Critical sections of the program:

1) class Boss:
   a. *getWork()*

   This method is called from the class Worker when he asks his Boss for a block of resources to mine. This method is a critical section because all Workers can call this method, therefore if we do not protect this mehod then multiple Workers could be mining the same resource resulting in more resources mined than there actually is.

2) class Lorry:
   a. *fillLorry()*

   This method is used by Workers so that they can fill the Lorry once they finished mining their block. All Workers are loading resources on the same Lorry (which has a max capacity) and if we do not synchronize this method multiple Workers could fill the Lorry resulting in exceeding the capacity of the Lorry.

   b. *getLorryState(), getCurrentCap()*

   These methods also need to be synchronized because the Lorry's state and current capacity can both be changed by another thread in the fillLorry() method. Therefore, we need to get a lock on the Lorry object so that during our queries no changes to the object can be made by another thread.

3) class Ferry:
   a. *synchronize()*

   This part of the program is also a critical section because the Ferry works like a Barrier, and it has a counter for Lorries. When a thread that runs the Lorry class calls this method, it will sleep and be awoken once a specific number of Lorries is on board the Ferry. Multiple Lorries calling this method at once without locking the Ferry object could result in unexpected behaviour.

4) class Logger:
   a. *log()*

   The Logger class is basically a database of all Logs that were made by other objects during the run of the program. And as we know from KIV/DB1 - updates to the database need to be atomic operations because if not, the objects that are logging could be updating an old version of the database where the newer changes do not exist yet. This way we would lose some of our logs.