网络应用程序设计暨实验总复习

初识 Java

1. JVM、JRE、JDK 分别为何？功能为何？三者关系为何？

   JVM (Java Virtual Machine 虚拟机) 在内存中开辟一块空间将 Java 源文件编译成字节码文件。Java 源文件的后缀名为 ".java" (文本文档即可打开)，编译后的字节码文件后缀名为 ".class"。

   JRE (Java Runtime Environment) ，编译后的字节码文件必须在运行环境中执行。JRE 提供运行别人写好的 java 程序的环境。

   JDK (Java Development Kit，开发工具包)，开发时需要用到的工具。 javac.exe 编 译 工 具 、java.exe 执行工具、…等工具执行档均在底层 DOS 命令窗口中执行。

   三者关系为 JVM ⊂ JRE ⊂ JDK

2 Java 语言有哪些特性(至少说明 5 项)

   1) 简单性

      Java 语言是由 C 和 C++ 语言转变而来的，Java 语言对 C++ 进行了简化和提高。例如，使用接口取代多重继承，取消指针；通过实现垃圾自动收集、异常处理等，大大简化了程序设计人员的工作。

   2) 面向对象

      面向对象编程思想是 Java 语言的重要特性，它本身就是一种完全面向对象的程序设计语言，Java 语言所有的元素都要通过类和对象来访问。

   3) 跨平台

      早期跨平台指的是相容于各厂家的硬件芯片，现在则指的是跨各种操作系统。

   4) 健壮性(鲁棒性)

      Java 的设计目标之一，是编写可靠的应用程序，Java 检查程序在编译和运行时的错误。类型检查帮助用户检查出许多在开发早期出现的错误。

      集成开发工具(如 Eclipse、NetBeans、myEclips、IDEA、……)的出现使编译和运行 Java 程序更加容易。

   5) 多线程

      多线程机制能够使应用程序在同一时间并行执行多项任务，带来更好的交互能力和实时行为，提高了执行性能，但降低了安全性，故线程安全是重要课题。

   6) 大数据开发相关

      Java 为 Hadoop、Spark 大数据框架平台底层的基础。

3. 如何运用文本记事本与命令行窗口编写、编译和执行一段 Java 源代码

   1) 源代码是一堆英文按照某种规则编写而成，以文本文档形式存储在计算机的硬盘上，源代码以后缀名 .java 结尾。

   2) 文件打开里面的内容是一堆按照规律写的英文，先写一个关键字 class (类)，类后面起一个名字(理论上是可以随便写的)，命名规则和规约：

      规则：名字中可以含有如下的信息；

      字母 (区分大小写，52 个)；

      数字 (0-9，但不能用作开头)；

符号 (英文符号，只能用 “_”, “$”)；

中文 (强烈不推荐)；

规约：类名字，首字母大写，如果两个以上的单词，每个单词的首字母都大写，例如 TestOne；

起名字要见名知义，建议尽量使用英文单词。

3) 类名称后紧跟着一对 { } 大括号；

4) 利用 JDK 包中提供的 javac.exe 编译工具进行代码的编译以及 java.exe 执行工具执行。

Java 语言基础

1. Java 有哪些基本数据类型？在内存空间内需配置多少比特(bit)空间？

   整数类型

   byte (8 bit)

   short (16 bit)

   int (32 bit)

   long (64 bit)

   浮点数类型

   float (32 bit)

   double (64 bit)

   字符类型

   Char (16 bit)

   布尔类型(又称逻辑类型)

   Boolean (1 bit)

2. public class Demo{

   public static void main(String[] args){

   byte x;

   x = 1;

   }

   }

   1) 上述 Java 代码编译执行，计算机底层内存运作原理为何？

   2) x 是什么？存在哪儿？

   3) 1 是什么？存在哪儿？

   4) = 是什么？做了什么？

   1) byte x; //声明一个变量空间，空间的名字是 x，空间内要求存储的数据类型是 byte 整数

   x = 1; //将一个 1 赋值到 x 空间内进行存储

   (如果画示意图会更好)

   2) x 是变量，存储在栈内存

   3) 1 为常量，存储在方法区的常量缓冲区(常量池)

   4) = 为赋值运算符，将 = 右边的值赋给 = 左边的空间存储

3. Java 语言提供了哪几种添加注释的语法？

   Java 语言提供了 3 种添加注释的方法，分别为

   单行注释：“//”为单行注释标记，从符号“//”开始直到换行为止的所有内容均作为注释而被编译器忽略。

   多行注释：“/* */”为多行注释标记，符号“/*”与“*/”之间的所有内容均为注释内容。注释中的内容可以换行。

   文档注释：“/** */”为文档注释标记。符号“/**”与“*/”之间的内容均为文档注释内容。当文档注释出现在声明 (如类的声明、类的成员变量的声明、类的成员方法声明等) 之前时，会被 Javadoc 文档工具读取作为 Javadoc 文档内容。

4.
```
public class Demo{
    public static void main(String[] args){
        byte a = 1;
        short b = 2;
        int c = 3;
        long d = 4;
        float e = 5.6F;
        double f = 7.8;
        char g = 'a';
        boolean h = true;
        String i = "abc";
    }
}
```
上述代码中，哪些运行时系统会进行自动转换？

byte a = 1;

short b = 2;

long d = 4;

5.
```
int a = 1;
int b = 1;
for(int i = 1; i <=100; i++){
        a = a++;
        b = ++b;
}
```
上列程序运行后 a = ？ b =？ 为什么？

a = 1, b = 101

x 在做值交换的时候，会产生一个临时的副本空间(备份)。
++在变量的前面则先自增后备份；
++在变量的后面则先备份后自增；
最后会由副本空间内的值赋给对应的变量。

6.
```
int m = 1;
int n = 2;
int num = m++ + ++n - n-- - --m + n-- - --m;
```
上列程序运行后

m = ？ n = ？ num = ？

num =	m++	+	++n	-	n--	-	--m	+	n--	-	--m
	1(m=2)		(n=3)3		3(n=2)		(m=1)1		2(n=1)		(m=0)0

m==0, n==1, num==2

注：()内为内存空间的值，()外是备份空间的值

7. Java 语法中，= 和 == 区别与功能何在？

= 为赋值运算符号，将 = 后面的结果(值 or 引用)存入 = 左边的变量空间内。

== 为比较运算符号，比较 == 前面和 == 后面的元素(值 or 引用)是否一致。

8. 何谓 条件与(短路与) &&，它和 逻辑与 & 有何区别？

所谓短路与，指的是当 && 前面的比较表达式结果为 false 时，会发生短路；如果发生了短路情况，&& 之后的所有比较表达式的计算则不再处理，性能会比逻辑与好。

& 逻辑与和 && 短路与，从执行的最终结果来看没有任何区别。短路与不一定能提高运行的

性能，只有当 && 前面为 false 的时候才会发生短路，性能才会提高。

9. 何谓 条件或(短路或)||，它和 逻辑或 | 有何区别？

   所谓短路或，指的是当 || 前面的比较表达式结果为 true 时，会发生短路；如果发生了短路情况，|| 之后的所有比较表达式的计算则不再处理，性能会比逻辑或好。

   | 逻辑或和 || 短路或，从执行的最终结果来看没有任何区别。短路或不一定能提高运行的性能，只有当 || 前面为 true 的时候才会发生短路，性能才会提高。

10. 请说明 & 和 && 的区别。

    & 可以为逻辑运算符，也可以为位运算符；&& 只能当作逻辑运算符使用。

    如果两个符号都当作逻辑运算符来使用时候，如下区别：

    (1) & 前后两个条件都是 true，最终结果就是 true；

    (2) 条件与(短路与) && 正常情况下和逻辑与 & 执行结果是一致；当前面条件为 false 的时候，发生短路则不再处理后面的条件比较，直接给定最终结果 false。

11. 请说明 | 和 || 的区别。

    | 可以为逻辑运算符，也可以为位运算符；|| 只能当作逻辑运算符使用。

    如果两个符号都当作逻辑运算符来使用时候，如下区别：

    (1) | 前后两个条件都是 false，最终结果就是 false；

    (2) 条件或(短路或) || 正常情况下和逻辑或 | 执行结果是一致；当前面条件为 true 的时候，发生短路则不再处理后面的条件比较，直接给定最终结果 true。

12. 请以最有效率的方式计算 2 * 8 的结果。

    以 2 * 8 而言，运用位运算最有效率，2 的二进制左位移 3 位(即 2 << 3)相当于 2 乘以 2 的 3 次幂，结果为 16。

13. int a = 1;

    int b = 2;

    如何将两个变量的值进行互换？

```
//方式一：采用一个中间变量空间
int a = 1;
int b = 2;
int c = a;
a = b;
b = c;
//优点是容易理解，值不会越界；
//缺点在于需要额外的内存空间。
```

```
方式二
int a = 1;
int b = 2;
a = a + b; //a空间存储的是两个元素之和，b没有变化
b = a - b; //利用两个元素和减原来的b，剩下是原来的a，赋值给b
a = a - b; //利用两个元素和减原来的a，剩下原来的b，赋值给a
//优点是不需额外内存空间；
//缺点在于，较不容易理解，加法运算可能会产生值的越界。
```

```
方式三：采用一个数按位异或同一个数两次，其值不会改变
int a = 1;
int b = 2;
a = a ^ b; //a=异或中间值 (即 1^2)
b = a ^ b; //b=异或中间值按位异或b (即 1^2^2)==1
a = a ^ b; //a=异或中间值按位异或b (即 1^2^1)==2
//优点是不需额外内存空间，值不会越界；缺点是不容易理解。
```

Java 语法结构(流程控制)

1. 编写应用程序代码时，应确保程序代码具备哪些特性，以养成良好编写代码习惯。

   1) 可读性

      命名规范、规约

      代码缩进

      多添加注释

   2) 健壮性

      程序严谨

   3) 优化

      实现功能的基础上，能不能做优化？

      (代码结构简单、减少冗余、提升性能、减少内存占用空间等因素考量)

2. 以下程序为利用 if 语句实现一个判断给定月份对应的季节，3, 4, 5 月 春天，6, 7, 8 月 夏天，9, 10, 11 月 秋天，12, 1, 2 月 冬天。

```
import java.util.Scanner;
public class SeasonDemo{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        System.out.println("请输入一个月份");
        int month = input.nextInt();
        if(month>=3 && month<=5){
            System.out.println("春天");
        }else if(month>=6 && month<=8){
            System.out.println("夏天");
        }else if(month>=9 && month<=11){
            System.out.println("秋天");
        }else if(month==12 || (month>=1 && month<=2)){
            System.out.println("冬天");
        }else{
            System.out.println("您输入的数据有误！");
        }
    }
}
```

```
if(month<1 || month>12){
    System.out.println("您输入的数据有误！");
}else if(month>=3 && month<=5){
    System.out.println("春天");
}else if(month>=6 && month<=8){
    System.out.println("夏天");
}else if(month>=9 && month<=11){
    System.out.println("秋天");
}else{
    System.out.println("冬天");
}
```

请问该程序是否有优化的可能性？如果可以优化请优化之。(框出待优化区域，并直接在代码空白区域作答)

3. 运用 switch 语句实现一个判断学生成绩对应的区间，<60 不及格，60-69 及格，70-79 中，80-89 良，90-99 优秀，100 满分，除此之外，数据有误。

```
import java.util.Scanner;
public class SwitchDemo{
    public static void main(String[] args){
        //1.创建一个变量 score 用来存储学生成绩
        Scanner input = new Scanner(System.in);
        System.out.println("请输入一个学生成绩,我来帮您判断区间");
        int score = input.nextInt();
        //2.利用成绩的值 来进行判断所在区间
        switch(score/10){
            case 0:
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
```

```
                System.out.println("<60 不及格");
                break;
            case 6:
                System.out.println("60-69  及格");
                break;
            case 7:
                System.out.println("70-79  中");
                break;
            case 8:
                System.out.println("80-89  良");
                break;
            case 9:
                System.out.println("90-99  优秀");
                break;
            case 10:
                if(score==100){
                    System.out.println("100  满分");
                    break;
                }
            default:
                System.out.println("输入数据有误");
                break;
        }
    }
}
```

4. 运用 switch 语句实现：利用 Scanner 输入一个值(代表星期几)，为小明同学制定一个学习计划。1, 3, 5 学习编程语言，2, 4, 6 学习英文，7 休息。

```
import java.util.Scanner;
public class SwitchDemo{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        System.out.println("请您输入一个星期，我来帮您指定学习计划");
        int day = input.nextInt();
        switch(day){
            case 1:
            case 3:
            case 5:
                System.out.println("应该学习编程");
                break;
            case 2:
            case 4:
            case 6:
                System.out.println("应该学习英文");
                break;
            case 7:
                System.out.println("休息");
                break;
            default:
                System.out.println("输入数据有误");
                break;
        }
    }
}
```

5. 运用 for 循环语句实现：操场上有 100 多人，让他们排队。三个人一组多 1 个，4 个人一组多 2 个，5 个人一组多 2 个。求解操场上的人数为多少？(请先描述问题解决的思路)

问题解决思路
设操场的人数 x，其中 x 满足 x%3==1, x%4==2, x%5==2。
在 100～200 之间，挨个的尝试一下，哪一个数字符合上述问题的条件。

```
public class ForDemo{
    public static void main(String[] args){
        for( int x = 100 ; x < 200 ; x++ ){
            if( x%3 == 1 && x%4 == 2 && x%5 == 2 ){
                System.out.println("操场上的人数为: "+x);
            }
        }
    }
```

```
}
```

6. 运用 for 循环语句实现：甲、乙、丙、丁 4 个人加工零件，加工的总零件数为 370 个，如果甲加工的零件数多 10，如果乙加工的零件数少 20，如果丙加工的零件数乘以 2，如果丁加工的零件数除以 2，则 4 个人加工的零件数就相等啦，求 4 个人加工的零件个数分别是多少？(请先描述问题解决的思路，可以不考虑优化)

```
public class ForDemo{
    public static void main(String[] args){
        for( int x = 1 ; x < 370 ; x++){
            if((x-10) + (x+20) + (x/2) + (x*2) == 370){
                System.out.println("甲加工的零件数为:"+(x-10));
                System.out.println("乙加工的零件数为:"+(x+20));
                System.out.println("丙加工的零件数为:"+(x/2));
                System.out.println("丁加工的零件数为:"+(x*2));
            }
        }
    }
}
```

问题解决思路
设 4 个人相等时的数为 x。
则 (x-10) + (x+20) + (x/2) + (x*2) == 370。
在 1～370 之间，挨个的尝试一下，哪一个数字符合上述问题的条件。

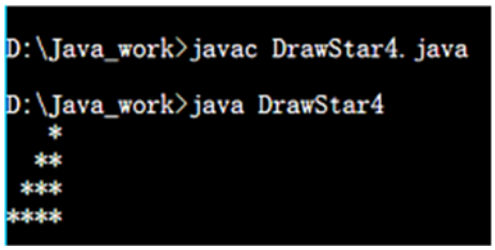
7. 运用 for 循环语句实现：小鸡、小兔子关在同一个笼子里，小鸡两只脚，小兔子四只脚，小鸡和小兔子总数 50 只，脚的总数 160 只，求小鸡和小兔子各多少只？(请先描述问题解决的思路)

```
public class ForDemo{
    public static void main(String[] args){
        for( int x = 1 ; x < 50 ; x++){
            if( x*2 + (50-x)*4 == 160){
                System.out.println("小鸡的个数为: "+x);
                System.out.println("小兔子个数为: "+(50-x));
            }
        }
    }
}
```

问题解决思路
设小鸡 x 只，则小兔子 (50-x) 只。
x*2 + (50-x)*4 == 160。
给计算机一个范围，挨个尝试某一个 x 满足上述的条件。

8. 运用 for 循环嵌套实现控制台输出 4 行星星，每一行星星个数和该行的行数相同，如右图编译执行的结果。(请在代码中注解代码含义)

```
public class DrawStar4{
    public static void main(String[] args){
        for( int i = 1 ; i <= 4 ; i++){//控制四行
            //画占位符
            for( int j = 1 ; j <= 4-i ; j++){
                System.out.print(" ");
            }
            //画星星
            for( int j = 1 ; j <= i ; j++){
                System.out.print("*");
            }
            System.out.println(); //换行
        }
    }
}
```



9. 写出下列代码编译执行后的结果，空格请以方格□表示。

```
public class NestingDemo {
    public static void main(String[] args){
        for( int i = 1 ; i <= 4 ; i++){
            if(i==1){
                for( int j = 1 ; j <= 7 ; j++){
                    System.out.print("*");
                }
            else{
                for( int j = 1 ; j <= 5-i ; j++){
                    System.out.print("*");
                }
```

```
                for( int j = 1 ; j <= 2*i-3 ; j++){
                    System.out.print(" ");
                }
                for( int j =1 ; j <= 5-i ; j++){
                    System.out.print("*");
                }
            }
            System.out.println();
        }
    }
}
```

10. 运用 for 循环嵌套实现如右图编译执行的结果。(请在代码中注解代码含义)

```
D:\Java_work>javac NestingDemo1.java

D:\Java_work>java NestingDemo1
*******
*** ***
**   **
*     *
```

```
public class NestingDemo1 {
    public static void main(String[] args){
        for( int i = 1 ; i <= 4 ; i++){//控制行数
            if(i==1){//第一行规则
                //画星星
                for( int j = 1 ; j <= 7 ; j++){ // 7 颗星
                    System.out.print("*");
                }
                else{ //后 3 行规则
                    //画星星
                    for( int j = 1 ; j <= 5-i ; j++){ // 5-i 颗星
                        System.out.print("*");
                }
                //画空格
                for( int j = 1 ; j <= 2*i-3 ; j++){ // 2i-3 个空格
                    System.out.print(" ");
                }
                //画星星
                for( int j =1 ; j <= 5-i ; j++){
                    System.out.print("*");
                }
            }
            //换行
            System.out.println();
        }
    }
}
```

11. 下列代码编译执行后的结果是什么？

```
public class TestBreak {
    public static void main(String[] args){
        int i = 1;
        int j = 1;
        for( ; i <= 5 ; i++){
            for( ; j <= 5 ; j++){
                if( j == 3){
                    break;
                }
                System.out.println("Java 应用开发");
            }
        }
        System.out.println( "i = " + i );
        System.out.println( "j = " + j );
    }
}
```

```
Java 应用开发
Java 应用开发
i = 6
j = 3
```

12. 下列代码编译执行后的果是什么？

```
public class TestBreak {
    public static void main(String[] args){
        int i = 1;
        int j = 1;
        tag_i: for ( ; i <= 5 ; i++){ //
            tag_j: for ( ; j <= 5 ; j++){
                if( j == 3 ){
                    break tag_i;
                }
                System.out.println("Java 应用开发");
            }
```

```
Java 应用开发
Java 应用开发
i = 1
j = 3
```

```
        }
        System.out.println( "i = " + i );
        System.out.println( "j = " + j );
    }
}
```

13. 下列代码编译执行后的结果是什么？

```
public class TestContinue {
    public static void main(String[] args){
        int i = 1;
        int j = 1;
        for( ; i <= 5 ; i++){
            for( ; j <= 5 ; j++){
                if( j == 3){
                    continue;
                }
                System.out.println("Java 应用开发");
            }
        }
        System.out.println( "i = " + i );
        System.out.println( "j = " + j );
    }
}
```

```
Java 应用开发
Java 应用开发
Java 应用开发
Java 应用开发
i = 6
j = 6
```

14. 下列代码编译执行后的结果是什么？

```
public class TestContinue {
    public static void main(String[] args){
        int i = 1;
        int j = 1;
        tag_i: for ( ; i <= 5 ; i++){
            tag_j: for ( j = 1 ; j <= 5 ; j++){
                if( j == 3 ){
                    continue tag_i;
                }
                System.out.println("Java 应用开发");
            }
        }
        System.out.println( "i = " + i );
        System.out.println( "j = " + j );
    }
}
```

```
Java 应用开发
Java 应用开发
Java 应用开发
Java 应用开发
Java 应用开发
Java 应用开发
Java 应用开发
Java 应用开发
Java 应用开发
Java 应用开发
i = 6
j = 3
```

15. 下列代码编译执行后的结果是什么？

```
public class TestWhile {
    public static void main(String[] args){
        int i = 1;
        while( i <= 5 ){
            System.out.println( "执行 ing: " + i );
            i++;
        }
        System.out.println("执行完毕: " + i );
    }
}
```

```
执行 ing: 1
执行 ing: 2
执行 ing: 3
执行 ing: 4
执行 ing: 5
执行完毕: 6
```

16. 下列代码编译执行后的结果是什么？

```
public class TestDoWhile {
    public static void main(String[] args){
        int i = 1;
        do {
            System.out.println("执行 ing: " + i );
            i++;
        } while ( i <= 5 );
        System.out.println("执行完毕: " + i );
    }
}
```

```
执行 ing: 1
执行 ing: 2
执行 ing: 3
执行 ing: 4
执行 ing: 5
执行完毕: 6
```

17. 有一个水池，已经盛满了 120 立方米的水。有一个进水管，每小时进水 18 立方米；有一个排水管，小时排水 30 立方米；两个水管一起开，经过多少小时，水池的水排放干净？(请在代码中注解代码含义)

```
public class TestWhile{
    public static void main(String[] args){
        int sum = 120; //表示水池中的水
        int hour = 0; //  记录经过小时数
        while( sum > 0 ) {
            sum += 18 ;
            sum -= 30 ;
            hour++; //记录一次小时数
            System.out.println("本次循环完毕: " + sum );
        }
        System.out.println( "经过  " + hour + "  小时排水完毕" );
    }
}
```

Java 数组的使用

1. 下列 Java 数组的特点，何者有误？(多选)
   (A) 数组本身是一个引用数据类型，以 [] 表示；
   (B) 数组内部存储的元素为基本类型；
   (C) 数组元素是存储在堆内存中，如果需要用变量来进行存储，
   变量空间是存储在栈内存中，且变量中存储的是数组引用；
   (D) 数组的长度随时可以透过静态或动态初始化予以改变。

2. 下列 Java 数组的声明，何者有误？
   (A) int[] array;
   (B) []int array;
   (C) int []array;
   (D) int array[];

3. 下列 Java 数组的初始化，何者有误？
   (A) int[] array = new int[] { 10,20,30,40,50 };
   (B) int[] array = new int[5];
   (C) int[] array = new int[5] { 10,20,30,40,50 };
   (D) int[] array = { 10,20,30,40,50 };

4. int[] array = { 1, 2, 3, 4, 5 };
   int x = array[5];
   请问上述代码经过编译、运行后，其结果是下列何者情况？
   (A) 编译时报错，ArrayIndexOutOfBoundsException；
   (B) 编译时报错，ArraySizeOutOfBoundsException；
   (C) 运行时报错，ArrayIndexOutOfBoundsException；
   (D) 运行时报错，ArraySizeOutOfBoundsException。

5. int[] array = { 1, 2, 3, 4, 5 };
   int x = array[-5];
   请问上述代码经过编译、运行后，其结果是下列何者情况？
   (A) 编译时报错，NegativeArraySizeException；
   (B) 编译时报错，ArrayIndexOutOfBoundsException；
   (C) 运行时报错，NegativeArraySizeException；
   (D) 运行时报错，ArrayIndexOutOfBoundsException。

6. int[] array = int[-5];
   请问上述代码经过编译、运行后，其结果是下列何者情况？
   (A) 编译时报错，ArrayIndexOutOfBoundsException；
   (B) 运行时报错，ArrayIndexOutOfBoundsException；
   (C) 编译时报错，NegativeArraySizeException；
   (D) 运行时报错，NegativeArraySizeException。

7. 请说明数组的概念、特点、语法结构、初始化方式、元素访问(存取)方式、遍历(轮询)方式、以

及内存如何存储数组对象。

1)概念：数组是用来存储一组相同数据类型的数据(可以理解是一个容器)，将一组数据统一的管理起来。

2)特点：数组本身是一个引用数据类型，以 [ ] 表示。

数组内部存储的元素，可以是基本类型，也可以是引用类型。

数组的长度一旦确定，是不能再次发生改变。

3)语法结构(创建、声明)：数组数据类型[ ] 数组名字;

4)初始化(创建数组并赋值)：

静态初始化(有长度，有元素)

int[] array = new int[] {1,2,3,4,5};

动态初始化(只有长度，没给定元素值(但有默认值))

int[] array = new int[长度];

如果长度<0，运行时报错，NegativeArraySizeException(数组长度不合法，长度为负数)。

5)元素访问(存/取)：通过元素在数组中的位置(index 索引)访问。

数组的索引是从 0 开始到 (长度-1) 为止；如果索引出现前述范围以外的值，运行时报错，ArrayIndexOutOfBoundsException(数组索引越界)。

6)遍历轮询：

正常 for 循环，有索引、可以赋值、取值，写法相对来说比较麻烦。

增强 for 循环，写法相对简单，没有索引、只能取值。

7)数组本身是一个引用数据类型：

数组元素是存储在堆内存中一串具有相同数据类型且为连续地址的空间。

数组如果需要用变量来进行存储，变量空间在栈内存中，变量中存储的是数组引用(地址)。

8. 正常 for 循环以及增强 for 循环在语法结构上有何差别？用於数组遍历有何优缺点？

1)正常的 for 循环语法为 for(初始值; 终点判定条件; 变化量){ }

遍历时通过 index 索引，找到某一个元素的位置。

优点：可以通过 index 直接访问数组的某一个位置，可以存值/取值；

缺点：不好在於写法相对来说比较麻烦。

2)增强的 for 循环为 JDK1.5 版本后新的特性，有两个条件(取值的变量、遍历的数组)，没有 index 索引，语法为 for (自己定义的变量: 遍历的数组) { }。

优点：写法相对比较容易；

缺点：只能遍历取值，不能遍历存值。

9. 下列代码编译执行后的结果是什么？

```
public class TestArray{
    public static void main(String[ ] args){
        int[ ] array = new int[ ]{ 10,20,30,40,50 };
        for( int index = 0 ; index < 5 ; index++){
            int value = array[ index ];
            System.out.println( value );
        }
        System.out.println("-----");
        for( int value : array){
            value = 100;
            System.out.println( value );
        }
```

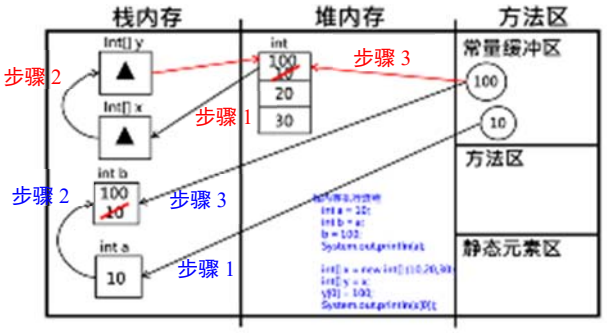
```
10
20
30
40
50
-----
100
100
100
100
100
```

```
        }
    }
```

10. 下列两组代码经过编译、运行后，a == ？ 和 x[0] == ？，并画出该两组代码在 JVM 内存运作的情形，说明基本数据类型和引用数据类型在内存结构上的区别。

int a = 10;	int[] x = new int[] { 10, 20, 30 };
int b = a;	int[] y = x;
b = 100;	y[0] = 100;

a == 10, x[0] == 100



所有的变量空间都存储在栈内存，变量空间可以存储基本数据类型，也可以存储引用数据类型。

如果变量空间存储的是基本数据类型，存储的是值，一个变量的值改变，另一个不会跟着改变。

如果变量空间存储的是引用数据类型，存储的是引用(地址)，一个变量地址对应的值改变，另一个跟着改变。

11. 给定两个数组 a{1,2,3,4}、b{5,6,7,8}，运用 for 循环语句交换数组中对应的元素，将两个数组内的元素对应位置互换。(请在代码中注解代码含义)

```
public class ArrayApp1{
    public static void main(String[] args){
        //1.创建两个数组
        int[] a = {1,2,3,4};
        int[] b = {5,6,7,8};
        //2.元素对应位置的互换(交换数组中对应的元素)
        for( int i = 0 ; i < a.length ; i++){ //为了控制四次
                int x = a[ i ];
                a[ i ] = b[ i ];
                b[ i ] = x;
        }
        //3.分别输出两个数组元素看一看
        for( int value : a){
                System.out.println( value );
        }
        System.out.println("--------");
        for( int value : b){
                System.out.println( value );
        }
    }
}
```

12. 给定两个数组 a{1,2,3,4}、b{5,6,7,8}，运用直接交换变量 a, b 的数组引用(地址)，将两个数组内的元素对应位置互换。(请在代码中注解代码含义)

```
public class ArrayApp{
    public static void main(String[] args){
        //1.创建两个数组
        int[] a = {1,2,3,4};
        int[] b = {5,6,7,8};
        //2.元素对应位置的互换(直接交换变量 a b 的数组引用)
        int[] temp = a;
        a = b;
        b = temp;
        //3.分别输出两个数组元素看一看
        for( int value : a){
                System.out.println( value );
        }
```

```
        System.out.println("--------");
        for( int value : b){
            System.out.println( value );
        }
    }
}
```

13. 给定一个数组 a{1,2,3,4,5,6}，将这个数组中的元素头尾对应互换位置。(请在代码中注解代码含义)

```
public class ArrayApp{
    public static void main(String[] args){
        int[] array = {1,2,3,4,5,6};
        for(int i = 0 ; i < array.length/2 ; i++){//控制交换的次数，数组长度的一半
            int x = array[i];
            array[i] = array[(array.length-1) - i ];
            array[(array.length-1) – i ] = x;
        }
        for(int value : array){
            System.out.println(value);
        }
    }
}
```

14. 给定一个数组 a{1,3,5,7,9,0,2,4,6,8}，找寻数组中的最大值和最小值。(请在代码中注解代码含义)

```
public class ArrayApp{
    public static void main(String[] args){
        int[] a = {1,3,5,7,9,0,2,4,6,8};
        //1.创建两个变量，记录最大值、最小值信息
        int min = a[0];
        int max = a[0];
        //2.挨个寻找数组中的元素并与变量中的元素进行比较
        for(int i = 1 ; i < a.length ; i++){
            if( a[i] < min ){
                min = a[i];
            }
            if( a[i] > max ){
                max = a[i];
            }
        }
        System.out.println( "数组中的最小值为: " + min);
        System.out.println( "数组中的最大值为: " + max);
    }
}
```

15. 给定两个数组 a{1,2,3}、b{4,5}，合并两个数组。(请在代码中注解代码含义)

```
public class ArrayApp{
    public static void main(String[] args){
        //1.创建两个数组
        int[] a = {1,2,3};
        int[] b = {4,5};
        //2.因为数组长度一旦确定，不能再次改变，需要创建一个新的数组
        int[] newArray = new int[ a.length + b.length ]; //只有长度，元素默认值 0
        //3.分别将 a 和 b 数组中的元素存入新数组内
        for( int i = 0 ; i < a.length ; i++){ //将所有 a 数组元素存入新数组内
            newArray[ i ] = a[ i ];
        } //   newArray ---> {1,2,3,0,0}
        for( int i = 0 ; i < b.length ; i++){ //将 b 数组元素取出来，存入新数组后面位置
            newArray[ a.length + i ] = b[ i ];
        }//   newArray ---> {1,2,3,4,5}
        //4.验证一下，看一看
```

```
        for(int value : newArray){
            System.out.println(value);
        }
    }
}
```

16. 给定一个数组 a{1,2,3,9,4,5}，按照数组中的最大值位置将数组拆分成两个数组 {1,2,3}、{4,5}。(请在代码中注解代码含义)

```
public class ArrayApp{
    public static void main(String[] args){
        //1.需要一个原来的数组
        int[] oldArray = {1,2,3,9,4,5};
        //2.找寻最大值的索引位置，通过这个位置确定两个小数组的长度
        int max = oldArray[0]; //数组的第一个元素值
        int index = 0; //数组的第一个索引位置(找寻最大值过程，需要一个用来记录最大值的索引位置)
        for( int i = 1 ; i < oldArray.length ; i++){
            if( oldArray[i] > max ){
                max = oldArray[i];
                index = i;
            }
        }
        System.out.println( "最大值为: " + max );
        System.out.println( "最大值位置: " + index );
        //3.需要两个小数组分别承载元素
        int[] newa = new int[ index ];
        int[] newb = new int[ oldArray.length - index - 1 ];
        //4.分别将两个小数组填满
        for( int i = 0 ; i < newa.length ; i++){
            newa[ i ] = oldArray[ i ];
        }
        for( int i = 0 ; i < newb.length ; i++){
            newb[ i ] = oldArray[ (index + 1) + i ];
        }
        //5.分别验证两个新数组，看一看
        for( int value : newa ){
            System.out.println(value);
        }
        System.out.println("------");
        for( int value : newb ){
            System.out.println(value);
        }
    }
}
```

17. int[] array = new int[5];

数组的长度一旦确定，是不能再次发生改变。在应用上是否有办法在不增加栈内存变量空间的基础上，改变数组的长度为 10 ？

```
int[] array = new int[5];
//运用变量存储数组引用的特性
array = new int[10]; //旧数组没有任何引用，将由 GC 回收
```

18. 以下代码运行结果，下列何者正确？并于代码上圈出有错的地方。

```
Public class Foo{
    Public static void main(String[] args){
        String s;
        System.out.println(s);
    }
}
```

(A) 输出 null

(B) 输出" "

(C) 编译错误

(D) 出现异常

19. 以下代码运行结果，下列何者正确？

```
public static void main(String[] args){
    int i = 1;
    int j = 1;
    for( ; i <= 5; ){
        for( ; j <= 5; ){
            if( j == 3){
                break;
            }
            System.out.print( j++ );
        }
        i++;
    }
    System.out.print( "-" );
    System.out.print( i );
    System.out.print( j );
}
```

(A) 1212121212-63

(B) 1212121212-66

(C) 12-63

(D) 12-66

20. 以下代码运行结果，下列叙述何者正确？

```
public static void main(String[] args){
    int x = 1;
    int y = 2;
    int result =x++ - y-- - ++x - --y + ++y + x--;
    System.out.println( x );
    System.out.println( y );
    System.out.println( result );
}
```

(A) x = 2, y = 1, result = 1

(B) x = 1, y = 2, result = 0

(C) x = 1, y = 2, result = 1

(D) x = 2, y = 1, result = 0

21. 以下代码运行结果，下列何者正确？

```
public class ArrayTest{
    public static void main(String[] args){
        int f1[], f2[];
        f1 = new int[10];
        f2 = f1;
        System.out.println( "f2[0]=" + f2[0] );
    }
}
```

(A) 打印 f2[0]=0

(B) 打印 f2[0]=null

(C) 第 5 行编译错误

(D) 第 6 行出现异常

22. 下列数组声明何者正确？

(A) int ia[] = new int[15];

(B) float fa = new float[20];

(C) char d[] = new char('d');

(D) int ia[][] = {4,5,6},{1,2,3};

23.
```
int[][] array = new int[3][];
array[0][0] = 10;
System.out.printline( array[0][0] )
```
上述代码编译、执行后的结果为

(A) 0

(B) 10

(C) 编译时异常 NullPointerException

(D) 运行时异常 NullPointerException

面向对象(I)

1. 请以「相声小品-赵本山《钟点工》— 大象装冰箱，总共分几步？」为例，说明面向过程编程思想与面向对象编程思想的区别。

   面向过程的编程思想乃是解决问题的时候按照一定的过程(流程)，大段的过程，拆分成小段，不强调到底是由哪一个实体来做的，以过程为本，复用性差，易增加许多冗余。

   例：相声小品-赵本山《钟点工》— 大象装冰箱，总共分几步？

   开门 -> 装大象 -> 关门

   每一步骤均以解决当前过程目标为主，以致于开/关冰箱门、装大象和开/关仓库门、装长颈鹿的编程因目标物的不同就必须各自单独撰写，即便是程序方法相同也不好复用。

   面向对象的编程思想解决问题的时候则是按照现实生活中的规律来考虑问题。考虑在这个问题的过程中，有几个实体参与其中，并将实体予以归类。可以理解为：归类是描述类的特点和动作行为，实体才是特点和动作行为的支配者，没有实体，特点、动作行为就发生不了。

   例：相声小品-赵本山《钟点工》— 大象装冰箱，总共分几步？

   参与实体的归类：人、大象、冰箱

   分析每一类个体都有什么特点(属性)？做了哪些事情(方法)？

   人的特点：能做事情；做了哪些事？ 开/关冰箱门、装大象。

   大象的特点：体积大、体重很重；做了哪些事？ 此处不做任何事。

   冰箱的特点：有门、有容积；做了哪些事？ 存放大象。

   面向对象的编程思想运用类来描述定义属性、方法，而真正解决问题则以对象实体来执行，如此，将冰箱门和仓库门归类为「门」的类，大象和长颈鹿归类为「大型动物」的类，而开/关门和装东西为做事情的方法，程序方法相同在类中定义一次即可，复用性佳。因此，解决问题时可以针对不同的实体创建不同的对象去执行相同的方法，符合现实生活中的规律而且应用上又非常灵活。

2. 如何在计算机中创建(描述)类和对象？

   1)先创建一个类(class)；

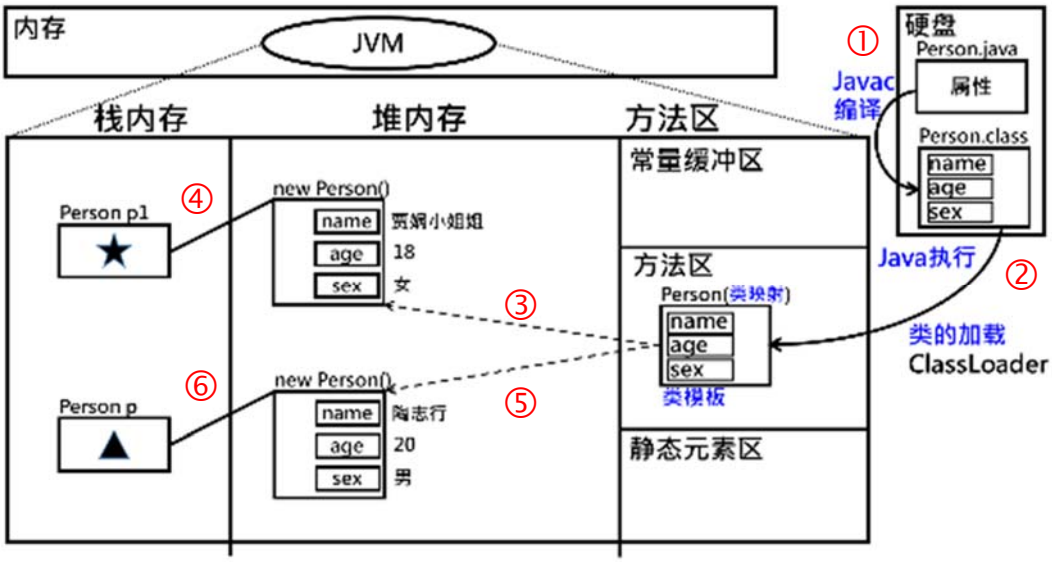
   2)类的里面，利用属性或方法去描述这个类；

   3)创建一个当前类的对象，让对象调用属性/方法做事。

3. 编译、执行下列程序代码，JVM 内存如何运行操作框中属性的存储？

```
public class Person {
    public String name;
    public int age;
    public String sex;
}
```

```
public class MainMethod {
    public static void main(String[] args){
        Person p = new Person();
        p.name = "王小明";
        p.age = 20;
        p.sex = "男";
        Person p1 = new Person();
        p1.name = "贾娲小姐姐" ;
        p1.age = 18 ;
        p1.sex = "女" ;
    }
}
```



4. 下列有关 C++和 Java 语言叙述，何者正确？(多选)

(A) C++语言为一面向过程的程序设计语言；

(B) Java 语言为一面向过程的程序设计语言；

(C) C++ 语言为一面向对象的程序设计语言；

(D) Java 语言为一面向对象的程序设计语言。

5. 下列计算机中创建类和对象的顺序，何者正确？

(A) 创建对象 → 创建类 → 调用对象属性或方法；

(B) 创建对象 → 调用对象属性或方法 → 创建类；

(C) 创建类 → 创建对象 → 调用对象属性或方法；

(D)调用对象属性或方法 → 创建对象 → 创建类。

6. 下列在类中描述属性的语法结构，何者正确？(多选)

(A) public static String name;

(B) public String name;

(C) public String name = "王小明";

(D) public name = "王小明";

7.
```
public class Person{
    public int age;
}
```

下列何者为正确创建上述 Person 类的对象？

(A) Person p = new Person{};

(B) Person p = new Person;

(C) Person p = new Person();

(D) Person p = Person();

8.
```
public class Person{
    public int age;
}
```

```
public class Main{
    public static void main(String[] args){
        Person p1 = new Person();
        p1.age = 20;
        Person p2 = p1;
        p2.age = 40;
    }
}
```

上述代码编译、运行后，最后结果，下列何者正确？(多选)

(A) p1.age == 20

(B) p2.age == 40

(C) p1.age == 40

(D) p2.age == 20

9. 下列在类中描述方法的语法结构，何者正确？(多选)

(A) public void eat(){…}

(B) public String eat(){…return "大米饭";}

(C) public void eat(){…return "大米饭";}

(D) public String eat(){…}

10.
```
public void eat(){
    System.out.println("吃了 1 碗大米饭");
}
```

创建类的对象为 eated，下列调用类中 eat() 方法的语法何者正确？

(A) eated eat();

(B) eated.eat();

(C) eated_eat();

(D) eat();

11. 类在创建对象后，调用类中的方法，让方法执行一遍，请问：

1) 方法存储在内存的哪里?

2) 方法在内存的哪里执行?

1) 方法存储在堆内存的对象空间内；

2) 方法在栈内存中开辟一块临时的方法执行空间。

12.
```
public void eat(){
    System.out.println("吃了 1 碗大米饭");
}
```

创建类的对象为 eated，下列调用类中 eat() 方法的语法，何者正确？

(A) String eating = eated.eat();

(B) eated.eat();

(C) String eating = eated.eat("大米饭");

(D) eated.eat("大米饭");

13.
```
public String eat(){
   return "1  碗大米饭";
}
```
创建类的对象为 eated，下列调用类中 eat() 方法的语法，何者正确？(多选)

(A) String eating = eated.eat();

(B) eated.eat();

(C) String eating = eated.eat("大米饭");

(D) eated.eat("大米饭");

14.
```
public void eat(String eating){
      System.out.println("吃了  1  碗" + eating);
   }
```
创建类的对象为 eated，下列调用类中 eat() 方法的语法，何者正确？

(A) String eating = eated.eat();

(B) eated.eat();

(C) String eating = eated.eat("大米饭");

(D) eated.eat("大米饭");

15.
```
public String eat(String eatWhat){
      return "吃了  1  碗" + eating;
   }
```
创建类的对象为 eated，下列调用类中 eat() 方法的语法，何者正确？

(A) String eating = eated.eat();

(B) eated.eat();

(C) String eating = eated.eat("大米饭");

(D) eated.eat("大米饭");

16.
```
public static void main(String[] args){
   int[] a = {1,3,5,7,9,0,2,4,6,8};
   boolean flag = true; //flag==true 找最大值
   //1.创建一个变量
   int value = a[ 0 ];
   //2.利用遍历数组的方式挨个与  value  比较
   for(int i = 1 ; i < a.length ; i++){
      if( flag && a[ i ] > value ){ //找寻最大值
            value = a[ i ];
      }else if( !flag && a[ i ] < value ){ //找寻最小值
            value = a[ i ];
      }
   }
   System.out.println( "数组中极值为: " + value );
}
```
请将上述面向过程的代码改写为面向对象的方法。(请在代码中注解代码含义)
```
public class Demo {
   //是否需要参数及返回值
   //需要参数  提供一个数组   提供一个值(最大  最小)
   //需要返回值   需要一个值(最大  最小)
   public int findMinOrMax( int[] array, boolean flag ){
      //创建一个变量
      int value = array[ 0 ];
      //利用遍历数组的方式挨个与  value  比较
```

```
        for( int i = 1; i < array.length; i++ ){
            if( flag && array[ i ] > value ){ //找寻最大值
                value = array[ i ];
            }else if( !flag && array[ i ] < value ){ //找寻最小值
                value = array[ i ];
            }
        }
        return value;
    }
    public static void main(String[] args){
        int[] a = {1,3,5,7,9,0,2,4,6,8};
        boolean flag = true; //flag==true 找最大值
        //创建一个 Demo 对象
        Demo d = new Demo();
        //利用 d 对象调用方法执行操作
        int minOrMax = d.findMinOrMax( a, flag );
        System.out.println( "数组中极值为: " + minOrMax );
    }
}
```

17.
```
public class Demo {
    public void changeTwoArray(int[] x, int[] y){
        for( int i = 0; i < x.length; i++ ){
        int temp = x[i];
        x[i] = y[i];
        y[i] = temp;
    }
    public static void main(String[] args){
        Demo d = new Demo();
        int[] a = {1,2,3,4};
        int[] b = {5,6,7,8};
        d.changeTwoArray(a, b);
        for(int value : a){
            System.out.println(value);
        }
        System.out.println("------");
        for(int value : b){
            System.out.println(value);
        }
    }
}
```

下列何者为上述代码执行的结果？

(A)
```
1
2
3
4
------
5
6
7
8
```

(B)
```
5
6
7
8
------
1
2
3
4
```

(C)
```
1
2
3
4
----
5
6
7
8
```

(D)
```
5
6
7
8
----
1
2
3
4
```

18.
```
public class Demo {
    public int[][] changeTwoArray(int[] x, int[] y){
        int[] temp = x;
        x = y;
        y = temp;
        int[][] result = { x, y }
        return result;
    }
    public static void main(String[] args){
        Demo d = new Demo();
```

```
        int[] a = {1,2,3,4};
        int[] b = {5,6,7,8};
        int[][] changed = d.changeTwoArray(a, b);
        a = changed[0];
        b = changed[1];
        for(int value : a){
            System.out.println(value);
        }
        System.out.println("------");
        for(int value : b){
            System.out.println(value);
        }
    }
}
```

下列何者为上述代码执行的结果？

(A)
```
1
2
3
4
------
5
6
7
8
```

(B)
```
5
6
7
8
------
1
2
3
4
```

(C)
```
1
2
3
4
----
5
6
7
8
```

(D)
```
5
6
7
8
----
1
2
3
4
```

19. 下列叙述何者正确？(多选)

(A) 调用方法有返回值，可以不用变量来存储

(B) 调用方法无返回值，可以用变量来存储

(C) 调用方法时，栈内存创建临时空间执行方法体的代码

(D) 调用方法结束时，内存空间由垃圾回收器(GC)回收

20. 下列叙述何者正确？

(A) 方法设计中可以返回多个返回值

(B) 调用方法有返回值，可以不用 return 关键字返回值

(C) 调用方法时，堆内存创建临时空间执行方法体的代码

(D) 调用方法结束时，内存空间立即回

21. 下列有关编译检测的悲观性原则叙述何者正确？

(A) 方法设计中可以返回多个返回值

(B) 调用方法有返回值，可以不用 return 关键字返回值

(C) 方法设计了返回值类型，就必须给返回值

(D) 调用方法结束时，内存空间立即回收

面向对象(II)

1. 方法重载中参数列表的不同体现在哪里？(多选)

   (A) 参数的个数多寡；

   (B) 参数的类型种类；

   (C) 参数的传值方式；

   (D) 参数的顺序位置。

2.
```
public class TestOverload {
   public void test( int...x ){
      System.out.println( "执行了  test  方法携带动态列表" );
   }
   public void test(){
      System.out.println( "执行了  test  方法没有携带参数" );
   }
   public static void main(String[] args){
      TestOverload to = new TestOverload();
      to.test();
   }
}
```

   调用上列方法重载的无参数方法，执行结果为？

   (A) 执行了 test 方法携带动态列表

   (B) 执行了 test 方法没有携带参数

   (C) 编译时报错

   (D) 运行时报错

3. 如图所示，下列叙述何者正确？(多选)

```
public class TestOverload{
    public void test( int...x ){
        ⋮
    }
    public static void main( String args ){
        TestOverload to = new TestOverload();
        to.test( 1,2,3,4 );
    }
}
```

   (A) int…x 用法称为动态参数列表

   (B) x 本质上就是一个数组

   (C) int…x 只能有一个，且必须放置在方法参数的末尾

   (D) int…x 和 int[] array 分别为方法的参数列表，可构成方法重载。

4. 如图所示，下列叙述何者正确？(多选)

```
public class TestOverload{
    public void test( int x, int...y ){
        ⋮
    }
    public static void main( String args ){
        TestOverload to = new TestOverload();
        to.test( 1, 2, 3, 4 );
    }
}
```

   (A) x == 1

   (B) x == 4

   (C) y[2] == 3

   (D) y[0] == 2。

5. 类的内部成员可以有：(多选)

   (A) 属性

   (B) 方法

   (C) 构造方法

   (D) 程序块(代码块)。

6. 下列叙述何者正确？
   (A) 类的内部成员只有属性、方法和构造方法
   (B) 每一个类只能有一个构造方法
   (C) 构造方法的作用是构建当前类的对象
   (D) 构造方法通过 this 关键字调用。

7. 下列有关构造方法的叙述何者正确？(多选)
   (A) 每一个类都有构造方法
   (B) 构造方法的作用是构建当前类的对象
   (C) 构造方法的名字与当前类名一致
   (D) 构造方法通过 new 关键字调用
   (E) 一般方法中，可以使用 this 关键字调用构造方法。

8. 在类中 this 关键字可以在一个构造方法内调用另一个构造方法，下列语法何者正确？
   (A) this.类名();
   (B) this();
   (C) this.方法名();
   (D) this{};

9. 下列有关程序块的叙述何者正确？(多选)
   (A) 程序块是类的成员之一
   (B) 每次调用构造方法时，系统会先调用程序块执行一遍
   (C) 程序块名字与当前类名一致
   (D) 程序块没有重载
   (E) 可以在类中定义多个程序块。

10. 如图所示，其代码目的为何？

```
public class Person{
    public Person(String num){
        System.out.println("吃了 "+num+" 碗面");
    }
}
```

   (A) 定义类属性成员
   (B) 定义类一般方法成员
   (C) 定义类构造方法成员
   (D) 定义构造方法重载
   (E) 定义类程序块成员

11. 如图所示，其代码目的为何？

```
public class Person{
    {
        System.out.println("我是被定义的类成员");
    }
}
```

   (A) 定义类属性成员
   (B) 定义类一般方法成员
   (C) 定义类构造方法成员
   (D) 定义构造方法重载
   (E) 定义类程序块成员

12. 如图所示，其代码目的为何？

```
public class Person{
    public String tellName(){
        return "My name is C.S. Tau";
    }
}
```

(A) 定义类属性成员
(B) 定义类一般方法成员
(C) 定义类构造方法成员
(D) 定义构造方法重载
(E) 定义类程序块成员

13. 如图所示，其代码目的为何？

```
public class Person{
    public Person(){
    }
    public Person(String num){
        System.out.println("吃了 "+num+" 碗面");
    }
}
```

(A) 定义类属性成员
(B) 定义类一般方法成员
(C) 定义类构造方法成员
(D) 定义构造方法重载
(E) 定义类程序块成员

14. 如图所示，其代码目的为何？

```
public class Person{
    public String name;
    public int age;
    public boolean sex;
}
```

(A) 定义类属性成员
(B) 定义类一般方法成员
(C) 定义类构造方法成员
(D) 定义构造方法重载
(E) 定义类程序块成员

15. 如图所示，其代码定义了哪些成员？(多选)

```
public class Person{
    public String name;
    public int age;
    public boolean sex;
    {
        System.out.println("人类的吃，可以有好多种选择");
    }
    Person(){}
    Person(String name, String num){
        this.name = name;
        System.out.println(name+"吃了"+num+"碗面");
    }
}
```

(A) 定义类属性成员
(B) 定义类一般方法成员
(C) 定义类构造方法成员
(D) 定义构造方法重载
(E) 定义类程序块成员

16. int value1 = input.nextInt();
String value2 = input.next();

消息队列

1	2	3	回车	王	小	明	回车		

消息队列内容如图所示，创建 Scanner 类 input 对象后，调用上列 Scanner 类的方法，下列结果何者正确？

(A) value1 == 123, value2 == "王小明"
(B) value1 == 123, value2 == ""
(C) value1 == "123", value2 == "王小明"
(D) value1 == "123", value2 == ""

17. int value1 = input.nextInt();

消息队列

1	2	3	空格	王	小	明	回车		

String value2 = input.nextLine();

消息队列内容如图所示，创建 Scanner 类 input 对象后，调用上列 Scanner 类的方法，下列结果何者正确？

(A) value1 == 123, value2 == "王小明"

(B) value1 == 123, value2 == " 王小明"

(C) value1 == "123", value2 == " 王小明"

(D) value1 == "123", value2 == ""

18. String value1 = input.nextLine();
int value2 = input.nextInt();

消息队列

王	小	明	回车	1	2	3	回车		

消息队列内容如图所示，创建 Scanner 类 input 对象后，调用上列 Scanner 类的方法，下列结果何者正确？

(A) value1 == "王小明", value2 == 123

(B) value1 == "王小明", value2 == " 123"

(C) value1 == "王小明 123", value2== " "

(D) value1 == "王小明", value2 == ""

19. String value1 = input.next();
String value2 = input.nextLine();

消息队列

王	小	明	空格	1	2	3	回车		

消息队列内容如图所示，创建 Scanner 类 input 对象后，调用上列 Scanner 类的方法，下列结果何者正确？

(A) value1 == "王小明 123", value2 == ""

(B) value1 == "王小明", value2 == "123"

(C) value1 == "王小明", value2 == " 123"

(D) value1 == "王小明", value2 == ""

20. String value1 = input.next();
String value2 = input.next();

消息队列

王	小	明	空格	1	2	3	回车		

消息队列内容如图所示，创建 Scanner 类 input 对象后，调用上列 Scanner 类的方法，下列结果何者正确？

(A) value1 == "王小明 123", value2 == ""

(B) value1 == "王小明", value2 == "123"

(C) value1 == "王小明", value2 == " 123"

(D) value1 == "王小明", value2 == ""

21. 设计类似数组容器(ArrayBox)的类(不必包含主方法)，数组元素的数据类型为 int 整数，默认初始长度为 10；ArrayBox 能添加(add 方法)、删除(remove 方法)、访问(get 方法)以及读取 ArrayBox 目前有效元素的长度(size 方法)，若索引值超出范围抛出自定义 BoxIndex Out of Bounds Exception 异常。

设计思路

ArrayBox 中需要：

1. 数组 int[] arrayBox; //默认长度为常量 DEFAULT_LENGTH = 10

有效元素个数 size //用来记录数组中有效元素的个数
构造方法(无参数) //用来同时设定 arrayBox 为默认长度的数组
构造方法(有参数) //用来同时设定 arrayBox 为用户指定长度的数组

2. 设计一个 add 方法，用来添加元素
需要参数 添加的元素值，需要返回值 确认添加元素是否成功
public boolean add(int element){}
判断 arrayBox 数组长度可以装得下添加的元素
(即有效长度 size 要小于 arrayBox 数组长度)
设计一个 ensureLength 方法来做这件事
ensureLength 方法需要参数 添加元素后的最小有效元素长度，无返回值
private void ensureLength(int minLength){}
判断 arrayBox 数组长度大于 minLength，否则，扩充数组容量
设计一个 growth 方法来做这件事
growth 方法需要参数 minLength，无返回值
private void growth(int minLength){}
扩容算法 //计算出新的数组长度 newLength
将原来有效元素从旧数组中搬到新数组去
设计一个 copyOut 方法来做这件事
copyOut 方法需要参数 旧数组、新数组长度，需要返回值 新数组
private int[] copyOut(int[] oldArray, int newLength){}
创建一个新数组
将旧数组元素全部移入新数组内
将新数组返回给 growth 方法
growth 方法返回 ensureLength 方法
ensureLength 方法返回 add 方法
add 方法将添加元素添加到新数组中
返回 true 添加成功，结束 add 方法

3. 设计一个 get 方法，用来访问元素
需要参数 访问元素的位置(相当于索引值)
需要返回值 访问位置内的元素
public int get(int index){}
检测 index 范围是否合法
设计一个 rangeCheck 方法来做这件事
rangeCheck 方法需要参数 index，无返回值
private void rangeCheck(int index){}
检测 index 范围是否合法

4. 设计一个 remove 方法，用来删除元素
需要参数 删除元素的位置(相当于索引值)
需要返回值 删除掉的那个元素(以防误删除，找不回删除的元素)
public int remove(int index){}

检测 index 范围(复用 rangeCheck 方法)
将 index 位置的删除元素值保存起来
从 index 开始 至 size-1 为止，将后面位置元素依次前移覆盖
返回被删除元素值给用户

5. 设计一个 size 方法，用来获取数组中有效元素的长度
无需参数，有返回值　有效元素个数(长度)

```
public class ArrayBox{
    private static final int DEFAULT_LENGTH = 10; //数组 arrayBox 默认长度
    private int[] arrayBox; //长度一旦确定不能再次发生改变
    private int size = 0; //记录 arrayBox 数组中有效的元素个数
    public ArrayBox(){
        arrayBox = new int[ DEFAULT_LENGTH ];
    }
    public ArrayBox( int arrayBoxLength ){
        arrayBox = new int[ arrayBoxLength ];
    }
    Public boolean add( int element ){
        //确保 arrayBox 属性数组的内部容量
        this.ensureLength( size + 1 );
        //如果上面的这一行代码可以执行完毕，arrayBox 肯定有空间
        //把 element 存入 arrayBox 里，有效元素个数加 1
        arrayBox[ size++ ] = element;
        //告知用户存储成功
        return true; //表示成功
    }
    private void ensureLength( int minLength ){
        if( minLength - arrayBox.length > 0){
            //证明需要的最小容量比原来 arrayBox 的容量还要大，存不下啦
            //扩充 arrayBox 容量
            this.growth( minLength );
        }
    }
    private void growth( int minLength ){
        //获取旧数组的长度
        int oldLength = arrayBox.length;
        //扩容算法以旧数组的 1.5 倍扩容
        int newLength = oldLength + ( oldLength >> 1 );
        //若按照算法扩容后，所需要的空间还不够，那就直接利用 minLength
        if( newLength - minLength < 0 ){
```

```
            newLength = minLength;
        }
        //最终获取到一个合理的长度  newLength
        //按照这个新的长度，创建一个新的数组，旧数组的元素全部搬入到新数组中
        arrayBox = this.copyOut( arrayBox, newLength );
        //新数组的位址引用还是指向  arrayBox
    }
    private int[] copyOut( int[] oldArray, int newLength ){
        //创建一个新的数组
        int[] newArray = new int[ newLength ];
        //将旧数组元素全部移入新数组内
        for( int i = 0; i < oldArray.length; i++){
            newArray[ i ] = oldArray[ i ];
        }
        //将新数组返回
        return newArray;
    }
    public int get( int index ){
        //检测  index  范围是否合法
        this.rangeCheck( index ); //rangeCheck  方法来检测  index  范围是否合法
        //如果上面一行代码可以走过去，证明  index  是合法
        return arrayBox[ index ];
    }
    private void rangeCheck( int index ){
        if( index < 0 || index >= size ){ //判断  index  不合法的条件
            //参考数组的操作，自己定义一个异常(自己创建的类)来说明这个问题
            throw new BoxIndexOutOfBoundsException( "Index: " + index + " + size: " + size );
        }
    }
    public int remove( int index ){
        //检测  index  范围
        this.rangeCheck( index );
        //如果这行代码可以执行过去，index  合法
        //将  index  位置的旧值保存起来
        int oldValue = arrayBox[ index ];
        //10 20 30 40 50 60 0 0 0 0 0 ---> 有效元素  size   6 个
        //box.remove(2); (将元素  30  删掉)
        //10 20 40 50 60 0 0 0 0 0 0 ---> 有效元素  size   5 个
        for(int i=index;i<size-1;i++){//i==2 3 4
            //从  index  开始至  size-1  为止，将后面位置元素依次前移覆盖
```

```
            arrayBox[ i ] = arrayBox[ i+1 ];
        } //10 20 40 50 60 0 0 0 0 0
        arrayBox[ --size ] = 0; //末尾元素清空
        //将旧值返回给用户
        return oldValue;
    }
    public int size(){ //用户只能读取 size 值，但无法改变其值
        return this.size;
    }
}

public class BoxIndexOutOfBoundsException{
    //描述这个类是一个(is a)我们自己定义的异常
    //自定义的异常可以透过继承 extends 或泛化(实现接口) implements 两种方式达成
    public BoxIndexOutOfBoundsException(){}
    public BoxIndexOutOfBoundsException( String msg ){
        super( msg ); //msg 提供给父类
    }
}
```

面向对象(III)

1. 类和类之间有哪些关系？

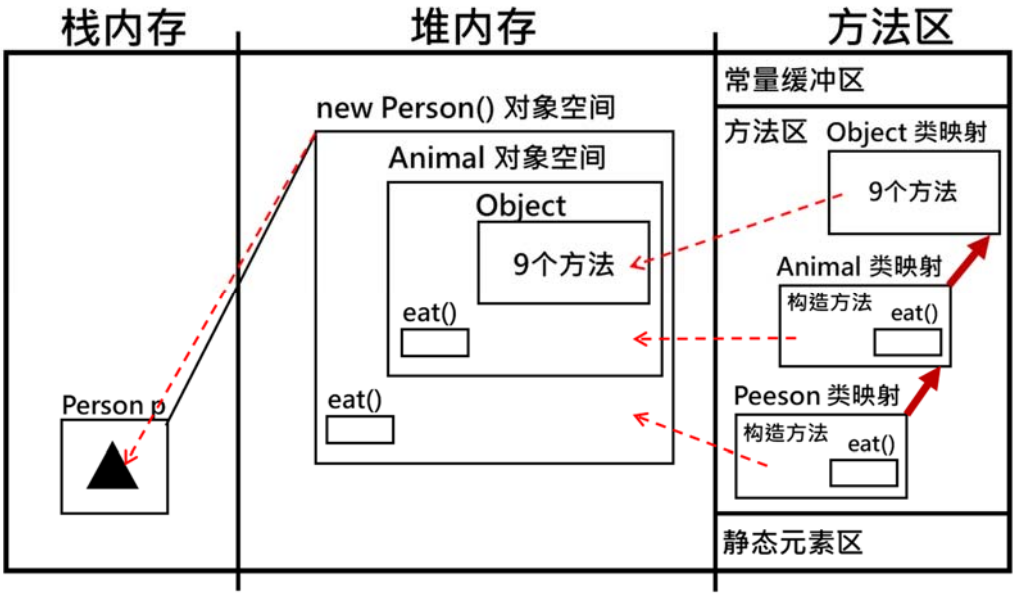
   A is-a B：继承（实现）。

   A has-a B：包含（组合、聚合、关联）。

   A use-a B：依赖（依赖）。or (A need-a B)

2. 请画出下列代码在 JVM 内存中的存储形式。

```
public class Animal {
   public Animal() { … }
   public void eat() { … }
}
public class Person extends Animal {
   public Person() { … }
   public void eat() { … }
}
public class Test {
   public static void main(String[] args){
      Person p = new Person();
   }
}
```



3. 执行下列主方法代码后的输出结果为何？

```
public class Animal {
   public Animal(){
      System.out.println("Animal 中无参数的构造方法");
   }
```

```
    public Animal(int a){
        System.out.println("Animal 中的 int 参数构造方法");
    }
    public void eat(){
        System.out.println("动物的吃饭方法");
    }
    public void sleep(){
        System.out.println("动物的睡觉方法");
    }
}
public class Person extends Animal {
    public Person(){
        this(10);
        System.out.println("person 中的无参数构造方法");
    }
    public Person(int a){
        System.out.println("person 中的 int 参数构造方法");
    }
    public void eat(){
        super.eat();
        this.study();
        System.out.println("人类的吃饭方法，讲究色香味俱全");
}
    public void study(){
        System.out.println("good good study, day day up");
    }
}
public class Test {
    public static void main( String[] args ){
        Person p = new Person();
        p.eat();
    }
}
```

```
Animal 中无参数的构造方法
person 中的 int 参数构造方法
person 中的无参数构造方法
动物的吃饭方法
good good study, day day up
人类的吃饭方法，讲究色香味俱全
```

4. 在 Java 中子类继承了父类，可以调用父类中（public 和 protected）的属性和方法；那子类是否也继承了父类的构造方法及块？

构造方法严格意义来说，不能算做是子类继承过来的；只是单纯的在子类调用构造方法时，默认调用父类的构造方法。

程序块严格意义来说，也不能算做是子类继承过来的（程序块子类不能直接调用，因为程序块没名字）；子类执行构造方法之前，默认调用了父类的构造方法，执行父类构造方法之前自动执行父类的程序块。

5. 方法重写（Override）和方法重载（Overload）有何区别？

	方法重写（Override）	方法重载（Overload）
类	产生两个继承关系的类，子类重写父类的方法	一个类中的一组方法
权限修饰符	子类可以大于等于父类	
特征修饰符	父类方法是 final，子类不能重写 父类方法是 static，子类不存在 父类方法是 abstract，除非子类也是抽象类，否则子类必须重写	没有要求
返回值	子类可以小于等于父类	没有要求
方法名	子类与父类一致	一个类中的同一组重载方法名必须一致
参数	子类与父类一致	每一个重载方法的参数必须不一致 (体现在参数的个数、类型、顺序)
异常	如果父类方法抛出运行时异常，子类可以不予理会 如果父类方法抛出编译时异常，子类抛出异常的个数少于等于父类，子类抛出异常的类型小于等于父类	没有要求
方法体	子类的方法内容与父类不一致	每一个重载方法的执行过程(内容)不一致

6. 包含关系从亲密程度来讲，有组合、聚合、关联三种不同层次，请描述这三种关系并举生活中的例子说明。Java 程序设计中如何描述包含的关系？

组合：整体和部分的关系，紧密到不可分割，要出现都出现，要消亡都消亡。

例如：人和大脑、人和心脏等的关系。

聚合：整体和部分的关系，创建时有可能是分开的。

例如：汽车和车轮子、电脑和主板等的关系。

关联：整体和部分的关系，可以分割，后来形成在一起。

例如：人有汽车、人有电脑等的关系。

包含的关系用 Java 程序来描述，乃是通过一个类的对象当做另一个类的属性来存储。

7. 请设计主方法来完成以下程序设计的赋值，已知车的参数信息为 "宝马", "Z4", "宝石蓝色"，车轮的参数信息为 "米其林", 400, "酷黑"。

```
package has_a_Relation;
public class Wheel {
   //属性
   public String brand;   //轮子品牌
   public int size;       //尺寸
```

```
    public String color;     //颜色
    //构造方法
    public Wheel(){ }
    public Wheel (String brand, int size, String color){
        this.brand = brand;
        this.size = size;
        this.color = color;
    }
    //方法
    public void turn(){
        System.out.println("车轮子可以旋转");
        }
}

package has_a_Relation;
public class Car {
    //属性
        public String brand;      //汽车品牌
        public String type;          //型号
        public String color;       //颜色
        public Wheel wheel;    //车类里面有轮子类的属性成员  -->  包含关系
    //构造方法
    public Car(){ }
    public Car( String brand, String type, String color, Wheel wheel ){
        this.brand = brand;
        this.type = type;
        this.color = color;
        this.wheel = wheel;
    }
    //方法
    public void showCar(){
        System.out.println("这是一辆"+brand+"牌"+type+"型号"+color+"的小汽车");
        System.out.println("车上搭载着"+wheel.brand+"牌"+wheel.size+"尺寸"+wheel.color+"颜色
                                的车轮子");
        wheel.turn();
    }
}
```

8. 设计类/接口的原则是「高内聚、低耦合」，其意义何在？类/接口的继承(实现)、包含(组合、聚合、关联)以及依赖关系的耦合紧密程度顺序为何？

高内聚指的是一个类的设计只跟这个类的特性有关系，不应该与其他类的特性有所牵扯。也就是说，这个类只干与自己这个类有关的事，如果有干别的类有关的事情，肯定设计是有问题，应该将其移到别的类中去描述。

低耦合指的是类的关系耦合度应该要愈低愈好，也就是一个类的变动儘可能不会影响到其他类跟着要变动。

继承(实现) > 包含(组合 > 聚合 > 关联) > 依赖

9. 模拟一个学生在机房中内使用电脑,请分析有几个具体类？类和类的关系为何？如何描述类？并将代码实现（含主方法）。

问题解决思路

机房有电脑　　学生进机房用电脑

分析有几个具体类：电脑、学生、机房。

分析类和类之间的关系：

学生-电脑：学生使用电脑 (依赖关系)。

机房-电脑：机房内有电脑 (聚合关系)。

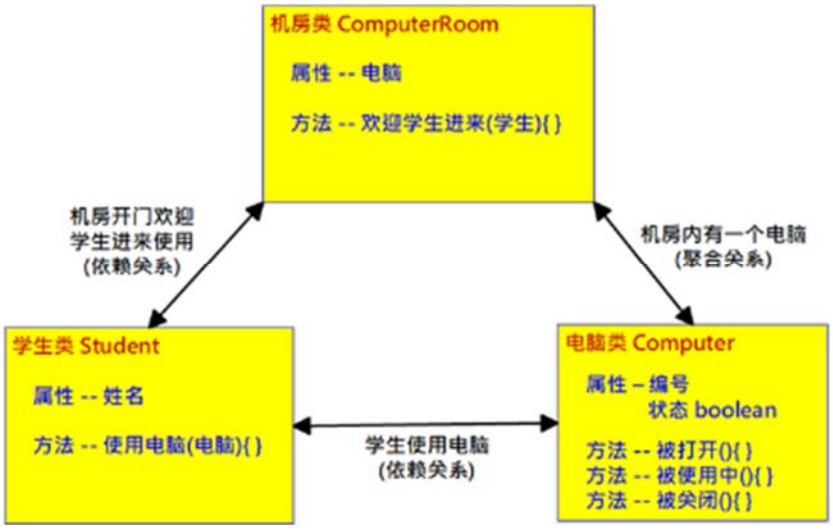
机房-学生：机房欢迎学生进来使用电脑 (依赖关系)。

分别描述三个类：

学生类：属性为姓名，方法为使用电脑方法。

电脑类：属性为电脑状态、编号，方法为电脑被打开、使用中、关闭方法。

机房类：属性为电脑，方法为欢迎学生进来方法。

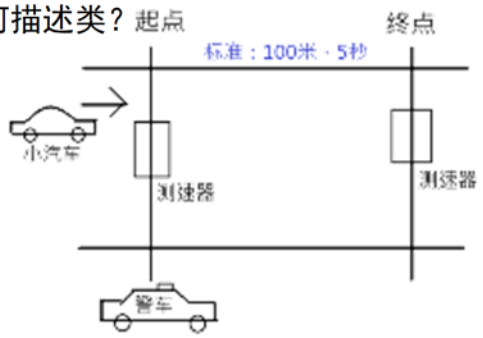



```
package class_relation;
public class Computer {
   //属性
   private int number;   //电脑编号
   //状态，true 开启的，   false 关闭的
   private boolean used = false;
```

```
    //构造方法
    public Computer(){ }
    public Computer( int number ){
        this.number = number;
    }
    //方法
    //获取电脑的编号方法
    public int getNumber(){
        return this.number;
    }
    public boolean isUsed(){    //获取电脑状态方法
        return this.used;
        }
    public void beOpen(){    //电脑被打开方法
        this.used = true;    //状态切换成开启的
        System.out.println( this.number + "号电脑被打开啦" );
    }
    public void beUsing(){    //电脑被使用方法
        System.out.println( this.number + "号电脑正在被使用中  ..." );
    }
    public void beClose(){    //电脑被关闭方法
        this.used = false;    //状态切换成关闭的
        System.out.println( this.number + "号电脑被关闭啦" );
    }
}

package class_relation;
public class Student {
    //属性
    private String name;    //学生的名字
    //构造方法
    public Student(){ }
    public Student( String name ){
        this.name = name;
    }
    //方法
    //获取学生名字方法
    public String getName(){
        return this.name;
    }
```

```
    // 学生使用电脑方法(依赖关系)
    public void useComputer( Computer computer ){
        System.out.println( this.name + "开始使用电脑啦" );
        computer.beOpen();
        computer.beUsing();
        computer.beClose();
    }
}

package class_relation;
public class ComputerRoom {
    //属性
    //机房有电脑(聚合关系)
    public Computer computer = new Computer( 1 );
    //方法
    //机房欢迎学生来使用电脑方法(依赖关系)
    public void welcomeStudent( Student student ){
        String studentName = student.getName();
        System.out.println( "欢迎" + studentName + "进入机房" );
        student.useComputer( computer );
    }
}

package class_relation;
public class Test {
    public static void main(String[] args){
        ComputerRoom room = new ComputerRoom();
        Student student = new Student( "ＸＸＸ" );
        room.welcomeStudent( student );
    }
}
```

10. 模拟一个警车、小汽车、测速器之间的关系。测速器测量小汽车的速度，标准为 100 米、5 秒钟；如果小汽车超速则警车追击；如果警车追击成功，输出追击时间；如果警车追不上，输出追不上啦。请分析有几个具体类？类和类的关系为何？如何描述类？并将代码实现（含主方法）。



问题解决思路

分析有几个具体类：小汽车、警车、测速器。

分析类和类之间的关系：

警车-小汽车：警车追小汽车 (依赖关系)。

测速器-小汽车：测速器测量小汽车（依赖关系）。

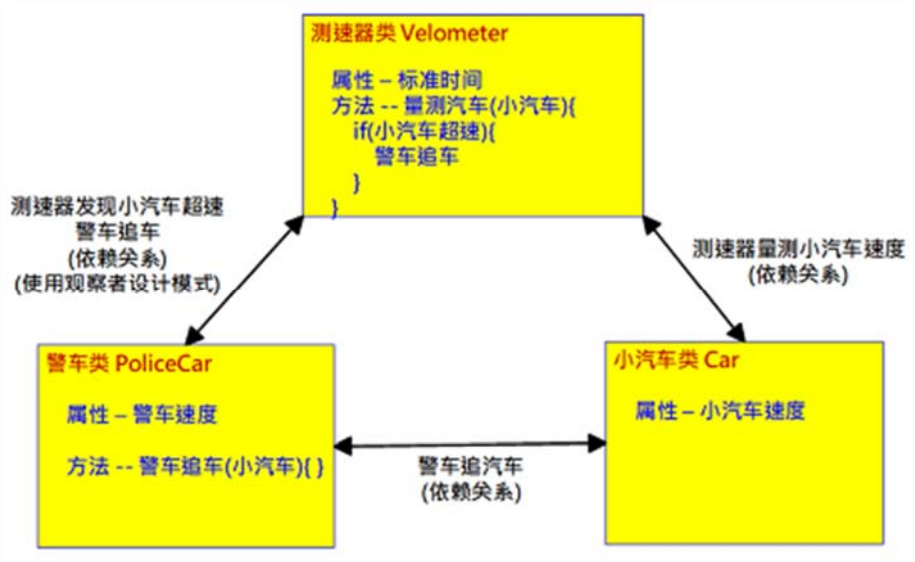
测速器-警车：测速器发现小汽车超速，警车追击（依赖关系）。

分别描述三个类：

小汽车类：属性为速度。

警车类：属性为速度，方法追小汽车方法。

测速器类：属性为标准时间，方法为测量小汽车方法。




```
package class_relation_1;
public class Car {
    //属性--小汽车自己的速度，假设匀速直线运动
    private int speed;
    //构造方法
    public Car(){ }
    public Car( int speed ){
        this.speed = speed;
    }
    //提供一个方法获取小汽车的速度
    public int getSpeed(){
        return this.speed;
    }
}
```

```
package class_relation_1;
public class PoliceCar {
    //属性
    private int speed;    //警车自己的速度
    //构造方法
    public PoliceCar(){ }
```

```
    public PoliceCar( int speed ){
        this.speed = speed;
    }
    //方法
    //警车追击小汽车(依赖关系)
    public void chase( Car car ){
        //获取小汽车速度
        int carSpeed = car.getSpeed();
        //比较两车速度
        if( this.speed > carSpeed ){    //可以追到
            System.out.println( "警车开始追击" );
            int time = 100/( this.speed - carSpeed );    //追击时间
            try {
                Thread.sleep( 3000 );    //编译时异常
            } catch( Exception e ){
                e.printStackTrace();
            }
            System.out.println( "小样儿跑了你啦，经过 " + time + "秒追到啦" );
        }else{    //追不到啦
            System.out.println( "小汽车飞的太快啦，望尘莫及" );
        }
    }
}

package class_relation_1;
public class Velometer {
    //属性
    //测速器规定好的标准时间
    private int standardTime;
    //构造方法
    public Velometer(){ }
    public Velometer( int standardTime ){
        this.standardTime = standardTime;
    }
    //方法
    //测速器测量小汽车速度(依赖关系)
    public void measureCar( Car car ){
        //获取小汽车的速度
        int carSpeed = car.getSpeed();
        //计算小汽车运行时间
```

```
        int carTime = 100/carSpeed;
        //比较
        if( this.standardTime <= carTime ){
            System.out.println( "速度正常，请保持安全行驶" );
        }else{    //超速啦
            System.out.println( "小汽车超速啦,警车可以追击啦" );
            //需要一个警车对象来做事追车
            //测速器发现小汽车超速，通知警车做事(观察者设计模式)
            PoliceCar pc = new PoliceCar( 25 );
            pc.chase( car );
        }
    }
}
```

```
package class_relation_1;
public class Test {
    public static void main( String[] args ){
        Car car = new Car( 18 );
        Velometer v = new Velometer( 5 );
        v.measureCar( car );
    }
}
```

11. 模拟屠夫杀猪问题。流程为 农夫养猪 → 屠夫杀猪 → 猪被杀，养猪须设计长肉的算法。请分析有几个具体类？类和类的关系为何？如何描述类？并将代码实现（含主方法）。

问题解决思路

农夫养猪 → 屠夫杀猪 → 猪被杀

分析有几个具体类：猪、农夫、屠夫。

分析类和类之间的关系：

农夫-猪：农夫养猪 (依赖关系)。

屠夫-猪：屠夫杀猪 (依赖关系)。

农夫-屠夫：在此问题中没有关系。

分别描述三个类：

猪 Pig 类：猪有名字、重量；猪长大、猪被杀的方法。

农夫 Farmer 类：养猪的方法。

屠夫 Butcher 类：杀猪的方法。

```
package use_a_Relation;
public class Pig {
    //属性
    private String name;    //名字
```

```
    private int weight = 20;    //重量(斤)
    //构造方法
    public Pig(){ }
    public Pig( String name ){
        this.name = name;
    }
    //方法
    //描述一个方法，表示小猪被杀啦
    public void beKilled(){
        System.out.println( this.name + "被杀啦，好惨呀！" );
    }
    //  描述一个方法，让猪长肉
    //  算法：每一个月长到前一个月的两倍
    public void growUp( int month ){
        for( int i = 1; i <= month; i++ ){
            this.weight *= 2;
        }
        return this.weight;
    }
    //描述一个方法，猪告知牠的体重
    public int getWeight(){
        return this.weight;
    }
    public String getName(){
        return this.name;
    }
}

package use_a_Relation;
public class Farmer {
    //方法
    //设计农夫养猪方法，返回值为一头猪，需要条件为养几个月
    public Pig feedPig( int month ){
        //在农夫的方法中使用到了猪的对象(依赖关系)
        Pig pig = new Pig( "小花" );
        pig.weight = pig.growUp( month );
        return pig;
    }
}
```

```
package use_a_Relation;
public class Butcher {
    //方法
    //描述一个屠夫杀猪的方法，需要提供条件为一头猪
    public void killPig( Pig pig ){
        System.out.println( "屠夫执行了杀猪方法" );
        String pigName = pig.getName();
        int pigWeight = pig.getWeight();
        System.out.println( pigName + "的体重为 " + pigWeight );
        pig.beKilled();
    }
}

package use_a_Relation
public class Test {
    public static void main( String[] args ){
        //创建农夫对象
        Farmer farmer = new Farmer();
        //农夫做一件事情 --> 养猪
        Pig pig = farmer.feedPig( 5 );
        //创建屠夫对象
        Butcher butcher = new Butcher();
        //屠夫做事 --> 杀猪
        butcher.killPig( pig );
    }
}
```

JavaEE 概述

1. 企业级应用程序的特点。

Ans: 分布式：企业应用程序通常不是运行在某个单独的 PC 上，而是通过局域网运行在一个组织内部，或通过 Intemet 连接分布在世界各地的部门或用户。

高速反应性：社会信息瞬息万变，企业组织必须不断地改变业务规则来适应社会信息的高速变化，相应地，对应用程序也不断提出新的需求。企业应用程序必须具备能力来及时适应需求的改变，同时又尽可能地减少资金的投入。

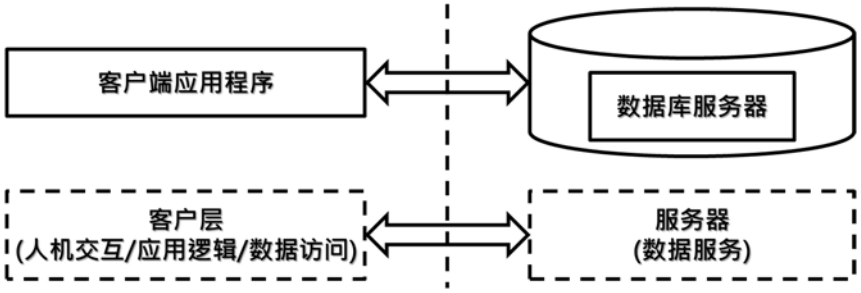
安全性：实现应用系统的正常操作和运转，对于企业的成功来说至关重要。但仅仅做到这点还不够，还必须保证系统运行的安全可靠性。

可扩展性：在网络环境内，应用的潜在用户可能有成百上千，在这种情况下，企业应用除了要考虑能够更加有效的利用企业不断增长的信息资源外，还要充分考虑用户群体的膨胀给应用带来性能上的扩展需求。

集成化：信息是企业资产的重要部分，但目前企业多数信息数据还存放在老的或已经过时的应用系统中。为了最大限度地利用信息资源，要求新的应用必须与目前存在的遗留应用系统相互集成。由于新旧系统间采用的技术、运行的平台可能不同，因此，这种集成要求对于应用开发来说是个很大的挑战。

2. 应用程序体系结构是指应用程序内部各组件间的组织方式，请说明两层体系结构应用程序。

Ans: 两层体系结构应用程序分为客户层 (Client) 和服务器层 (Server)，因此又称为 C/S 模式。



3. 两层体系结构应用程序有何缺点。

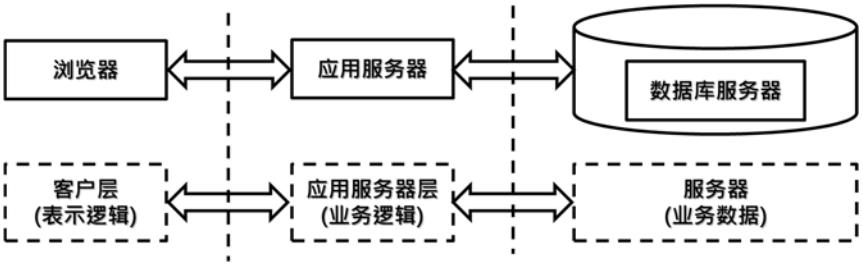
Ans: 安全性低：客户端程序与数据库服务器直接连接，非法用户容易通过客户端程序侵入数据库，造成数据损失。

部署困难：集中在客户端的应用逻辑导致客户端程序肥大，而且随着业务规则的不断变化，需要不断更新客户端程序，大大增加了程序部署工作量。

耗费系统资源：每个客户端程序都要直接连到数据库服务器，使服务器为每个客户端建立连接而消耗大量宝贵的服务器资源，导致系统性能下降。

4. 应用程序体系结构是指应用程序内部各组件间的组织方式，请说明三层体系结构应用程序。

Ans: 三层体系结构应用程序共分为客户层、应用服务器层、数据服务器层三个层次，又称为 B/S 模式。



应用服务器层是位于客户层与数据服务器层中间的一层，因此称“中间件服务器”或“中间件”，应用服务器层又被称做“中间件服务器层”。

5. 相对于两层体系结构应用程序，三层体系结构的应用程序具有哪些优点。

Ans: 安全性高：中间件服务器层隔离了客户端程序对数据服务器的直接访问，保护了数据信息的安全。

易维护：由于业务逻辑在中间件服务器上，当业务规则变化后，客户端程序基本不做改动，只需要升级应用服务器层的程序即可。

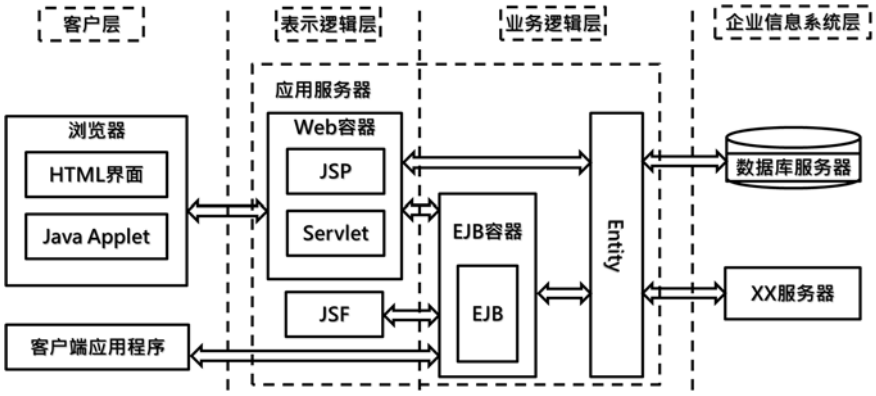
快速响应：通过中间件服务器层的负载均衡及缓存数据能力，可以大大提高对客户端的响应速度。

系统扩展灵活：基于三层分布体系的应用系统，通过在应用服务器部署新的程序组件来扩展系统规模。当系统性能降低时，可在中间件服务器层部署更多的应用服务器来提升系统性能，缩短客户端的响应。

6. 请说明 Java EE 的应用程序体系结构。

Ans: Java EE 是一种体系结构，而不是一门编程语言。Java EE 是个标准中间件体系结构，旨在简化和规范分布式多层企业应用系统的开发和部署。

Java EE 将企业应用程序划分为多个不同的层，并在每个层上定义组件。典型的 Java EE 结构的应用程序包括四层客户层、表示逻辑层(web 层)、业务逻辑层和企业信息系统层。



7. Java EE 为满足开发多层体系结构的企业级应用需求，提出“组件容器”的编程思想。请说明组件与容器之间的关系。

Ans: Java EE 应用的基本软件单元是 Java EE 应用组件。所有的 Java EE 组件都运行在特定的运行环境之中。

组件的运行环境被称为容器。Java EE 组件分为 Web 组件和 EJB 组件，相应地，Java EE 容器

也分为 Web 容器和 EJB 容器。
容器为组件提供必需的底层基础功能称为服务。组件通过调用容器提供的标准服务来与外界交互。

8. Java EE 技术框架为何？。

Ans: Java EE 技术框架可以分为 4 部分：组件技术、服务技术、通信技术和架构技术。

9. 组件是 Java EE 应用的基本单元。Java EE 提供哪些组件？

Ans: Java EE 提供的组件主要包括三类：客户端组件、Web 组件和 EJB 组件。

10. Java EE 体系结构具有哪些优点？

Ans: 独立于硬件配置和操作系统：Java EE 应用运行在 JVM (Java Virtual Machine, Java 虚拟机)上，利用 Java 本身的跨平台特性，独立于硬件配置和操作系统。JRE (Java Run time Environment, Java 运行环境) 几乎可以运行于所有的硬件和操作系统组合。

坚持面向对象的设计原则：面向对象和基于组件的设计原则构成了 Java EE 应用编程模型的基础。Java EE 多层结构的每层都有多种组件模型，因此开发人员所要做的就是为应用项目选择适当的组件模型组合，灵活地开发和装配组件，这样不仅有助于提高应用系统的可扩展性，还能有效地提高开发速度，缩短开发周期。

灵活性、可移植性和互操作性：利用 Java 的跨平台特性，Java EE 组件可以很方便地移植到不同的应用服务器环境中，在全异构环境下，Java EE 组件仍可彼此协同工作。这特征使得装配应用组件首次获得空前的互操作性。

轻鬆的企业信息系统集成：Java EE 技术出台后不久，很快就将 JDBC、JMS 和 JCA 等一批标准归纳自身体系之下，简化了企业信息系统整合的工作量，方便企业将诸如 Legacy System (遗产系统)，ERP 和数据库等多个不同的信息系统进行无缝集成。

搭建开发环境

1. Java EE 应用开发环境分为哪两大类？哪一类比较适合初学者？请举出网络上免费的 Java EE 应用开发环境。

Ans: 目前 Java EE 应用开发环境分为两大类：基于命令行的开发环境和集成开发环境。基于命令行的开发环境利用简单的文本编辑器编写程序代码，通过运行 Java 命令实现程序的编译、发布、运行等操作。

对于初学者来说最适合使用集成开发环境进行入门学习。

网络上一些免费集成开发环境如 NetBeans IDE、Eclipse 等

Servlet

1. 什么是 Servlet 应用程序，它与传统 Java 应用程序有何区别？

Ans: Servlet 是一种独立于操作系统平台和网络传输协议的服务器端的 Java 应用程序，它用来扩展服务器的功能，可以生成动态的 Web 页面。

Servlet 与传统 Java 应用程序最大的不同在于：它不是从命令行启动的，而是由包含 Java 虚拟机的 Web 服务器进行加载。

2. Applet 是运行于客户端浏览器的 Java 应用程序，Servlet 与 Applet 相比较，有何相似与不同之处。

Ans: 相似之处

它们都不是独立的应用程序，没有 main( ) 方法。

它们都不是由用户调用，而是由另外一个应用程序（容器）调用。

它们都有一个生存周期，包含 init( ) 和 destroy( ) 方法。

不同之处

Applet 运行在客户端，具有丰富的图形界面。

Servlet 运行在服务器端，没有图形界面。

3. 请叙述 Servlet 的基本工作流程。

Ans: 客户机将请求发送到服务器。

服务器上的 Web 容器实例化(装入) Servlet，并为 Servlet 进程创建线程。

Web 容器将请求信息发送到 Servlet。

Servlet 创建一个响应．并将其返回到 Web 容器。

Web 容器将响应发回客户机。

服务器关闭或 Servlet 空闲时间超过一定限度时，调用 destrory( )方法退出。

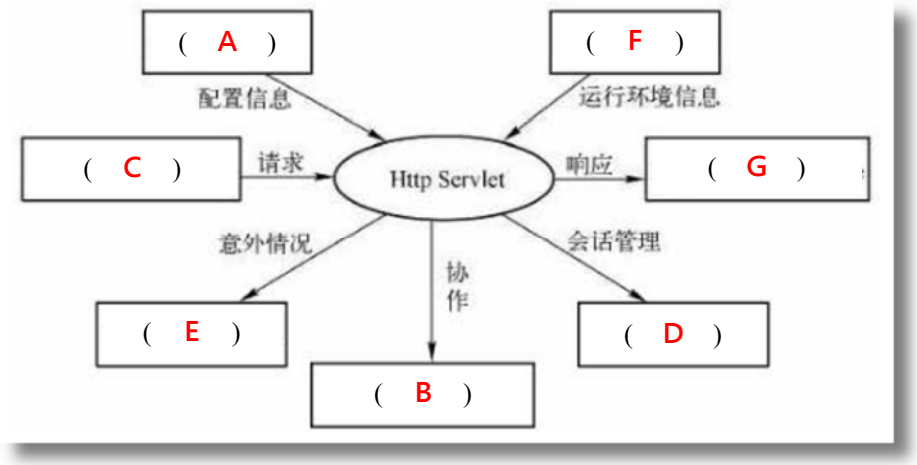
4. 常见会话跟踪技术有哪几种？

Ans: 常见会话跟踪技术有：Cookie、URL 重写和隐藏表单域。

5. 什么是 cookie?

Ans: Cookie 是一小块可以嵌入到 HTTP 请求和响应中的数据。典型情况下，Web 服务器将 Cookie 值嵌入到响应的 Header，而浏览器则在其以后的请求中都将携带同样的 Cookie。Cookie 由浏览器保存在客户端，通常保存为一个文本文件。Cookie 还含有一些其他属性，诸如可选的注释、版本号及最长生存期。

6. 要编写在 Web 上使用的 HTTP Servlet 接口，通常采用继承 HTTP Servlet 接口类的形式。下图以 HttpServlet 接口类为中心，请将下列应用处理与 Servlet 编程密切相关的接口正确对应。



(A) ServletConfig
(B) RequestDispatcher
(C) HttpServletRequest
(D) HttpSession
(E) ServletException
(F) ServletContext
(G) HttpServletResponse

JSP

1. 除了普通 HTML 代码之外，有哪几种 JSP 成分可以嵌入 JSP 页面。

Ans: 除了普通 HTML 代码之外，嵌入 JSP 页面的其他成分主要有三种：脚本元素(Scripting Element)、指令(Directive)和动作(Action)。
脚本元素用来嵌入 Java 代码，这些 Java 代码将成为转换得到的 Servlet 的一部分；
JSP 指令用来从整体上控制 Servlet 的结构；
动作用来引入现有的组件来控制 Web 容器的行为。

2. 什么是 JSP 页面？

Ans: JSP 是一种实现普通静态 HTML 和动态 HTML 混合编码的技术。JSP 页面文件通常以 .jsp 为扩展名，而且可以安装到任何能够存放普通 Web 页面的位置。
从代码编写来看，JSP 页面更像普通 Web 页面而不像 Servlet，但实际上， JSP 最终会被转换成标准的 Servlet。JSP 到 Servlet 的转换过程一般在出现第一次页面请求时进行。
JSP 设计的目的在于简化表示层的表示。JSP 并没有增加任何本质上不能用 Servlet 实现的功能。
JSP 借助内容和外观的分离，页面制作中不同性质的任务可以方便地分开。例如，由页面设计专家进行 HTML 设计，同时留出供 Java 程序员插入动态内容的空间。

3. 脚本<!- comment ->与<%-- comment --%>差别和在？

Ans: 在 JSP 页面中有两种类型的注释：
输出到客户端的注释 <!- comment ->。
不输出到客户端的注释 <%-- comment --%>。它表示是 JSP 注释，在服务器端将被忽略，也不转化为 HTML 的注释，在客户端查看源代码时是看不到的。这种注释一般是开发人员用来实现对程序代码的说明，但同时又防止在客户端看到，有效保护代码的安全性。

4. 下列程序执行后，输出结果为何？

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
  <head>
    <met http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <tile>EL 有效表达式</title>
  </head
  <body>
    ${true} <br>
    ${23+15.28} <br>
    ${12>10} <br>
    ${(12>10)&&(a!=b)}
    ${(12>10)&&('a'!= 'b')}
```

```
    </body>
</html>
```

Ans: true

38.28

true

false

true

5. 请将下列 JSP 格式与其用途或名称正确对应。

(C)	<%＝(a+b)*c %>	(A) 注释
(A)	<!- Hello World ->	(B) 声明方法
(B)	<%! String sayHello(){ return “Hello”; } %>	(C) 输出表达式
(D)	<%@page language="java"%>	(D) JSP 指令
(G)	<jsp:include page="add.jsp">	(E) 逻辑运算式
		(F) 声明变量
		(G) JSP 动作组件

JavaBean

1. 什么是 JavaBean，为什么 JSP 需要与 JavaBean 相结合。

Ans: 目前，JSP 在动态 Web 应用开发中得到了越来越广泛的应用，JSP 提供的内置对象大大方便了 JSP 页面的控制和开发，但由于应用程序功能要求越来越强，JSP 页面变得越来越臃肿且难以控制。因此 JSP 与 JavaBean 相结合成为最常见的 Web 应用程序开发方式。
JavaBean 组件是一些可移植、可重用，并可以组装到应用程序中的 Java 类。
通过在 JavaBean 中封装事务逻辑、数据库操作等，然后将 JavaBean 与 JSP 语言元素一起使用，可以很好地实现后台业务逻辑和前台表示逻辑的分离，使得 JSP 页面更加可读、易维护。

2. 符合哪些设计规则，任何 Java 类都是一个 JavaBean。

Ans: 对于数据类型 “protype” 的每个可读属性，Bean 必须有下面签名的一个方法：public proptype getProperty() { }
对于数据类型 “protype” 的每个可写属性，Bean 必须有下面签名的一个方法：public setProperty(proptype x) { }
定义一个不带任何参数的构造函数。

3. 在 JSP 页面中与 JavaBean 有关的标记有哪些？如何使用？

Ans: <jsp:useBean>、<jsp:setProperty> 和 <jsp:getProperty>。
首先通过标记 <jsp:useBean> 引入 JavaBean。
然后分别利用 <jsp:getProperty>或<jsp:setProperty>标记和直接调用 JavaBean 对象的方法来获取或设置 JavaBean 属性。

4. 说明 JavaBean 的生命周期范围

Ans: 每个 JavaBean 都有一个生命周期范围，Bean 只有在它定义的生命周期范围里才能使用，在它的生命周期范围外将无法访问到它。<jsp:useBean> 利用 Scope 属性来声明 JavaBean 的生命周期范围。
JSP 为它设定的生命周期范围有：page、request、session 和 application。

JDBC

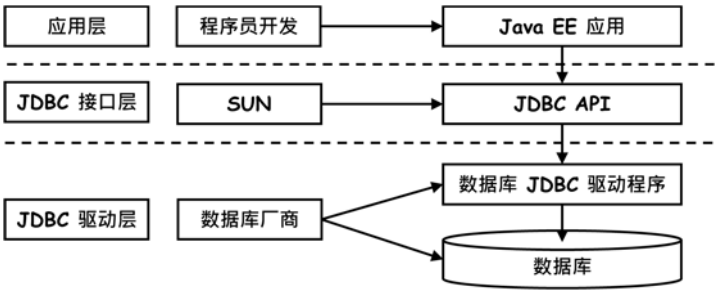
1. 请说明基于 JDBC 的数据库访问模式。

Ans: 基于 JDBC 的数据库访问模式下，数据库访问过程可以清晰地分为 3 层。

最上层为应用层，Java EE 程序开发人员在程序开发过程中通过调用 JDBC 进行数据库访问。

中间层为 JDBC 接口层，它为 Java EE 程序访问各种不同的数据库提供一个统一的访问接口。

最底层为 JDBC 驱动层，它由特定数据库厂商提供的 JDBC 驱动程序来实现与数据库的真正交互。



2. JDBC 驱动是由哪两个类负责与数据库建立连接？

Ans: 从 Java EE 编程技术出发，JDBC 驱动有两个类负责与数据库建立连接。

第一个类是 DriverManager，DriverManager 负责管理已注册驱动程序的集合，实质上就是提取使用驱动程序的细节，这样程序员就不必直接处理它们。

第二个类是实际的 JDBC Driver 类，它是由独立厂商提供的。JDBC Driver 类负责建立数据库连接和处理所有与数据库的通信。

3. 什么是"服务提供商机制(SPM, Service Provider Mechanism)"？

Ans: 当调用 DriverManager 对象的 getConnection() 获取一个代表数据库连接的 java.sql.connection 对象时，DriverManager 会尝试从初始化时已经加载的 JDBC 驱动程序库中选择合适的驱动，或者在当前应用的同一个类加载器中明确加载使用过的驱动。这种功能特性称为服务提供商机制(SPM, Service Provider Mechanism)。

4. 有哪几种 Statement 对象可发送 SQL 语句到数据库对数据库进行操作？

Ans: 有三种 Statement 对象：

Statement 对象用于执行不带参数的简单 SQL 语句；

PreparedStatement 对象用于执行带或不带参数的预编译 SQL 语句；CallableStatement 对象用于执行对数据库存储过程的调用。

5. Statement 对象提供几种执行 SQL 语句的方法？

Ans: Statement 对象提供了三种执行 SQL 语句的方法：executeQuery()、executeUpdate() 和 execute()。

executeQuery() 方法用于产生单个结果集的语句，如 SELECT 语句。

executeUpdate() 方法用于执行 INSERT、UPDATE 或 DELETE 语句及 SQL DDL(数据定义语言)语句，如 CREATE TABLE 和 DROP TABLE 。

execute() 方法用于执行返回多个结果集、多个更新计数或二者组合的语句。

6. 何谓存储过程(Stored Procedure)。

Ans: 存储过程(Stored Procedure) 是数据库系统中一组为了完成特定功能的 SQL 语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给出参数(如果该存储过程有参数)来执行它。

7. 利用 JDBC 访问数据库过程中，主要涉及哪几种资源？

Ans: 对数据库的连接对象 Connection

SQL 语句对象 Statement

访问结果集对象 ResultSet

8. 什么是基于连接池的数据库访问体系结构？

Ans: 为每个客户端请求创建新连接非常耗时，对于连续接收大量请求的应用程序尤其如此。为了改变这种情况，JDBC 会在连接池中创建和维护大量的连接。任何应用程序需要访问数据库传入的请求，将使用连接池中已创建的连接。同样，当请求完成时，连接不会关闭，但是会返回到连接池。