Cats and Mouse Design Answers – By Christopher Katigbak

1. AvailableBowls, is a semaphore that allows only as many animals as there are bowls to look for a bowl to eat from.
   bowlStatus, is an array of Booleans representing whether the bowl at each index is not being used.
   bowlLock, is a lock that ensures that no animals are accessing bowlStatus at the same time.
   catLock, is a lock that ensures that no cats are accessing hungryCats, catsHaveEaten, or miceHaveEaten at the same time.
   mouseLock, is a lock that ensures that no mice are accessing hungryMice, miceHaveEaten, or catsHaveEaten at the same time.
   hungryCats, keeps track of how many cats haven't eaten since the mice last finished eating.
   hungryMice, keeps track of how many mice haven't eaten since the cats last finished eating.
   catsHaveEaten, is a condition variable that blocks all mice, and then releases them once all cats have eaten once.
   miceHaveEaten, is a condition variable that blocks all cats, and then releases them once all mice have eaten once.

2. AvailableBowls, blocks cats and mice that are waiting to eat until there is a bowl available. This way there will be no animals searching for a free bowl when there aren't any.
   bowlStatus, at index j is checked when an animal wants to know if bowl j is available. If bowl j is available bowlStatus[j] will be true and the animal will eat from it. This prevents animals from eating from a bowl that is being used.
   bowlLock, makes all other animals wait while an animal is checking the status of a bowl, or modifying the status of a bowl. This way now animal is accessing bowlStatus while it is being modified.
   catLock, blocks all other cats while hungryCats is being modified, so it is only being modified by one cat at a time. It also ensures that only one cat is accessing the condition variables catsHaveEaten and miceHaveEaten at once. That way only one cat is being put in the catsHaveEaten wchan at a time. It also ensures that the mice threads that are waiting in the MiceHaveEaten wchan are only broadcasted to once.
   mouseLock, is the mouse equivalent of catLock.
   hungryCats, allows the application to determine whether all the cats have eaten, allowing it to unblock the mice threads.
   hungryMice, is the mouse equivalent of hungryCats.
   catsHaveEaten, blocks all mouse threads so that no mouse will attempt to eat while the cats are eating, and unblocks them when the cats are done.
   miceHaveEaten, is the cat equivalent of catsHaveEaten.

3. AvailableBowls ensures that the amount of creatures looking for a free bowl is equal to the amount of free bowls. The array bowlStatus ensures that each

creature knows if a bowl is taken, thus it will not try to take a bowl in use, and bowlLock makes sure that the bowlStatus remains correct as only one thread is modifying it at a time.

4. When the mouse threads first run they are blocked by the catsHaveEaten condition variable, thus allowing cats to eat first. Each cat thread increments the hungryCats variable, and after each cat eats it is then blocked by miceHaveEaten, and the amount of hungryCats is reduced. The mouse threads are only unblocked after hungryCats is 0. Then the same pattern is repeated, with the mice incrementing hungryMice, eating, getting blocked again by catsHaveEaten, and unblocking cats once there are 0 hungryMice. This cycle repeats until all animals have eaten the appropriate amount of times. Therefore one species is always blocked while the other is waiting or eating. Therefore a cat and a mouse are never trying to eat at the same time.

5. Cats and mice will not starve with this technique because each species only has to wait for each member of the other species to eat before they get a chance to eat again. Thus each animal has to wait at most the amount of time it takes for every other animal to eat once. If an animal were to starve in this time it would be impossible to feed it without another animal starving.

6. My design is slightly biased towards cats as they get to eat first in every instance this application is run, otherwise it is very fair. This is so because no animal is allowed to eat again until all other animals have eaten. This does come at a cost of efficiency as all but one of the bowls are not being used while the last member of a species is eating.