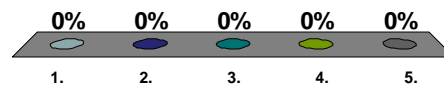


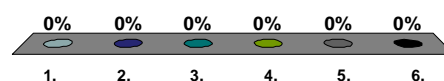
Why might we use wait with a timeout?

1. To allow us to give up on being notified and do something else instead
2. To give our thread a higher priority
3. So that we can busy-wait
4. So that we have smaller critical sections
5. So that we can send a notify to our self



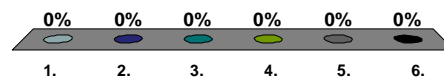
How do we know that wait timed out?

1. The timeout flag is set to true
2. A WaitTimedOut exception is thrown
3. We return from wait()
4. The notified flag is set to false
5. A ThreadNotified exception has not been thrown
6. By checking the fields of our monitor object



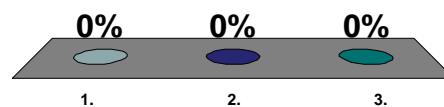
What are the 3 ways of stopping a thread that we discussed in the lectures?

1. `thread.stop()`, `thread.suspend()`, `thread.resume()`
2. `thread.stop()`, `thread.stopRequested()`, `thread.interrupt()`
3. `thread.stop()`, `thread.wait()`, `thread.notifyAll()`
4. `thread.stop()`, `thread.wait()`, `thread.sleep()`
5. `thread.stop()`, `thread.kill()`, `thread.die()`
6. `thread.stop()`, `thread.interrupt()`, `thread.wait()`



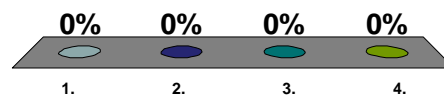
Which of the 3 ways is best?

1. `thread.stop()`
2. `thread.stopRequested()`
3. `thread.interrupt()`



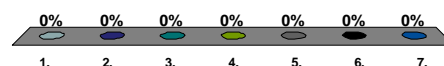
Why is it best to use `thread.interrupt()`?

1. it's deprecated
2. we have to write two additional methods to get and set the interrupt flag
3. as well as checking each time around our main "while" we can exit immediately if sleeping or waiting
4. we will stop regardless of where we are in "run"



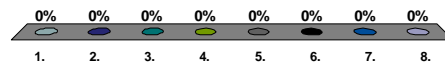
How many ways can we represent an active object in UML?

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7



How many ways can we represent an active class in UML?

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8



How do we represent a nested class in UML?

1. thick lines
2. thin lines
3. filled in diamond
4. filled in square
5. + inside circle
6. - inside circle
7. # inside circle

