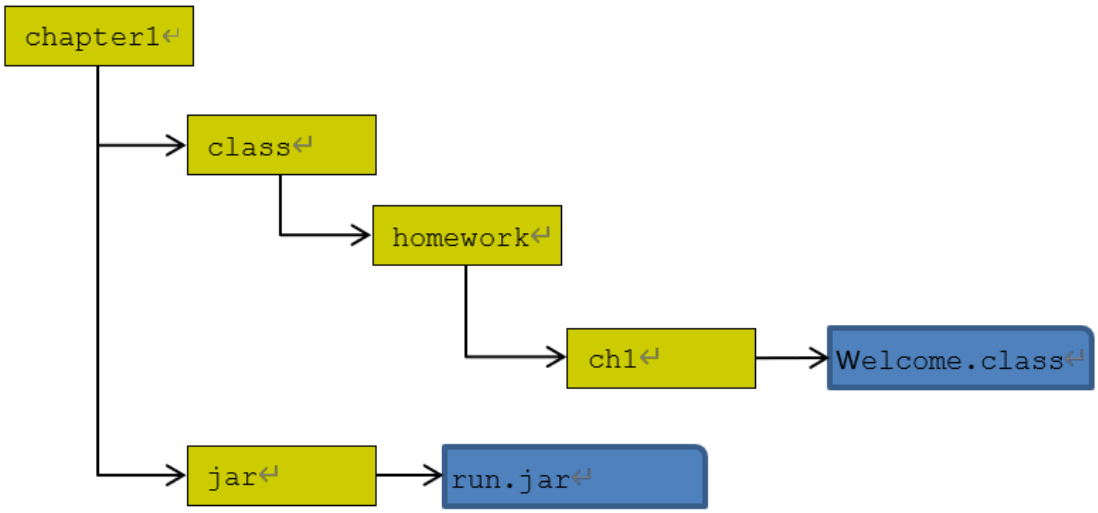
1. 给定编译好的Java程序，**请在控制台运行**下面编译好的类：

（1） Welcome类，该文件位于package homework.ch1里

（2） 运行run.jar里面的Welcome类，该文件位于package homework.ch1里。

其中，Welcome类和run.jar所在的相对目录结构下图所示：



（3）要求提交控制台运行的脚本文件

```
set JAVA_HOME=D:\codeLanguage\jdk-13.0.2
set PROJECT_HOME=D:\garfield\study\大二下\java\LearnJava\homework\第1章作业
\chapter1
set path=%path%;%JAVA_HOME%\bin
set classpath=%classpath%;%PROJECT_HOME%\jar\run.jar

java -classpath %classpath% homework.ch1.Welcome
```

一、填空

1. Java语言中有4种基本的整数类型，哪种类型所占的内存空间最小，写出定义该类型的关键字byte。
2. Java语言中有4种基本的整数类型，哪种类型所占的内存空间最大，写出定义该类型的关键字long。
3. Java中存在一种基本的数据类型，该类型定义的变量不能与其他类型转换，定义该类型用boolean。
4. 布尔型定义的成员变量是有默认值的，它的值是 false。
5. 定义变量保存含有小数的数据时，使用 double 定义的变量精度比较高。
6. Java源程序经过编译后生成被称为 字节码 的特殊机器语言码，然后经过 JVM（java虚拟机） 解释运行。
7. 声明一个值为3.14的double型常量PI的语句为 `final double PI = 3.14;`。
8. 下列程序中，首先声明和初始化三个变量ch = ‘a’，变量d=0.1，变量 l =12L，并将该三个变量打印输出.请将程序补充完整.

```
public class Assign{
    public static void main(String args[]){
            ____char___ ch = ‘a’;
            ___double____ d = 0.1;
            ____long___ l = 12L;
            System.out.println("ch=" + __ch__);
            System.out.println("d=" + __d__);
            System.out.println("l=" + __l__);
    }
}
```

9. 执行下列语句后，变量x，y的值分别为 x = 0, y =1 ，原因是 判断A&&B时，如果A为假，则B不去进行判断。

```
int x = 0, y = 0;
System.out.println( ((x > 1) && (++x == 0)) + " " + x);
System.out.println( ((y < 1) | (y++ == 0)) + " " + y);
```

10. 阅读下面代码

    以上代码错误的地方是 switch内的变量不支持long类型，

    应该为 `int i = new Scanner(System.in).nextInt();`。

```
System.out.print("Please input your choice[1,2]:");
 long i = new Scanner(System.in).nextLong();
 switch (i){
   case 1 :
     System.out.println("Your choice is 1");
     break;
   case 2 :
     System.out.println("Your choice is 2");
     break;
   default:
     System.out.println("Wrong choice");
 }
```

知识点总结：

- Java 4种基本的整数类型：`byte, short, int, long`，占用空间分别为1字节，2字节，4字节，8字节
- Java 2种基本浮点数类型：`float, double`，占用空间分别为4字节，8字节，默认值为`0.0f, 0.0d`
- Java 1种布尔类型：`boolean`，占用空间为1字节或者4字节，默认值为false
- Java 1种基本字符类型：`char`，占用空间为2字节
- Java 中用final关键字声明常量
- switch 语句中开关变量不能为long

二、选择

1．以下说法正确的是（A）。

（A）Java中所有的方法都必须在类内定义

（B） Java中主方法可以不在类内定义，其他方法都必须定义在类内

（C） Java中主方法必须定义在类内，其他方法可以不必定义在类内

（D） Java中所有方法都不必在类内定义

2. Java源文件和编译后的文件扩展名分别是（C）。

（A）.class 和 .java　　（B）.class 和 .class

（C）.java 和.class　　（D）.java 和 .java

3. 关于布尔类型说法正确的是 （A）。

（A）boolean表示布尔类型，它的取值只有true和false

（B）bool表示布尔类型，它的取值只有true和false

（C）boolean表示布尔类型，它的取值只有1和0

（D）bool表示布尔类型，它的取值只有1和0

4. 下面表达式错误的是（B）。

(A) int i = 100;

(B) float f = 100.0;

(C) Object o = “Hello world”;

(D) char c = ‘\u1234’;

5. 下面代码的输出是（A）。

```
    public static void main (String [] args) {
        int x = 1;
        System.out.print((x > 1) & (x++ > 1));
        System.out.print(" ");
        System.out.print((x > 1) && (x++ > 1));
    }
```

(A) false true

(B) true false

(C) false false

(D) true true

6. 下列叙述中正确的是（A）。

(A) Java语言的标识符是区分大小写的

(B) Java源程序文件名可以任意命名

(C) Java源程序文件的扩展名为.jar

(D) 一个Java源程序文件里public类的数目不限

7. 下列标识符中，合法的是（A）。

(A) _name (B) 4Person

(C) public (D) -3.1415

三、编程题

```java
public static int addAllDigits(int target){
            int res = 0;        // result int
            String target2 = "";
            target2 = target2 + target;
            char[] targetcopy = target2.toCharArray();
            for (int i = 0; i<targetcopy.length; i++){
                res+= ((int)targetcopy[i] - (int)'0');
            }
            return res;
}
```

一、 填空题

1：假设

String s1 = "Welcome to Java";

String s2 = s1;

String s3 = new String("Welcome to Java");

那么下面表达式的结果是什么?

(1) s1 == s2　　true

(2) s1 == s3　　false

(3) s1.equals(s2)　　true

(4) s2.equals(s3)　　true

(5) s1.compareTo(s2);　0

(6) s2.compareTo(s3);　0

(7) s1.charAt(0);　　W

(8) s1.indexOf('j');　　-1

(9) s1.indexOf("to");　8

(10) s1.lastIndexOf("o",15)　9

(11) s1.substring(3, 11);　come to

(12) s1.endsWith("Java")　true

(13) s1.startsWith("wel"); false

(14) "　We come ".trim(); We come

(15) s1.toUpperCase();　WELCOME TO JAVA

(16) s1.replace('o', 'T');　WelcTme tT Java

2．如果

StringBuffer s1 = new StringBuffer("Java");

StringBuffer s2 = new StringBuffer("HTML");

假设下列每个语句是独立的，每条语句结束后，写出相应结果

(1) s1.append(" is fun");　s1为___Java is fun___

(2) s1.append(s2);　　s1为___JavaHTML___

(3) s1.insert(2, "is fun");　s1为___Jais funva_

(4) s1.insert(1,s2);　　s1为___JHTMLava_____

(5) char c = s1.charAt(2);　c为__**v**____

(6) int i = s1.length(); i为__**4**___

(7) s1.deleteCharAt(3); s1为___Jav_____

(8) s1.delete(1,3); s1为___Ja_____

(9) s1.reverse(); s1为__**avaJ**___

(10) s1.replace(1,3, "Computer"); s1为 _ JComputera__

(11) String s3 = s1.substring(1,3);

s3为__***av***___，s1为____Java_____

(12) String s4 = s1.substring(2);

S4为____va___，s1为____Java___

3. 假设StringBuffer s = new StringBuffer("Welcome to JAVA");

将s的内容清空的语句是__ s.delete(0,s.length());

____。

4.如果

```
String s1 = "Welcome";
String s2 = new String("Welcome");
String s3 = s2.intern();
String s4 = "Wel" + "come";
String s5 = "Wel";
String s6 = "come";
String s7 = s5 + s6;
String s8 = "Wel" + new String("come");
```

那么下面表达式的结果为：

（1）s1 == s2 false____

（2）s1 == s3 true____

（3）s1 == s4 true____

（4）s1 == s7 false____

（5）s1 == s8 true____

（6）s1.equals(s2) ***true***__

（7）s1.equals(s3) true____

（8）s1.equals(s4) true____

（9）s1.equals(s7) true____

（10）s1.equals(s8) true____

二、单项选择题

1．可以获取字符串s的最后一个字符的表达式是**C**__。

(A) s.length()

(B) s[s.length() - 1]

(C) s.charAt(s.length() - 1)

(D) charAt(s, length(s))

2. 下面程序

```
class C {
    public static void main(String[] args) {
        String s = "null" ;
        if(s == null)
            System.out.print(“a”);
        else if(s.length() == 0)
            System.out.print(“b”);
        else
            System.out.print(“c”);
    }
}
```

的输出为**C**__。

(A) a　　(B) b

(C) c　　(D) null

3. 下面的程序

```
class C {
    public static void main(String[] args) {
        String s = “Welcome to ”;
        concat(s);
        System.out.print(s);
    }

    public static void concat(String s) {
        s += “Java”;
    }

}
```

的输出为 A__。

(A) Welcome to　　(B) Welcome to Java

（C）编译错误　　（D）运行时异常

三、编程题

1：编写程序，从控制台或对话框任意输入一个英文字符串，统计字符串中每个英文字母出现的次数并输出到控制台（大小写不敏感）。

```
public static void main(String[] args) {
        // 字符串输入
        String str;
        str = new Scanner(System.in).next();

        // 字符串处理
        // 1. 变成小写
        // 2. 转换为char数组进行排序
        str.toLowerCase();
        char[] str2 = str.toCharArray();
        Arrays.sort(str2);

        // 处理输出
        int count = 1;
        int i = 1;
        for (; i < str2.length; i++) {
            if (i == (str2.length - 1)) {
                if (str2[i] == str2[i - 1]) {
                    System.out.println(str2[i - 1] + " " + ++count);
                } else {
                    System.out.println(str2[i - 1] + " " + count);
                    System.out.println(str2[i] + " " + 1);
                }
            } else {
                if (str2[i] == str2[i - 1]) {
                    count++;
                } else {
                    System.out.println(str2[i - 1] + " " + count);
                    count = 1;
                }
            }
        }
    }
```

2：假设一个车牌号码由三个大写字母和后面的四个数字组成。编写一个程序. 随机生

成5个不重复的车牌号码。

```
    public static void main(String[] args) {
        String[] res = new String[]{"", "", "", "", ""};
        for (int i = 0; i < 5; i++) {
            StringBuffer str = generateLicense();
            res[i] = str.toString();
            if (i > 0) {
                for (int j = i - 1; j >= 0; j--) {
```

```
                    if (res[j].toString().equals(res[i].toString())) {
                        i--;
                        break;
                    }
                }
            }
        }

        for (int i = 0; i < 5; i++) {
            System.out.println(res[i]);
        }

    }


    public static StringBuffer generateLicense() {
        char c1 = (char) (int) (Math.random() * 26 + 65);
        char c2 = (char) (int) (Math.random() * 26 + 65);
        char c3 = (char) (int) (Math.random() * 26 + 65);
        char c4 = (char) (Math.random() * 10 + '0');
        char c5 = (char) (Math.random() * 10 + '0');
        char c6 = (char) (Math.random() * 10 + '0');
        char c7 = (char) (Math.random() * 10 + '0');

        StringBuffer str = new StringBuffer();

 str.append(c1).append(c2).append(c3).append(c4).append(c5).append(c6).append(c7
);
        return str;
    }
```

一、填空题

1. 函数重载是指在函数名相同，但 形参列表不同 不同
2. 创建大小为2行4列的二维char型数组的语句为 `char[][] arr = new char[2][4]` ，数组创建后每个元素的值为 '\u0000' 。
3. 创建一个大小为10的整型数组，且数组元素的值分别为1,2,3,4,5,6,7,8,9,10的语句为 `int[] arr = new int[]{1,2,3,4,5,6,7,8,9,10};`。
4. 用final关键字修饰一个方法形参的含义是 方法内部不允许修改该参数。
5. 下列程序存在的错误是重复定义参数，形参中已经有i了，但是for循环中又定义了一遍i___。

```
public static void m(int i){
    for(int i = 0 ; i < 10; i++){
        System.out.println(i);
    }
}
```

二、选择题

1. 下列语句的输出结果是（C）。

```
String[][] a = {
          {"Beijing","Wuhan"},
          {"Shanghai","Guangzhou","Xian"},
          {"Chongqing","Chengdu"}
};

System.out.println(a[a.length - 1].length);
System.out.println(a[a.length - 1][a[a.length - 1].length - 1].length());
```

A. 2, 5　　B. 3, 4　　C. 2, 7　　D. 3, 8

2. String[]s={“Monday”,”Tuesday”,”Wednesday”,”Thirsday”,”Friday”,”Sataday”,”Sunday”}; ，则下列语句正确的是 (C)。

```
A. int a = s.length, b = s[1].length;
B. int a = s.length( ), b = s[1].length;
C. int a = s.length, b = s[1].length();
D. int a = s.length( ), b = s[1].length( );
```

3. 若有下面程序

```
class C {
    public static void main(String[] args) {
         int[] array = new int[10];
         increase(array);
         System.out.print(array[0]);
    }
    public static void increase(int[] array) {
         for(int i = 0; i < array.length; i++) {
             array[i]++;
         }
    }

}
```

则输出为 (B)。

A. 0　　B. 1　　C. 2　　D. 10

4. 下面的数组申明和初始化语句不合法的是(C)。

```
A. int a[ ] = null;

B. int[ ] b = { };

C. int[ ] c = new int{1,2,3,4}; 改正为int[] c = new int[]{1,2,3,4};

D. int [] d[] = new int[5][ ];
```

三、判断题

1. 局部变量在使用前必须通过初始化或者赋值语句显式地给一个值。(T)
2. 一个方法必须要有一个return语句。(F)
3. 如果定义int[] nValues={1,2,3,4}; 那么nValues为引用类型。(T)
4. 不能基于函数返回类型来重载函数。 (T)
5. 二维数组的行数和列数是相同的。 (F)

四、阅读程序题

1.

```
public class Test2 {
    public static void main(String[] args){
        int[] a = {1};
        String[] s = {"Hello"};
        int i = a[0];
        m(s,a,i);
        for(String v:s){
            System.out.println(v);
        }
```

```
        for(int v:a){
            System.out.println(v);
        }
        System.out.println(i);
    }
    public static void m(String[] a1, int[] a2, int i){
        for(int j =0; j < a1.length;j++){
            a1[j] = "Java";
        }
        for(int j =0; j < a2.length;j++){
            a2[j]++;
        }
        i++;
    }
}
```

输出：

```
Java
2
1
```

五、编程题

1. 实现下面二个方法，并在Test3里添加入口main函数测试运行。

Tips：注意检查输入参数row的值，当输入负数，0时如何处理也考虑进来，如何处理这种情况不做要求，可以简单地打印出提示信息，或者抛出异常。但最简单的办法就是当出现这些边界条件，直接返回null引用就行了。由这个方法的调用者去处理。另外也不考虑当row的值太大导致内存溢出的情况。

```
public class Test3 {
    /**
     *  创建一个不规则二维数组
     *  第一行row列
     *  第二行row - 1列
     *  ...
     *  最后一行1列
     *  数组元素值都为默认值
     * @param row 行数
     * @return 创建好的不规则数组
     */

    public static int[][] createArray(int row) {
        if (row <= 0) {
            return null;
        }
        int[][] arr = new int[row][];
        for (int i = 0; i < row; i++) {
            arr[i] = new int[row - i];
        }
        return arr;
    }

    /**
     * 逐行打印出二维数组，数组元素之间以空格分开
     * @param a
```

```
     */
    public static void printArray(int[][] a) {
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a[i].length; j++) {
                System.out.print(a[i][j]);
            }
            System.out.println("");
        }
    }

}
```

一、填空题

1. 当希望一个类中的成员不能在类的外部访问时，应使用__private__修饰符定义该成员。
2. 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。
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. 在类的非静态成员函数中，使用__this__关键字来表示当前调用该函数的对象。
5. 假设A为一个类，则执行A [] array = new A[10];语句时，一共调用 **0** 次A的构造函数。

二、单项选择题

1. 下面关于构造函数的说法不正确的是_**B**。

A. 构造函数的调用时机是实例化对象时

B. 一个类必须且只能定义一个构造函数

C. 一个类可以不定义构造函数

D. 构造函数一定要和类名相同, 并且不能有返回值

2. 下列哪个修饰符可以使在一个类中定义的成员变量只能被同一包中的类访问？ B

   A．private　　B．无修饰符　　C．public　　D．protected

3. 给出下列代码，如何使成员变量m 被方法fun（）直接访问**C**。

```
class Test {
   private int m;
   public static void fun()
   {
            ...
   }
}
```

   A．将private int m 改为 protected int m
   B．将private int m 改为 public int m
   C．将private int m 改为 static int m
   D．将private int m 改为 int m

4. 对于class A，如果在另一个包中的class B中，语句 A a = new A( ); a.m=10;成立，则下列定义正确的是 **C**。

A．class A { int m; }　　B．class A {public int m; }

C．public class A{ public int m; }　　D．public class A { int m; }

5. 关于下面程序，说法正确的是 **C。**

```
class AA{
    private long i = 0;
    AA(int i){
       this.i =i;
    }
    String AA(long i) {
       this.i = i;
       return "i = " + this.i;
    }
}

public class Test_1_6 {
    public static void main(String[] args) {
       System.out.println(new AA(10).AA(20));
    }
}
```

A. 以上代码编译出错，一个类的构造函数不能有返回值；

B. 以上代码编译通过，输出结果为i = 10;

C. 以上代码编译通过，输出结果为i = 20;

D. 以上代码编译出错，一个类的构造函数不能重载；

6. 对于以下代码，说法正确的是**A**

```
package homework.ch9.p1;
public class A {
  private int i = 0;
  protected int j = 0;
}


package homework.ch9.p2;
import homework.ch9.p1.A;
public class B extends A {
  public void m() {
    new A().j = 10;
    this.j = 10;
  }
}
```

A．new A().j = 10; 有编译错误，在方法m里无法访问new A().j ;this.j无编译错误，在方法m里可以访问this.j;

B．new A().j = 10; 无编译错误，在方法m里可以访问new A().j ;this.j有编译错误，在方法m里不可以访问this.j;

C．new A().j = 10; 无编译错误，在方法m里可以访问new A().j ;this.j无编译错误，在方法m里可以访问this.j;

D．new A().j = 10; 有编译错误，在方法m里不可以访问new B().j ;this.j有编译错误，在方法m里不可以访问this.j;

三、判断对错题

1. 若a是类A的实例化对象，且a.fun();能顺利执行，则函数fun一定是实例方法。(F)
2. protected修饰的类成员只能被其子类访问。(F)
3. 类的静态变量被该类的所有实例共享。(T)
4. **Double类型的变量是值类型。(F)**
5. 当类的实例方法的形参变量与类的实例变量同名时，优先访问形参变量。(T)

四、阅读下列程序，写出输出结果：

```
public class Circle{
    private double radius;
    public static int count = 0;
    public Circle(double r){
        radius = r;
        count ++;
    }

    public Circle(){
       this(1.0);
    }

    public static void main(String[] args){
       Circle c1 = new Circle ();
       Circle c2 = new Circle (15.0);
       c1. count ++;
       c2. count ++;
      Circle. count ++;
        System.out.println("count ="+ count);
    }
}
```

输出结果为 count =5

一、填空

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 属性常量 和 抽象方法。
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 Object B A I2 I1。

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 throws 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 非必检异常 类型异常，打开一个不存在的文件会抛出 必检异常 类型异常，通过空引用调用实例方法会抛出 非必检异常 类型异常, 数组越界访问会抛出 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 必检异常 类型异常。

二、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 (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 (D)。静态方法中只能访问静态变量!!!

```
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 (A)。Base类中没有默认无参构造，必须在Derived中显示调用super(s)

```
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 (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)。ClassCastException

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 (C)。

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

        }

    }
    public void m2() {
        m1();               // 此处编译报错,未处理异常
    }
}
```

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 (A)。因为RuntimeException不是必检异常

```
public class HomeWorkTest {
    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。(F) 抽象类不一定要包含抽象方法
2. 一个接口只能继承一个直接父接口。（F） java中的接口支持多继承。类不支持
3. 非抽象类的子类不能是抽象类。（F）非抽象类的子类可以是抽象类
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。(T)
5. import语句通常出现在package语句之前。(F)
6. 抽象类中不能定义构造方法。(F) 可以定义抽象方法
7. this关键字可以在类的所有方法里使用。(F) 静态方法中不能用this
8. 类A的所有实例方法都可以在A的子类中进行覆盖(Override)。(F) 被final修饰的方法不能被覆盖
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。(F) 不能初始化实例数据成员
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。(F)

四、阅读题

阅读程序写出结果

```
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( );

    }

}
```

输出结果：

```
static in SuperClass
static in SubClass
SupuerClass is called
SubClass is called
SupuerClass is called
```

阅读程序写出指定语句输出结果，并解释原因

```
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)");
    }
}
```

① D's m(int,int)；子类D中重写（覆盖）了 参数为两个int类型的m方法，此时调用的是子类重写的方法

② C' m(double,double)；显然

③ C' m(double,double)；这里1.0f会自动转型，float转为double

④ D's m(int,int)；子类引用指向自身，在自身类中找方法，找不到再到父类中去找

⑤ C' m(double,double)；double不能自动转型为float，到D中去找方法

⑥ D's m(float,float)；显然

父类引用指向子类对象这种形式，有以下几种情况：

- 如果子类重写了父类的某个方法，那么此时调用的是子类重写的方法
- 如果子类没有重写父类的某个方法，那么此时调用的是父类中的方法
- 如果子类新增了父类中不存在的方法，那么这个父类引用是不能调用这个仅在子类中存在的方法中，因为子类对象自动向上转型为了父类对象
- 如果子类与父类有同名的成员变量和静态变量，那么由于子类自动向上转型为父类对象，那么输出的必然是父类的成员变量和静态变量
- 子类引用指向自身，子类可以继承父类的成员变量和静态变量，同时可以覆盖父类的成员变量和静态变量

阅读程序写出结果

```
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");
    }
}
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
    }
}
```

① A's m1

② C's m2

③ B's m1

④ A's m1

⑤ C's m2

静态方法不能被override。 而且调用只和引用类形有关， 也就是说你用哪个类的引用（无论heap上存的真实类型是什么），你就是在调用哪个类的static方法，所以多态对静态方法是不适用的。

一、填空

1. 语句Class clz = null；的含义是 clz是Class类型的空引用，没有指向任何Class类型的对象 。

2. 给定下列类的定义：

```
class GeometricObject {}
class PolyGon extends GeometricObject {}
class Rectangle extends PolyGon {}
GeometricObject o = new Rectangle ();
Class clz1 = o. getClass ();
```

(1) 声明一个指向PolyGon及其子类的类型信息的引用变量clz的语句应该是

`Class< ? Extends PolyGon> clz;` ;

(2) System.out.println(o.getClass().getSimpleName());的输出结果是 Rectangle;

(3) 下列语句中有错误的是____ ②___③_____;

```
Class<PolyGon> clz3 = null;
clz3 = PolyGon.class;              ①
clz3 = Rectangle.class;            ②
Class<? extends PolyGon> clz4 = null;
clz4 = GeometricObject.class;      ③
clz4 = PolyGon.class;              ④
clz4 = Rectangle.class;             ⑤
```

错误原因是（按错误题号解释）第二个语句中clz3被泛型化为PolyGon，因此只能指向PolyGon类型信息，不能指向非PolyGon类型信息；第三个语句中clz4被泛型化为PolyGon类型以及他的子类，因此可以指向PolyGon类型以及它的子类，但不能指向父类 。。

3. 下面五条语句中，错误的有 (2) (3)。

```
（1）ArrayList<String> lists = new ArrayList<String>();
（2）ArrayList<Object> lists = new ArrayList<String>();
（3）ArrayList<String> lists = new ArrayList<Object>();
（4）ArrayList<String> lists = new ArrayList();
（5）ArrayList lists = new ArrayList<String>();
```

错误原因是 前后参数泛型类型不统一。

使用泛型通配符?将错误的语句修改正确的方法是

```
ArrayList<? extends Object> lists = new ArrayList<String>();
ArrayList<? super String> lists = new ArrayList<Object>();
```

4. 下面代码给出了泛型类和非泛型类的定义：

```
class Holder<T> {
  T value;
  public Holder (T value) {this.value = value;}
  public T getValue () {return value;}
}

class RawHolder {
  Object value;
  public RawHolder (Object value) {this.value = value;}
  public Object getValue () {return value;}
}
```

基于上面二个类的定义，有下面四段代码：

```
①
Holder<String> h1 = new Holder<>("aaa");
String s1 = h1. getValue ();
System.out.println(s1);

②
RawHolder h1 = new RawHolder("aaa");
String s1 = (String)h1. getValue ();
System.out.println(s1);

③
Holder<String> h1 = new Holder<> (Integer.valueOf(111));
String s1 = h1. getValue ();
System.out.println(s1);

④
RawHolder h1 = new RawHolder (Integer.valueOf(111));
String s1 = (String)h1. getValue ();
System.out.println(s1);
```

上面四段代码中编译通过运行不出错的是_____①___②____，

上面四段代码中编译通过运行出错的是④，原因是 变量类型不匹配，Integer类不能强制转换乘String类，

上面四段代码中编译不通过是③，原因是 无法推断参数类型，泛型不匹配，

这个例子说明泛型的作用是 类型检查以及不用强制类型转换。

二、单项选择题

1. 泛型参数代表的是（D）。

A. 任意类型

B. 某类型的子类型

C. 某类型的父类型

D. 固定指代某种类型

2. 泛型通配符<?>代表的是（A）。

A. 任意类型

B. 某类型的子类型

C. 某类型的父类型

D. 固定指代某种类型

3. 下面泛型定义中不正确的是 （D）。

A. class Test1 {}

B. interface Test2 {}

C. class Test3{ void test () {}}

D. class Test4{void test () {}}

4. 泛型通配符<? extends T>代表的是（B）。

A. 任意类型

B. 某类型T的子类型

C. 某类型T的父类型

D. 固定指代某种类型

5. 泛型通配符<? super T>代表的是（C）。

A. 任意类型

B. 某类型T的子类型

C. 某类型T的父类型

D. 固定指代某种类型

6. 关于下面代码，描述正确的是（C）。

```
List<String> list = new ArrayList<String>();
list.add("test");
list.add("red");
list.add (100);
System.out.println(list. size ());
```

A. 输出2

B. 输出3

C. 编译错误

D. 运行时报异常

7. 关于下面代码，描述正确的是（B）。

```
List<Integer> ex_int= new ArrayList<Integer> ();
List<Number> ex_num = ex_int;
System.out.println(ex_num. size ());
```

A. 0

B. 编译错误

C. 运行时报异常

D. 1

8. 下列语句编译时不出错的是（D）。无法将任何元素（`null`除外）放入`List<?>`中。

A. List<?> c1 = new ArrayList (); c1.add (new Object ());

B. List<?> c2 = new ArrayList (); c2.add (new String ("1"));

C. List<?> c3 = new ArrayList (); c3.add ("1");

D. List<?> c4 = new ArrayList (); c4.add(null);

9. 给定下列代码：

```
class Shape {
}

class Circle extends Shape {
}

class Triangle extends Shape {
}

class Test2_9 {
    public static void main(String[] args) {
        List<? extends Shape> list1 = new ArrayList<Triangle>();
        List<? extends Shape> list2 = new ArrayList<Circle>();
        System.out.println(list1 instanceof List<Triangle>);      ①
        System.out.println(list2 instanceof List);                ②
        System.out.println(list1.getClass() == list2.getClass()); ③
    }
}
```

则关于语句①②③说法正确的是：_D__。

A. ①②③输出结果为true、false、false

B. ①②③输出结果为true、true、true

C. ①编译出错，②③输出结果为false、false

D. ①编译出错，②③输出结果为true、true

三、多项选择题（一个或多个正确选项）

1. 对于泛型类class A { … }，T在A类里可以用作不同的地方，在A类类体内，下面语句正确的有 *ABDG*。

A. T x;

B. T m1() {return null;}

C. static T y;

D. void m2(T i) {}

E. static T s1() {return null;}

F. static void s2(T i) {}

G. static void s3(T1 i, T1 j){}

使用泛型时，类型参数不允许为静态(`static`)。由于静态变量在对象之间共享，因此编译器无法确定要使用的类型

2. 下列语句编译时不出错的是AEGH_____。

```
A.    List<? super Integer> x1 = new ArrayList<Number> ();
B.    List<? super Number> x2 = new ArrayList<Integer> ();
C.    List<? super Number> x3 = new ArrayList<Short> ();
D.    List<? super Integer> x4 = new ArrayList<Short> ();
E.    List<? extends Number> x5 = new ArrayList<Integer> ();
F.    List<? extends Number> x6 = new ArrayList<Object> ();
G.    List<Number> x7 = new ArrayList<> ();
H.    List<? extends Comparable<Double>> x8 = new ArrayList<Double> ();
I.    List<? extends Number> x9 = new ArrayList<int> ();
```

3. 下面泛型类是List<?>的子类的是__ABC____。

A. List

B. List

C. List

D. List

4. 泛型参数应该写在的位置是BD_____。

A. 类名前

B. 类名后

C. 方法名前

D. 方法返回值类型前

5. 关于java泛型，下面描述正确的是___ABCD_____。

A. 泛型的类型参数只能是类类型（包括自定义类），不能是基本类型

B. 泛型的类型参数可以有多个

C. 不能对泛型的具体实例类型使用instanceof操作，如 o instanceof ArrayList，否则编译时会出错。

D. 不能创建一个泛型的具体实例类型的数组，如 new ArrayList[10]，否则编译时会出错。

6. 给定下列类和泛型方法的定义：

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}
public class Test2_9{
    public static <T> void m (List<? super T> list1, List<? extends T> list2) {}
}
```

则下面6段代码编译出错的是___CEF___。

A.

List **l1 = new ArrayList<> ();**

List l2 = new ArrayList<> ();

Test2_9.m (l1, l2);

B.

List l3 = new ArrayList<> ();

List l4 = new ArrayList<> ();

Test2_9.m (l3, l4);

C.

List l5 = new ArrayList<> ();

List l6 = new ArrayList<> ();

Test2_9.m (l5, l6);

D.

List l7 = new ArrayList<> ();

List l8 = new ArrayList<> ();

Test2_9.m (l7, l8);

E.

List l7 = new ArrayList<> ();

List l8 = new ArrayList<> ();

Test2_9. m (l7, l8);

F.

List l9 = new ArrayList<> ();

List l10 = new ArrayList<> ();

Test2_9.m (l9, l10);

阅读下列程序，并填写表格

```
import java.util.*;
class A {}
class B extends A {}
class Test {
   public static void m1(List<? extends A> list) {}
   public static void m2(List<A> list) {}
   public static void m3(List<? super A> list) { }
   public static void main (String [] args) {
     List<A> listA = new ArrayList<A> ();
     List<B> listB = new ArrayList<B> ();
     List<Object> listO = new ArrayList<Object> ();
// insert code here
  }

}
```

在上面代码插入点插入的代码	结果（从下面结果选项中选择）
m1(listA);	C
m2(listA);	C
m3(listA);	C
m1(listB);	C
m2(listB);	A
m3(listB);	A
m1(listO);	A
m2(listO);	A
m3(listO);	C
结果选项	
A. 编译出错	
B. 编译正确，运行出错	
C.编译正确，运行正确	

一、填空题

1. 创建线程的方式有 实现Runnable接口 和 继承Thread类 。
2. 程序中可能出现一种情况：多个线种互相等待对方持有的锁，而在得到对方的锁之前都不会释放自己的锁，这就是 死锁 。
3. 若在线程的执行代码中调用yield方法后，则该线程将 从running变为ready，允许其他线程执行（自己也可能立即执行） 。
4. 线程程序可以调用 *sleep* ***方法，使线程进入睡眠状态，可以通过调用*** *setPriority* 方法设置线程的优先级。
5. 获得当前线程id的语句是 Thread.currentThread().getId() 。

二、单项选择题

1. 能够是线程进入死亡状态的是_C_。

A. 调用Thread类的yield方法

B. 调用Thread类的sleep方法

C. 线程任务的run方法结束

D. 线程死锁

2. 给定下列程序：

```
public class Holder {
    private int data = 0;
    public int getData () {return data;}
    public synchronized void inc (int amount) {
       int newValue = data + amount;
       try {Thread.sleep(5);
       } catch (InterruptedException e) {}
       data = newValue;
    }

    public void dec (int amount) {
       int newValue = data - amount;
       try {Thread.sleep(1);
       } catch (InterruptedException e) {}
       data = newValue;
    }
}

public static void main (String [] args) {
    ExecutorService es = Executors.newCachedThreadPool();
    Holder holder = new Holder ();
    int incAmount = 10, decAmount = 5, loops = 100;
    Runnable incTask = () -> holder.inc(incAmount);
    Runnable decTask = () -> holder.dec(decAmount);
```

```
    for (int i = 0; i < loops; i++) {
       es. execute(incTask);
       es. execute(decTask);
    }
    es. shutdown ();
    while (! es. isTerminated ()) {}
}
```

下列说法正确的是B。

A. 当一个线程进入holder对象的inc方法后，holder对象被锁住，因此其他线程不能进入inc方法和dec方法

B. 当一个线程进入holder对象的inc方法后，holder对象被锁住，因此其他线程不能进入inc方法，但可以进入dec方法

C. 当一个线程进入holder对象的dec方法后，holder对象被锁住，因此其他线程不能进入dec方法和inc方法

D. 当一个线程进入holder对象的dec方法后，holder对象被锁住，因此其他线程不能进入dec方法，但可以进入inc方法

3. 给定下列程序：

```
class Test2_3 {

    private static Object lockObject = new Object();
    /**
     * \* 计数器
     */
    public static class Counter {
        private int count = 0;
        public int getCount() {
            return count;
        }

        public void inc() {
            synchronized (lockObject) {
                int temp = count + 1;
                try {
                    Thread.sleep(5);
                } catch (InterruptedException e) {
                }
                count = temp;
            }
        }

        public void dec() {
            synchronized (lockObject) {
                int temp = count - 1;
                try {
                    Thread.sleep(5);
                } catch (InterruptedException e) {
                }
                count = temp;
            }
```

```
        }
    }


    public static void main(String[] args) {
        ExecutorService es = Executors.newCachedThreadPool();
        Counter counter1 = new Counter();
        Counter counter2 = new Counter();
        int loops1 = 10, loops2 = 5;
        Runnable incTask = () -> counter1.inc();
        Runnable decTask = () -> counter2.dec();
        for (int i = 0; i < loops1; i++) {
            es.execute(incTask);
        }
        for (int i = 0; i < loops2; i++) {
            es.execute(decTask);
        }
        es.shutdown();
        while (!es.isTerminated()) {
        }
    }
}
```

下面说法正确的是_C__。

A. incTask的执行线程进入counter1对象的inc方法后，counter1对象被上锁，会阻塞decTask的执行线程进入counter2对象的dec方法

B. incTask的执行线程进入counter1对象的inc方法后，counter1对象被上锁，不会阻塞decTask的执行线程进入counter2对象的dec方法

C. incTask的执行线程进入对象counter1的inc方法后，lockObject对象被上锁，会阻塞decTask执行线程进入counter2对象的方法dec

D. incTask的执行线程进入对象counter1的inc方法后，lockObject对象被上锁，不会阻塞decTask执行线程进入counter2对象的方法dec

4. 给定下列程序：

```
class Test2_4 {
    public static class Resource {
        private int value = 0;

        public int sum(int amount) {
            int newValue = value + amount;
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
            }
            return newValue;
        }

        public int sub(int amount) {
            int newValue = value - amount;
            try {
```

```
                Thread.sleep(5);
            } catch (InterruptedException e) {
            }
            return newValue;
        }
    }


    public static void main(String[] args) {
        ExecutorService es = Executors.newCachedThreadPool();
        Resource r = new Resource();
        int loops1 = 10, loops2 = 5, amount = 5;
        Runnable sumTask = () -> r.sum(amount);
        Runnable subTask = () -> r.sub(amount);
        for (int i = 0; i < loops1; i++) {
            es.execute(sumTask);
        }
        for (int i = 0; i < loops2; i++) {
            es.execute(subTask);
        }
        es.shutdown();
        while (!es.isTerminated()) {
        }
    }
}
```

下面说法正确的是_C__。

A. 由于方法sum和sub都没有采取任何同步措施，所以sumTask和subTask的执行线程都可以同时进入共享资源对象r的sum方法或sub方法，造成对象r的实例成员value的值不一致；

B. 由于方法sum和sub都没有采取任何同步措施，所以sumTask和subTask的执行线程都可以同时进入共享资源对象r的sum方法或sub方法，造成方法内局部变量newValue和形参amount的值不一致；

C. 虽然方法sum和sub都没有采取任何同步措施，但Resource类的sum和sub里的局部变量newValue和形参amount位于每个线程各自的堆栈里互不干扰，同时多个线程进入共享资源对象r的sum方法或sub方法后，对实例数据成员value都只有读操作，因此Resource类是线程安全的

D. 以上说法都不正确

5. 给定下列程序：

```
class Test2_5 {
    public static class Resource {
        private static int value = 0;

        public static int getValue() {
            return value;
        }

        public static void inc(int amount) {
            synchronized (Resource.class) {
                int newValue = value + amount;
                try {
                    Thread.sleep(5);
                } catch (InterruptedException e) {
```

```
                }
                value = newValue;
            }
        }

        public synchronized static void dec(int amount) {
            int newValue = value - amount;
            try {
                Thread.sleep(2);
            } catch (InterruptedException e) {
            }
            value = newValue;
        }
    }

    public static void main(String[] args) {
        ExecutorService es = Executors.newCachedThreadPool();
        int incAmount = 10, decAmount = 5, loops = 100;
        Resource r1 = new Resource();
        Resource r2 = new Resource();
        Runnable incTask = () -> r1.inc(incAmount);
        Runnable decTask = () -> r2.dec(decAmount);
        for (int i = 0; i < loops; i++) {
            es.execute(incTask);
            es.execute(decTask);
        }
        es.shutdown();
        while (!es.isTerminated()) {
        }
    }
}
```

下面说法**错误的**的是_B__。

A. 同步的静态方法public synchronized static void dec (int amount) {} 等价于public static void dec (int amount) {synchronized (Resource. class) {}}

B. incTask的执行线程访问的对象r1，decTask访问的是对象r2，由于访问的是不同对象，因此incTask的执行线程和decTask的执行线程之间不会同步

C. 虽然incTask的执行线程和decTask的执行线程访问的是Resource类不同对象r1和r2，但由于调用的是Resource类的同步静态方法，因此incTask的执行线程和decTask的执行线程之间是被同步的

D. 一个线程进入Resource类的同步静态方法后，这个类的所有静态同步方法都被上锁，而且上的是对象锁，被锁的对象是Resource.class。但是这个锁的作用范围是Resource类的所有实例，即不管线程通过Resource类的哪个实例调用静态同步方法，都将被阻塞

6. 假设一个临界区通过Lock锁进行同步控制，当一个线程拿到一个临界区的Lock锁，进入该临界区后，该临界区被上锁。这时下面的说法正确的是D___。

A. 如果在临界区里线程执行Thread.sleep方法，将导致线程进入阻塞状态，同时临界区的锁会被释放；如果在临界区里线程执行Lock锁的条件对象的await方法，将导致线程进入阻塞状态，同时临界区的锁会被释放

B.如果在临界区里线程执行Thread.sleep方法，将导致线程进入阻塞状态，同时临界区的锁不会被释放；如果在临界区里线程执行Lock锁的条件对象的await方法，将导致线程进入阻塞状态，同时临界区的锁不会被释放

C. 如果在临界区里线程执行Thread.sleep方法，将导致线程进入阻塞状态，同时临界区的锁会被释放；如果在临界区里线程执行Lock锁的条件对象的await方法，将导致线程进入阻塞状态，同时临界区的锁不会被释放

D. 如果在临界区里线程执行Thread.sleep方法，将导致线程进入阻塞状态，同时临界区的锁不会被释放；如果在临界区里线程执行Lock锁的条件对象的await方法，将导致线程进入阻塞状态，同时临界区的锁会被释放

三、问答题

1：有三个线程T1，T2，T3，怎么确保它们按指定顺序执行：首先执行T1，T1结束后执行T2，T2结束后执行T3，T3结束后主线程才结束。请给出示意代码。

```
    public static void main(String[] args) {
        Thread t1 = new T1();
        Thread t2 = new T2();
        Thread t3 = new T3();

        try {
            t1.start();
            t1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        try {
            t2.start();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        try {
            t3.start();
            t3.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
```