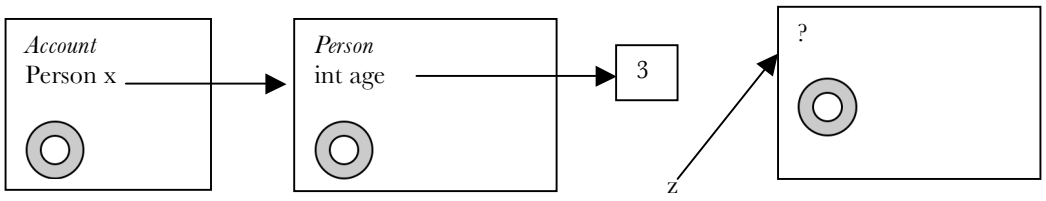
Deadlock

1. Review locks
    a. New analogy: Every Object has a magic ring[1] inside it.
    b. To enter a synchronized (x) block, a thread must wait until it can grab the ring in x
    c. The thread holds the ring while it is in the block
    d. The thread puts the ring back when it leaves the block
    e. Example:




```
// Thread 1:

public void run() {

        x.setAge(x.getAge() + 1);


}
```

```
// Thread 2:

public void run() {

        x.setAge(x.getAge() + 1);


}
```

2. Performance issues of locks (why not write synchronized everywhere?)

3. Deadlock!

4. Software engineering
    a. Communication: shared memory
    b. Threadsafe data structures
    c. Debugging threaded programs

[1] Magic ring = "Token" = "Lock" = "Mutex" when you're reading the notes

Example 2:

```
public class SmartCreature {
   public class Array2D<T> {
      …
      public void set(int x, int y, T v)
   }

   // This is just an example.  You do not need an Array2D<String> for your actual lab
   private static Array2D<String> names = new Array2D<String>();
   private Stack<Integer> todo = new Stack<Integer>();
   …
}
…
SmartCreature frodo = new SmartCreature();
SmartCreature eowyn = new SmartCreature();
new Thread(frodo).start();
new Thread(eowyn).start();
```