

**double** f(**double** x, **double** y) {

**return** x + y;

}

}

**public** **class** Test {

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

B obj = **new** B();

System.*out*.println("The program output is " + obj.f(6, 7));

}

}

11、阅读以下程序，写出输出结果。

class LargeCup {

LargeCup(int marker) {

System.*out*.println("LargeCup(" + marker + ")");

}

void f1(int marker) {

System.*out*.println("f1(" + marker + ")");

}

}

class Shelf {

static LargeCup *cup1* = new LargeCup(1);

Shelf() {

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

*cup2*.f1(1);

}

void f2(int marker) {

System.*out*.println("f2(" + marker + ")");

}

static LargeCup *cup2* = new LargeCup(2);

}

class Cupshelf {

static LargeCup cup3 = new LargeCup(3);

static LargeCup *cup4* = new LargeCup(4);

Cupshelf() {

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

*cup4*.f1(2);

}

void f3(int marker) {

System.*out*.println("f3(" + marker + ")");

}

static LargeCup *cup5* = new LargeCup(5);

}

public class Initialization {

static Shelf *shelf* = new Shelf();

static Cupshelf *cupshelf* = new Cupshelf();

public static void main(String[] args) {

System.*out*.println("Creating new Cupshelf1() in main");

new Cupshelf();

System.*out*.println("Creating new Cupshelf2() in main");

new Cupshelf();

*shelf*.f2(1);

*cupshelf*.f3(1);

new Initialization();

}

}

12、阅读以下程序，写出输出结果。

**public** **class** GroupFive {

**public** **abstract** **class** Student\_abstract {

**int** count;

String name;

**public** **abstract** **void** output();

}

**public** **class** Student **extends** Student\_abstract {

**public** Student(String n1) {

name = n1;

count++;

}

**public** **void** output() {

System.*out*.println(**this**.name + " count=" + **this**.count);

}

}

**public** GroupFive() {

Student s1 = **new** Student("A");

s1.output();

Student s2 = **new** Student("B");

s2.output();

}

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

GroupFive g5 = **new** GroupFive();

}

}

输出：

13、阅读以下程序，写出输出结果。

class Plate {

Plate(int marker) {

System.*out*.println("Plate(" + marker + ")");

}

void f1(int marker) {

System.*out*.println("f1(" + marker + ")");

}

}

class Desk {

static Plate *plate1* = new Plate(1);

Desk() {

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

p*late2*.f1(1);

}

void f2(int marker) {

System.*out*.println("f2(" + marker + ")");

}

static Plate *plate2* = new Plate(2);

}

class Board {

Plate plate3 = new Plate(3);

static Plate *plate4* = new Plate(4);

Board() {

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

*plate4*.f1(2);

}

void f3(int marker) {

System.*out*.println("f3(" + marker + ")");

}

static Plate *plate5* = new Plate(5);

}

public class Initialization {

static Desk *desk* = new Desk();

Board *Board* = new Board();

public static void main(String[] args) {

System.*out*.println("Creating new Board() in main");

new Board();

System.*out*.println("Creating2 new Board() in main");

new Board();

*desk*.f2(1);

*Board*.f3(1);

}

}

14、阅读以下程序，写出输出结果。

class Swap {

void sw1 (Person x, Person y){

int a = y.id;

y.id = x.id;

x.id = a;

}

void sw2 (Person x, Person y){

Person e;

e = x;

y = x;

x = e;

}

}

public class Person {

int id = 0;

Person (int id) {

this.id = id;

}

public static void main(String[] args) {

Swap cid = new Swap ();

Person p1 = new Person (10);

Person p2 = new Person (11);

Person p3 = new Person (12);

Person p4 = new Person (13);

cid.sw1(p1,p2);

cid.sw2(p3,p4);

System.*out*.println("p1: " + p1.id + "p2: " + p2.id);

System.*out*.println("p3: " + p3.id + "p4: " + p4.id);

}

}

**15、**阅读以下程序，写出输出结果。

public class Foo {

public static void main (String [] args) {

StringBuffer a = new StringBuffer (“This is in SCUT”);

StringBuffer b = new StringBuffer (“Twice”);

operate(a,b);

System.out.printIn(a + “,” +b);

}

static void operate (StringBuffer x, StringBuffer y) {

x.append(y);

y = x;

}

}

16、阅读以下程序，写出输出结果。

import static net.mindview.util.Print.\*;

class Candy {

static { print("Loading Candy"); }

}

class Gum {

static { print("Loading Gum"); }

}

class Cookie {

static { print("Loading Cookie"); }

}

public class SweetShop {

public static void main(String[] args) {

print("inside main");

new Candy();

print("After creating Candy");

try {

Class.forName("Gum");

} catch(ClassNotFoundException e) {

print("Couldn’t find Gum");

}

print("After Class.forName(\"Gum\")");

new Cookie();

print("After creating Cookie");

}

}

17、阅读以下程序，写出输出结果。

public class Class3 {

public static void main(String args[]) {

SubSubClass x = new SubSubClass(10, 20, 30);

x.show();

}

}

class SuperClass {

int a, b;

SuperClass(int aa, int bb) {

a = aa;

b = bb;

}

void show() {

System.*out*.println("a=" + a + "\nb=" + b);

}

}

class SubClass extends SuperClass {

int c;

SubClass(int aa, int bb, int cc) {

super(aa, bb);

c = cc;

}

}

class SubSubClass extends SubClass {

int a;

SubSubClass(int aa, int bb, int cc) {

super(aa, bb, cc);

a = aa + bb + cc;

}

void show() {

System.*out*.println("a=" + a + "\nb=" + b + "\nc=" + c);

}

}

18、阅读以下程序，写出输出结果。

class SmallPlate {

SmallPlate(int i) {

System.out.println("Plate constructor");

}

}

class DinnerPlate extends SmallPlate {

DinnerPlate(int i) {

super(i);

System.out.println("DinnerPlate constructor");

}

}

class Utensil {

Utensil(int i) {

System.out.println("Utensil constructor");

}

}

class Spoon extends Utensil {

Spoon(int i) {

super(i);

System.out.println("Spoon constructor");

}

}

class Fork extends Utensil {

Fork(int i) {

super(i);

System.out.println("Fork constructor");

}

}

class Knife extends Utensil {

Knife(int i) {

super(i);

System.out.println("Knife constructor");

}

}

// A cultural way of doing something:

class Custom {

Custom(int i) {

System.out.println("Custom constructor");

}

}

public class PlaceSetting extends Custom {

private Spoon sp;

private Fork frk;

private Knife kn;

private DinnerPlate pl;

public PlaceSetting(int i) {

super(i + 1);

sp = new Spoon(i + 2);

frk = new Fork(i + 3);

kn = new Knife(i + 4);

pl = new DinnerPlate(i + 5);

System.out.println("PlaceSetting constructor");

}

public static void main(String[] args) {

PlaceSetting x = new PlaceSetting(9);

}

}

19、阅读以下程序，写出输出结果。

**public** **class** Unchecked {

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

**try** {

*method*();

} **catch** (Exception e) {

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

} **finally** {

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

}

}

**static** **void** method() {

**try** {

*wrench*();

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

} **catch** (ArithmeticException e) {

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

} **finally** {

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

}

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

}

**static** **void** wrench() {

**throw** **new** NullPointerException();

}

}

20、阅读以下程序，写出输出结果。

**class** Member {

**static** **int** *classVar*;

**int** instanceVar;

**static** **void** setClassVar(**int** i) {

*classVar* = i;

}

**static** **int** getClassVar() {

**return** *classVar*;

}

**void** setInstanceVar(**int** i) {

*classVar* = i;

instanceVar = i;

}

**int** getInstanceVar() {

**return** instanceVar;

}

}

**public** **class** ClassMethodTest {

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

Member m1 = **new** Member();

Member m2 = **new** Member();

m1.*setClassVar*(1);

m2.*setClassVar*(2);

System.*out*.println("m1.classVar=" + m1.*getClassVar*() + " m2.classVar="

+ m2.*getClassVar*());

m1.setInstanceVar(11);

m2.setInstanceVar(22);

System.*out*.println("m1.InstanceVar=" + m1.getInstanceVar()

+ " m2.InstanceVar=" + m2.getInstanceVar());

}

}

输出：