1. 选择题
2. C 2, D 3,C 4,B 5,C 6,D 7,C 8,B 9,A 10,D

二、

1 X 2X

三、

1线程同步

什么是线程同步？

当使用多个线程来访问同一个数据时，非常容易出现线程安全问题(比如多个线程都在操作同一数据导致数据不一致),所以我们用同步机制来解决这些问题。

实现同步机制有两个方法：

1。同步代码块：

synchronized(同一个数据){} 同一个数据：就是N条线程同时访问一个数据。

2。同步方法：

public synchronized 数据返回类型 方法名(){}

就是使用 synchronized 来修饰某个方法，则该方法称为同步方法。对于同步方法而言，无需显示指定同步监视器，同步方法的同步监视器是 this 也就是该对象的本身（这里指的对象本身有点含糊，其实就是调用该同步方法的对象）通过使用同步方法，可非常方便的将某类变成线程安全的类

线程通讯：

为什么要使用线程通讯？

当使用synchronized 来修饰某个共享资源时(分同步代码块和同步方法两种情况）,当某个线程获得共享资源的锁后就可以执行相应的代码段，直到该线程运行完该代码段后才释放对该 共享资源的锁，让其他线程有机会执行对该共享资源的修改。当某个线程占有某个共享资源的锁时，如果另外一个线程也想获得这把锁运行就需要使用wait() 和notify()/notifyAll()方法来进行线程通讯了

注意事项

只有多个synchronized代码块使用的是同一个监视器对象，这些synchronized代码块之间才具有线程互斥的效果，假如a代码块用obj1作为监视器对象，假如b代码块用obj2作为监视器对象，那么，两个并发的线程可以同时分别进入这两个代码块中。 …这里还可以分析一下同步的原理。

对于同步方法的分析，所用的同步监视器对象是this

接着对于静态同步方法的分析，所用的同步监视器对象是该类的Class对象

2，

public static void main(String[] args) {

System.out.println("结果是："+Test.foo(30));

}

public static int foo(int i){

if(i<=0)

return 0;

else if(i>0 && i<=2)

return 1;

return foo(i-1) + foo(i-2);

}

3，java的内存分为两类，一类是栈内存，一类是堆内存。栈内存是指程序进入一个方法时，会为这个方法单独分配一块私属存储空间，用于存储这个方法内部的局部变量，当这个方法结束时，分配给这个方法的栈会释放，这个栈中的变量也将随之释放。

堆是与栈作用不同的内存，一般用于存放不放在当前方法栈中的那些数据，例如，使用new创建的对象都放在堆里，所以，它不会随方法的结束而消失。方法中的局部变量使用final修饰后，放在堆中，而不是栈中。