CMS1 3510 Homework 3

2. The original dining philosopher's problem can easily cause deadlock because when a philosopher picks up a fork they continuously wait until a second fork is available which results in deadlock if multiple philosophers are waiting to eat or grab forks at the same time. The solution we decided to use implements a waiter as a mutex so that only one philosopher can eat at a time even though there are more forks for them to do so. If one philosopher is already eating and another philosopher is requesting a fork all other philosophers wait even if forks are available for them. This will avoid both deadlock and starvation.

5. The Java class that would be appropriate to use for the holding area is the Bounded Buffer class. The way the bounded buffer class works is that the producer stores values into a buffer which is like a temporary storage space where the consumer goes to retrieve the value from the buffer when it is ready to do so. The pattern of synchronization provided by the bounded buffer class allows multiple consumers and producers to share one buffer. While stuff is being stored into the temporary holding space, the other thread can take stuff and deliver it to the appropriate place. This way, the receive thread can continue to bring things to the temporary space while the other thread will constantly consume those things and send them to their appropriate place.