comments

The results of you experiments probably persuaded you that locks on static methods/variables are shared (so work to prevent interleaving) whereas locks on instance methods/variables only lock on the instance so are not shared and do not help interleaving.

Remember, the rule is: *locks lock on objects*... so work out which object is being locked and you will understand the locking mechanism & the protection it offers.

Some questions still remain...

1. if you can lock on a class object which has been declared as abstract then does this imply that abstract classes have a concrete form somewhere in the background?

   Think about this... can you have a static variable associated with an abstract class? If so what does that imply about abstract classes?

2. What does it mean to synchronize in the following ways...
   synchronize(this) {...}
   synchronize(this.class) {...}
   synchronize(this.super) {...}

3. is it ok to pass locks around (as arguments to methods, etc) – why might you want to do this?

4. if you use multiple locks on both class and instance objects are they (i) nested (& dependent on each other) or independent? Why would you want to use multiple locks? What are the risks?