12.1

- idleWorkers keeps track of works that are not actively computing, using a queue to give them tasks so that each worker will work at a certain point in time
- busyWorkers keeps track of works that are actively computing to ensure that the server does not add works that have not been created to idleWorkers
- tasks uses a queue to keep track of each task that is not working and which client provided it, giving older tasks priority
- WorkerIds are the ids assigned to the workers, which are different for each worker

12.2

- maxWorkers is a limit on the number of workers
- totalWorkers ensures that the number created does not exceed the limit by keeping track of the total number of works currently on the server

12.3

- (a) If there is a worker available in idleWorkers, a worker is removed from the pair to busyWorkers and given the task to be computed.
- (b) If all workers are busy and the current number of workers does not exceed the limit, a new worker is added to the busyWorkers and given the task to be computed.
- (c) If there are no free workers and the maximum number of workers has been reached, the task is queued in the task queue.

12.4

- In the constructor, in the 'b' case of the onComputeTasks behaviour, and in the onCrach behaviour, when a new worker is spawned it is also added to the server's 'watch list'.
  The onCrash behaviour is executed under the ChildFailed signal, which quite simply spawns a new worker and adds it to the idle queue.

12.5

- (a) If the task queue is not empty, i.e. work to be completed, the worker who reports that the calculation has been completed will be given the next task in the queue.

- (b) If there is no work pending, the worker will be removed from the busyWorkers collection and added to the idleWorkers queue.