**No.1**

Classloader create three objects of HelloWorld. Three objects refer to three bytecodes. Before the program starts, the class is loaded into the method area by the class loader. The execution engine reads the bytecode of the method area. Then the PC register points to the location of the main function. The virtual machine starts to reserve a stack frame for the main function in the Java stack (each method corresponds to a stack frame). Then the local method interface is called, and the local operation method will allocate the local method stack for the local method when the local method runs.

**No.3**

No. Because the threads are based on different objects. So the method they used are different. The main method should be changed to the following one to ensure the synchronized block works correctly.

public static void main(String args[]){

Table obj1 = new Table();//only one object

MyThread1 t1=new MyThread1(obj1);

MyThread2 t2=new MyThread2(obj1);

t1.start(); t2.start();

}

**No.4**

If two or more threads share an object, and more than one thread updates variables in that shared object, race conditions may occur.

As it was shown in the picture, when Thread A and Thread B share the same object which contains the constant count, they both add 1 to the constant. It should be incremented twice and had the original value +2. However, as the race condition occur, the result will be incremented only once instead.

To solve this problem, the java synchronized block could be used to guarantee that only one thread can enter a given critical section of the code at any given time.