单选题

1. 定义了一个有10个int型元素的数组a后，下面引用错误的是（ ）。

    - A. a[0]=1;
    - B. a[10]=5;
    - C. a[1]=4*6;
    - D. a[2]=a[1]*a[0];

2. 下列数组的初始化语句中正确的是（ ）。

    - A. char c[]="hello";
    - B. char c[10]="hello";
    - C. char c[]={'h','e','l','l','o'};
    - D. char c[]={'hello'};

3. 若父类为Father，子类为Son，并且有如下语句，则下列选项中正确的是（ ）。

    - Father f= new Father();
    - Son s=(Son)f;
    - A. 只有第1行能通过编译
    - B. 第1、2行能通过编译,并正确运行
    - C. 第1、2行能通过编译,但第2行运行时出错
    - D. 两行都不能通过编译

4. 若有一个 final 变量 PI，其类型为 double，定义为 final double PI = 3.14159;，以下（ ）操作是不合法的。

    - A. System.out.println(PI);
    - B. double radius = 2.0; double area = PI * radius * radius;
    - C. PI = 3.14;
    - D. 将 PI 作为参数传递给一个方法

5. 抽象类和接口都具有的特点是（ ）。

    - A. 可以包含抽象方法
    - B. 可以被实例化
    - C. 成员变量只能是public
    - D. 都有默认构造函数

6. 对于 super 关键字，以下说法正确的是（ ）。

    - A. super 只能用于调用父类的构造方法
    - B. super 可以在任何类中使用
    - C. super 用于访问父类中被隐藏或重写的成员
    - D. super 的使用方法和 this 完全相同

7. 当一个类中有多个构造方法时，使用（ ）关键字可以在一个构造方法中调用另一个构造方法。

    - A. this
    - B. super

- C. new
- D. instanceof

8. 下列关于this的描述中,错误的是（ ）。

   - A. 每个对象都有一个名为this的引用,它指向当前对象本身
   - B. this可以调用本类中的属性
   - C. this可以调用本类中的其他方法
   - D. this不可以调用本类中的其他构造方法

9. 在Java中， this 关键字主要用于（ ）。

   - A. 调用父类的方法
   - B. 调用当前类的其他构造方法
   - C. 引用当前对象本身
   - D. 访问父类的成员变量

10. 如果父类 Vehicle 有属性 speed ，子类 Car 在继承 Vehicle 后，（ ）。

    - A. Car 一定能直接访问 speed 属性
    - B. 若 speed 在 Vehicle 中是私有属性， Car 不能直接访问
    - C. Car 只能通过新定义的方法来访问 speed
    - D. Car 可以随意修改 speed 的值

11. 假设有父类 Animal 和子类 Dog ，如果 Animal 中有一个方法 makeSound() ，在 Dog 类中重写这个方法时，以下（ ）是正确的。

    - A. 重写后的方法不能有相同的方法名
    - B. 重写后的方法返回类型必须与父类方法返回类型严格相同
    - C. 重写后的方法访问权限不能比父类方法访问权限更严格
    - D. 重写后的方法参数列表可以和父类方法不同

12. 在Java中，以下关于继承的说法正确的是（ ）。

    - A. 一个类只能继承一个父类
    - B. 一个类可以继承多个父类
    - C. 子类可以继承父类的所有成员（包括私有成员）
    - D. 继承关系中，父类一定比子类功能更强大

13. 以下哪个关键字用于在类中定义一个常量（ ）？

    - A. final
    - B. static
    - C. const
    - D. public

14. 如果一个方法或变量是"private"访问级别，那么它的访问范围是：（ ）。

    - A.在当前类，或者子类中
    - B.在当前类或者它的父类中
    - C.在当前类，或者它所有的父类中

- D.在当前类中

15. 下列关于被私有访问控制符private修饰的成员变量的说法中，正确的是（ ）。

    - A.可以被3种类所引用:该类自身、与它在同一个包中的其他类、在其他包中的该类的子类
    - B.可以被两种类访问和引用:该类本身、该类的所有子类
    - C.只能被该类自身所访问和修改
    - D.只能被同一个包中的类访问

16. 以下关于类的成员访问修饰符的说法正确的是（ ）。

    - A. private修饰的成员只能在同一个包中访问
    - B. protected修饰的成员可以被任何类访问
    - C. public修饰的成员可以在任何地方访问
    - D. 没有修饰符（默认）的成员只能在本类中访问

17. 下列关于构造方法的叙述中,错误的是（ ）。

    - A.Java语言规定构造方法名与类名必须相同
    - B. Java语言规定构造方法没有返回值,但不用void声明
    - C.Java语言规定构造方法不可以重载
    - D.Java语言规定构造方法只能通过new自动调用

18. 若有如下类定义：

```
class Person {

    private String name;

    public Person(String n) {
        name = n;
    }
}
```

    以下创建对象的语句正确的是（ ）。

    - A. Person p = new Person;
    - B. Person p = new Person("John");
    - C. Person p; p = "John";
    - D. Person p = new String("John");

19. 下列（ ）不是面向对象程序设计的基本特征。

    - A.封装
    - B.继承
    - C.多态
    - D.过程调用

20. 下列那条语句定义了3个元素的数组？（ ）。

- A)int[] a={20,30,40};
- B) int a[]=new int(3);
- C) int[3] array;
- D) int[] arr;

21. Java中定义数组名为xyz，下面哪项可以得到数组元素的个数？（）。

- A)xyz.length()
- B) xyz.length
- C) len(xyz)
- D)ubound(xyz)

22. 数组中可以包含什么类型的元素（）。

- A) int型
- B) string型
- C) 数组
- D)以上都可以

23. 下列语句序列执行后，i的值是（）。

```
int  i=10;
do {
    i/=2;
} while( i>1 );
```

- A) 1
- B) 5
- C) 2
- D) 0

24. 以下由do-while语句构成的循环执行的次数是（）。

```
int  m = 8;
do {
    ++m;
} while ( m < 8 );
```

- A)一次也不执行
- B)执行1次
- C)8次
- D)有语法错，不能执行

25. 若有循环：

```
int x=5,y=20;
do {
```

```
    y-=x;
    x+=2;
} while(x<y);
```

则循环体将被执行（）。

- A) 2次
- B) 1次
- C) 0次
- D) 3次

26. 下列语句序列执行后，j 的值是（）。

```
int  j=8, i=6;
while( i >4 ) {
    i-=2;
    --j;
}
```

- A) 5
- B) 6
- C) 7
- D) 8

27. 下列语句序列执行后，i 的值是（）。

```
int s=1,i=1;
while( i<=4 ) {
    s*=i;
    i++;
}
```

- A) 6
- B) 4
- C) 24
- D) 5

28. 以下由 for 语句构成的循环执行的次数是（）。

```
for ( int  i = 0; true ; i++) ;
```

- A)有语法错，不能执行
- B)无限次
- C)执行1次
- D)一次也不执行

29. 下列语句序列执行后，j 的值是（ ）。

```
int  j=2;
for( int i=7; i>0; i-=2 ){
    j*=2;
}
```

- A) 15
- B) 1
- C) 60
- D) 32

30. 下列语句序列执行后，j 的值是（ ）。

```
int  j=1;
for( int i=5; i>0; i-=2 )
{
    j*=i;
}
```

- A) 15
- B) 1
- C) 60
- D) 0

31. 设int 型变量 a、b，float 型变量 x、y，char 型变量 ch 均已正确定义并赋值，正确的switch语句是（ ）。

- A) switch (x + y) { ..... }
- B) switch (ch + 1) { ..... }
- C) switch ch { ..... }
- D) switch (a + b); { ..... }

32. 若a和b均是整型变量并已正确赋值，正确的switch语句是（ ）。

- A) switch(a+b); { ..... }
- B) switch(a+b*3.0) { ..... }
- C) switch a { ..... }
- D) switch (a%b) { ..... }

33. 语句byte b=011;System.out.println(b);的输出结果为（ ）。

- A. B
- B. 11
- C.9
- D.011

34. 下列运算结果为float类型值的是（ ）。

- A.100/10
- B. 100*10
- C.100.0+10
- D. 100-10

35. 下面那条语句不能定义一个float类型变量f？（）。

- A. float f=3.1415E10;
- B. float f=3.14f;
- C. float f=3.1415F;
- D. float f=3.14F;

36. 下面哪个是对字符串s1的不正确定义？（）。

- A. String s1=”abcd”;
- B.String s1;
- C. String s1=”abcd\0”;
- D. String s1=”\abcd”;

37. 下面哪个语句能定义一个字符变量char？（）。

- A. char chr=’abcd’;
- B. char chr=’\uabcd’;
- C. char chr=”abcd”;
- D.char chr=\uabcd;

38. 在Java语言中，整形常量不可以是（）。

- A. double
- B. long
- C. int
- D.byte

39. 下面哪个是Java语言中正确的标识符？（）。

- A. byte
- B. new
- C. next
- D.rest-1

40. 下面哪个单词是Java语言的关键字。（）。

- A. double
- B. this
- C. String
- D.bool

41. 多线程编程的主要目的是什么？（）。

- A. 提高程序的安全性
- B. 提高程序的可读性

- C. 提高程序的执行效率
- D. 减少程序的内存使用

42. 下列说法错误的是（ ）。

- A 线程就是程序
- B 线程是一个程序的单个执行流
- C 多线程是指一个程序的多个执行流
- D 多线程用于实现并发

43. 如果一个线程被中断，会发生什么？（ ）。

- A.线程会立即停止
- B.线程会抛出一个异常
- C.线程会忽略中断请求
- D.线程会记录中断状态，但不会做任何事情

44. 线程在声明周期中要经历5种基本状态，包括新建、（ ）、运行、阻塞和死亡。

- A 准备
- B 休眠
- C 就绪
- D 等待

45. 在Java中，哪个类是所有线程类的父类？（ ）。

- A. Object
- B. Runnable
- C. Thread
- D. Process

填空题

1. 内部类（ ）(可以/不可以)直接访问外部类的私有成员变量。

2. 下列语句序列执行后，k 的值是（ ）。

```
int  x=6, y=10, k=5;
switch( x%y )
{
    case 0:  k=x*y;
    case 6:  k=x/y;
    case 12:  k=x-y;
    default:  k=x*y-x;
}
```

3. 下列语句序列执行后，k 的值是（ ）。

```
int  i=10, j=18, k=30;
switch( j - i )
{
    case  8 :  k++;
    case  9 :  k+=2;
    case  10:  k+=3;
    default :  k/=j;
}
```

4. 下列语句序列执行后，k 的值是（ ）。

```
int i=4;
int j=5;
int k=9;
int m=5;
if(i>j||m<k){
    k++;
} else{
    k--;
}
```

5. 下列语句序列执行后，m 的值是（ ）。

```
int a=10, b=3, m=5;
if( a==b ){
    m+=a;
} else{
    m=++a*m;
}
```

6. 以下Java语句中，String str = “123456789”; str = str.subString(1,3); 执行后str中的值为（ ）。

7. 阅读程序题(给出【代码】注释标注的代码的输出结果)

```
interface Com {
    int add( int a, int b);
}

abstract class People {
    abstract int add( int a, int b);
}

class Student extends People implements Com {
    public int add(int a,int b) { return a + b;}
}
```

```
public class Main {
    public static void main(String args[ ]) {
        Student stu = new Student ();
        Com com = stu;
        int m = com.add(12,6);
        People p = stu;
        int n = p.add(12,8);
       System.out.printf("%d:%d",m,n); //【代码】
    }
}
```

8. 阅读程序题(给出【代码】注释标注的代码的输出结果)

```
class Animal {
    int m = 100;
    public int seeM() { return m;}
    public int getM() { return m;}
}

class Dog extends Animal {
    int m = 6;
    public int seeM() { return m;}
}

public class 阅读{
    public static void main (String args[ ]){
        Animal animal = new Dog();
        Dog dog = new Dog();
          System.out.printf("%d:%d:%d", dog.seeM(),
animal.getM(),animal.seeM())// 【代码】
    }
}
```

9. 阅读程序题(给出【代码】注释标注的代码的输出结果)

```
class A {
    int f(int x,int y) { return x + y;}
}

class B extends A {
    int f(int x, int y) {
        int m = super.f(x,y) + 10;
        return m;
    }
}

public class 阅读1 {
    public static void main(String args[ ]) {
        A a = new B();
```

```
            System.out.println(a.f(2,10)); //【代码】
        }
    }
```

10. 如果子类想引用父类被隐藏的成员，可以使用关键

当然，这里是你提供的填空题，去掉了答案部分：

填空题

1. 内部类 （可以/不可以）直接访问外部类的私有成员变量。

填空题

1. 内部类访问外部类的私有成员变量

```
class Outer {
    private int outerVar = 10;

    class Inner {
        void display() {
            System.out.println(outerVar); // 可以直接访问外部类的私有成员变量
        }
    }
}
```

2. switch 语句的执行结果

```
int x = 6, y = 10, k = 5;

switch (x % y) {
    case 0:
        k = x * y;
        break;
    case 6:
        k = x / y;
        break;
    case 12:
        k = x - y;
        break;
    default:
        k = x * y - x;
        break;
}

// k 的值是 54
```

3. switch 语句的执行结果

```
int i = 10, j = 18, k = 30;

switch (j - i) {
    case 8:
        k++;
        break;
    case 9:
        k += 2;
        break;
    case 10:
        k += 3;
        break;
    default:
        k /= j;
        break;
}

// k 的值是 2
```

4. if-else 语句的执行结果

```
int i = 4;
int j = 5;
int k = 9;
int m = 5;

if (i > j || m < k) {
    k++;
} else {
    k--;
}

// k 的值是 10
```

5. if-else 语句的执行结果

```
int a = 10, b = 3, m = 5;

if (a == b) {
    m += a;
} else {
    m = ++a * m;
}

// m 的值是 55
```

6. substring 方法的执行结果

```
String str = "123456789";
str = str.substring(1, 3);

// str 中的值为 23
```

7. 接口和抽象类的实现

```
interface Com {
    int add(int a, int b);
}

abstract class People {
    abstract int add(int a, int b);
}

class Student extends People implements Com {
    public int add(int a, int b) {
        return a + b;
    }
}

public class Main {
    public static void main(String args[]) {
        Student stu = new Student();
        Com com = stu;
        int m = com.add(12, 6);
        People p = stu;
        int n = p.add(12, 8);
        System.out.printf("%d:%d", m, n); // 18:20
    }
}
```

8. 继承和方法重写

```
class Animal {
    int m = 100;

    public int seeM() {
        return m;
    }

    public int getM() {
        return m;
```

```
    }
}

class Dog extends Animal {
    int m = 6;

    public int seeM() {
        return m;
    }
}

public class Test {
    public static void main(String args[]) {
        Animal animal = new Dog();
        Dog dog = new Dog();
        System.out.printf("%d:%d:%d", dog.seeM(), animal.getM(), animal.seeM());
// 6:100:6
    }
}
```

9. 方法重写和 super 关键字

```
class A {
    int f(int x, int y) {
        return x + y;
    }
}

class B extends A {
    int f(int x, int y) {
        int m = super.f(x, y) + 10;
        return m;
    }
}

public class Main {
    public static void main(String args[]) {
        A a = new B();
        System.out.println(a.f(2, 10)); // 22
    }
}
```

10. 使用 super 关键字引用父类成员

```
class Parent {
    void show() {
        System.out.println("Parent show()");
    }
```

```
}

class Child extends Parent {
    void show() {
        super.show(); // 引用父类的 show 方法
        System.out.println("Child show()");
    }
}
```

11. 使用 final 关键字定义常量

```
final int MAX_SIZE = 100;
```

12. String 类的常用方法

```
class StringTest {
    public static void main(String[] args) {
        String str = "abcdefghaijklmna";

        System.out.println("a出现的最后位置: " + str.lastIndexOf('a')); // G
        System.out.println("索引为10的字符为: " + str.charAt(10)); // J
        System.out.println("str字符串总长度: " + str.length()); // H
        System.out.println("获取从第1个a到第1个n之间(包含n)的字符序列: " +
str.substring(str.indexOf('a'), str.indexOf('n') + 1)); // E
        System.out.println("把str变化大写: " + str.toUpperCase()); // I
    }
}
```

13. String 类的常用方法

```
class StringTest {
    public static void main(String[] args) {
        String str = "abcdefghaijklmna";

        System.out.println("a出现的首位置: " + str.indexOf('a')); // D
        System.out.println("是否包含z: " + str.contains("z")); // F
        System.out.println("str字符串总长度: " + str.length()); // H
        System.out.println("str是否以abc作为开始字符串: " + str.startsWith("abc"));
// A
        str = str.replace('a', 'A'); // C
        System.out.println("已经实现把所有小写a替换为A: " + str);
    }
}
```

14. 表达式求值

```
int x = 20, y = 60;
double z = 50.0;
double result = x + (int) y / 2 * z % 10;

// x + (int) (60 / 2 * 50.0 % 10) = 20.0
```

15. 字符串拼接

```
String result = "20" + 30;

// 表达式："20" + 30 的值是 2030
```

16. 使用 final 关键字限定变量

```
final int CONSTANT = 100;
```

17. 合法的常量

```
// 合法的常量：30
```

18. 合法的标识符

```
// 不可用作用户标识符：float, 2ab
```

19. 内部类和外部类的定义和使用

```
import java.util.Scanner;

class Circle { // 外部类
    private double radius = 0;
    private static int count = 0; // 1 定义一个count变量用来存放Circle实例的个数

    public Circle(double radius) {
        this.radius = radius;
        count++; // 2 计算Circle实例的个数
    }
```

```
    class Draw { // 内部类
        public void drawShape() {
            System.out.println(radius); // 3 输出radius变量的值
            System.out.println(count); // 4 输出count变量的值
        }
    }
}

public class OuterDemo {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int r = in.nextInt(); // 圆的半径
        new Circle(r).new Draw().drawShape(); // 5 编写语句调用Draw类中的
drawShape() 方法
    }
}
```

20. 求两个列表的并集

```
import java.util.LinkedList;

public class Main {
    public static void main(String[] args) {
        LinkedList<String> list1 = new LinkedList<String>();
        LinkedList<String> list2 = new LinkedList<String>();

        list1.add("red");
        list1.add("yellow");
        list1.add("green");

        list2.add("red");
        list2.add("yellow");
        list2.add("blue");

        // list1和list2的并集
        list1.addAll(list2);

        for (int i = 0; i < list1.size(); i++) {
            System.out.print(list1.get(i));
        }
    }
}
```

21. 求两个列表的交集

```
import java.util.LinkedList;

public class Main {
```

```
    public static void main(String[] args) {
        LinkedList<String> list1 = new LinkedList<String>();
        LinkedList<String> list2 = new LinkedList<String>();

        list1.add("red");
        list1.add("yellow");
        list1.add("green");

        list2.add("red");
        list2.add("yellow");
        list2.add("blue");

        // list1和list2的交集
        list1.retainAll(list2);

        for (int i = 0; i < list1.size(); i++) {
            System.out.print(list1.get(i));
        }
    }
}
```

22. 求两个列表的差集

```
import java.util.LinkedList;

public class Main {
    public static void main(String[] args) {
        LinkedList<String> list1 = new LinkedList<String>();
        LinkedList<String> list2 = new LinkedList<String>();

        list1.add("red");
        list1.add("yellow");
        list1.add("green");

        list2.add("red");
        list2.add("yellow");
        list2.add("blue");

        // list1 - list2
        list1.removeAll(list2);

        for (int i = 0; i < list1.size(); i++) {
            System.out.print(list1.get(i));
        }
    }
}
```

23. 求两个集合的并集

```
import java.util.HashSet;

public class Main {
    public static void main(String[] args) {
        HashSet<String> set1 = new HashSet<String>();
        HashSet<String> set2 = new HashSet<String>();

        set1.add("A");
        set1.add("B");
        set1.add("C");

        set2.add("A");
        set2.add("C");

        // 两个集合的并集
        set1.addAll(set2);

        System.out.println("set1+set2: " + set1);
    }
}
```

24. 求两个集合的差集

```
import java.util.HashSet;

public class Main {
    public static void main(String[] args) {
        HashSet<String> set1 = new HashSet<String>();
        HashSet<String> set2 = new HashSet<String>();

        set1.add("A");
        set1.add("B");
        set1.add("C");

        set2.add("A");
        set2.add("C");

        // 两个集合的差
        set1.removeAll(set2);

        System.out.println("set1-set2: " + set1);
    }
}
```

25. 求两个集合的交集

```
import java.util.HashSet;
```

```
public class Test {
    public static void main(String[] args) {
        HashSet<String> set1 = new HashSet<String>();
        HashSet<String> set2 = new HashSet<String>();

        set1.add("A");
        set1.add("B");
        set1.add("C");

        set2.add("A");
        set2.add("C");

        // 两个集合的交集
        set1.retainAll(set2);

        System.out.println("set1nset2: " + set1);
    }
}
```

26. 复选框选中事件

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Winread extends JFrame {
    JCheckBox check;

    public Winread() {
        setLayout(new FlowLayout());
        check = new JCheckBox("Good");

        ItemListener l = new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                if (check.isSelected())
                    System.out.println(check.getText()); // Good
            }
        };

        check.addItemListener(l);
        add(check);
        setBounds(10, 10, 460, 360);
        setVisible(true);
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    }

    public static void main(String args[]) {
        new Winread();
    }
}
```

27. 文本框输入回车事件

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Wincomputer extends JFrame {
    JTextField text;

    public Wincomputer() {
        setLayout(new FlowLayout());
        text = new JTextField(8);

        ActionListener l = new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int m = Integer.parseInt(text.getText());
                System.out.println(m * m); // 400
            }
        };

        text.addActionListener(l);
        add(text);
        setBounds(10, 10, 460, 360);
        setVisible(true);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }

    public static void main(String args[]) {
        new Wincomputer();
    }
}
```

28. 文本区内容同步显示

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Main {
    public static void main(String args[]) {
        ComputerFrame fr = new ComputerFrame();
        fr.setTitle("计算");
    }
}

class ComputerFrame extends JFrame {
    TextArea inputNumber, showResult;

    public ComputerFrame() {
        setLayout(new FlowLayout());
```

```
        inputNumber = new TextArea(6, 20);
        showResult = new TextArea(6, 20);

        add(inputNumber);
        add(showResult);

        showResult.setEditable(false);

        inputNumber.addTextListener(new TextListener() {
            public void textValueChanged(TextEvent e) {
                showResult.setText(inputNumber.getText());
            }
        });

        setSize(300, 320);
        setVisible(true);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        validate();
    }
}
```

29. 计算器窗口

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Main {
    public static void main(String args[]) {
        ComputerFrame fr = new ComputerFrame();
        fr.setTitle("加减乘除建议运算器");
    }
}

class ComputerFrame extends JFrame {
    JTextField inputNumber1, inputNumber2, showResult;
    Button buttonMultiAdd, buttonMultiSub, buttonMulti, buttonMultiDiv;
    JLabel showOperator;

    public ComputerFrame() {
        setLayout(new FlowLayout());

        inputNumber1 = new JTextField(10);
        inputNumber2 = new JTextField(10);
        showResult = new JTextField(10);
        showOperator = new JLabel(" ", showOperator.CENTER);

        showOperator.setBackground(Color.green);

        add(inputNumber1);
        add(showOperator);
```

```
        add(inputNumber2);
        add(showResult);

        buttonMultiAdd = new Button("加");
        buttonMultiSub = new Button("减");
        buttonMulti = new Button("乘");
        buttonMultiDiv = new Button("除");

        add(buttonMultiAdd);
        add(buttonMultiSub);
        add(buttonMulti);
        add(buttonMultiDiv);

        buttonMultiAdd.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                double n1, n2, n;
                try {
                    n1 = Double.parseDouble(inputNumber1.getText());
                    n2 = Double.parseDouble(inputNumber2.getText());
                    n = n1 + n2; // 加法运算
                    showResult.setText(String.valueOf(n));
                    showOperator.setText("+");
                } catch (NumberFormatException ee) {
                    showResult.setText("请输入数字字符");
                }
            }
        });

        buttonMultiSub.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                double n1, n2, n;
                try {
                    n1 = Double.parseDouble(inputNumber1.getText());
                    n2 = Double.parseDouble(inputNumber2.getText());
                    n = n1 - n2; // 减法运算
                    showResult.setText(String.valueOf(n));
                    showOperator.setText("-");
                } catch (NumberFormatException ee) {
                    showResult.setText("请输入数字字符");
                }
            }
        });

        buttonMulti.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                double n1, n2, n;
                try {
                    n1 = Double.parseDouble(inputNumber1.getText());
                    n2 = Double.parseDouble(inputNumber2.getText());
                    n = n1 * n2; // 乘法运算
                    showResult.setText(String.valueOf(n));
                    showOperator.setText("*");
                } catch (NumberFormatException ee) {
                    showResult.setText("请输入数字字符");
```

```
                }
            }
        });

        buttonMultiDiv.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                double n1, n2, n;
                try {
                    n1 = Double.parseDouble(inputNumber1.getText());
                    n2 = Double.parseDouble(inputNumber2.getText());
                    n = n1 / n2; // 除法运算
                    showResult.setText(String.valueOf(n));
                    showOperator.setText("/");
                } catch (NumberFormatException ee) {
                    showResult.setText("请输入数字字符");
                }
            }
        });

        setSize(300, 320);
        setVisible(true);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        validate();
    }
}
```

填空题

30. 调用plus()方法求1+2+3的值并显示

```
public class Main {
    public static void main(String[] args) {
        MyPlus p = new MyPlus();
        System.out.println("1+2=" + p.plus(1, 2));
        // 调用plus()方法求1+2+3的值并显示,输出显示：1+2+3=6
        System.out.println("1+2+3=" + p.plus(1, 2, 3));
        // 调用plus()方法求1+2+3+4的值并显示,输出显示：1+2+3+4=10
        System.out.println("1+2+3+4=" + p.plus(1, 2, 3, 4));
    }
}

class MyPlus {
    public int plus(int a, int b) {
        int s;
        s = a + b; // 将a和b的和存入s中
        return s;
    }

    public int plus(int a, int b, int c) {
        int s;
        // 要求调用两个参数的plus()方法，得到a+b+c的和
```

```
        s = plus(a, b) + c;
        return s;
    }

    public int plus(int a, int b, int c, int d) {
        int s;
        // 要求调用三个参数的plus()方法，得到a+b+c+d的和
        s = plus(a, b, c) + d;
        return s;
    }
}
```

31. 定义静态成员和非静态成员

```
class Ca {
    int a;
    static int b; // 定义一个静态的整型属性b

    public static void m1() {
        b = 10; // 把属性b赋值为10
    }

    public void m2() {
        a = 5;
        b = 5;
    }

    public static void m3() {
        Ca.m1(); // 调用m1()方法
    }
}

public class Cb {
    public static void main(String[] args) {
        Ca s1 = new Ca();
        Ca s2 = new Ca();

        Ca.b = 5;
        s1.b = 6;
        s2.b = 7;

        System.out.println("s1.b=" + s1.b);
        System.out.println("s2.b=" + s2.b); // 显示s2.b的值，输出格式：s2.b=7
        System.out.println("Ca.b=" + Ca.b); // 显示Ca.b的值，输出格式：Ca.b=7
    }
}
```

32. 学习类的继承

```
class Person extends Object { // 继承自所有类的父类
    private String name; // 定义一个成员变量name

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}

class Student extends Person { // 继承Person类
    private String department;

    public void setDepartment(String department) { // 传递一个形参
        this.department = department;
    }

    public String getDepartment() {
        return department; // 返回department值
    }
}
```

33. 学习类的私有成员的定义及访问方法

```
class Person {
    String name;
    private int age; // 定义一个私有整型属性age

    public boolean setAge(int newAge) {
        if (newAge >= 5 && newAge <= 20) {
            age = newAge; // 将newAge赋值给age
            return true;
        } else {
            return false;
        }
    }

    public int getAge() {
        return age; // 返回age属性
    }
}

public class App {
    public static void main(String[] args) {
        Person s1 = new Person(); // 定义一个Person类的对象s1
        s1.name = "张三";
        if (s1.setAge(25)) { // 将s1的age属性设置为25
            System.out.println("我是" + s1.name + "今年" + s1.getAge() + "岁");
```

```
        } else {
            System.out.println("年龄错误");
        }
    }
}
```

34. 学习类的成员变量和成员方法的声明格式

```
public class Student {
    String name;
    int age;
    int chinese; // 定义一个整型属性chinese
    int math; // 定义一个整型属性math
    int english; // 定义一个整型属性english

    int total() {
        return chinese + math + english; // 返回chinese、math以及english三个整型属性
的总和
    }

    int average() {
        return (chinese + math + english) / 3; // 返回chinese、math以及english三个
整型属性的平均值
    }
}
```

35. 定义类的成员变量和方法

```
public class Tank {
    private double speed; // 声明double型变量speed,刻画速度
    private int bulletAmount; // 声明int型变量bulletAmount,刻画炮弹数量

    void speedUp(int s) {
        speed = s + speed; // 将s+speed赋值给speed
    }

    void speedDown(int d) {
        if (speed - d >= 0)
            speed = speed - d; // 将speed-d赋值给speed
        else
            speed = 0;
    }

    void setBulletAmount(int m) {
        bulletAmount = m;
    }

    int getBulletAmount() {
```

```
        return bulletAmount;
    }

    double getSpeed() {
        return speed;
    }

    void fire() {
        if (bulletAmount >= 1) {
            bulletAmount = bulletAmount - 1; // 将bulletAmount-1赋值给bulletAmount
            System.out.println("打出一发炮弹");
        } else {
            System.out.println("没有炮弹了,无法开火");
        }
    }
}

public class Fight {
    public static void main(String args[]) {
        Tank tank1, tank2;
        tank1 = new Tank();
        tank2 = new Tank();

        tank1.setBulletAmount(10);
        tank2.setBulletAmount(10);

        System.out.println("tank1的炮弹数量: " + tank1.getBulletAmount());
        System.out.println("tank2的炮弹数量: " + tank2.getBulletAmount());

        tank1.speedUp(80);
        tank2.speedUp(90);

        System.out.println("tank1目前的速度: " + tank1.getSpeed());
        System.out.println("tank2目前的速度: " + tank2.getSpeed());

        tank1.speedDown(15);
        tank2.speedDown(30);

        System.out.println("tank1目前的速度: " + tank1.getSpeed());
        System.out.println("tank2目前的速度: " + tank2.getSpeed());

        System.out.println("tank1开火: ");
        tank1.fire();

        System.out.println("tank2开火: ");
        tank2.fire();
        tank2.fire();

        System.out.println("tank1的炮弹数量: " + tank1.getBulletAmount());
        System.out.println("tank2的炮弹数量: " + tank2.getBulletAmount());
    }
}
```