Programming Exam #1
CSCI 201L Spring 2017
10% of course grade

## Read all instructions before starting!

Add buttons for each worker that trigger the corresponding worker to visit the Manager Office, located at the bottom right of the factory, once they are done with their task.

**Note:** A task includes going back to the Task Board to update the product table. If you, the manager, clicks on the "Worker 4" button, that worker should finish making their product and go back to the Task Board before going to the Manager Office.

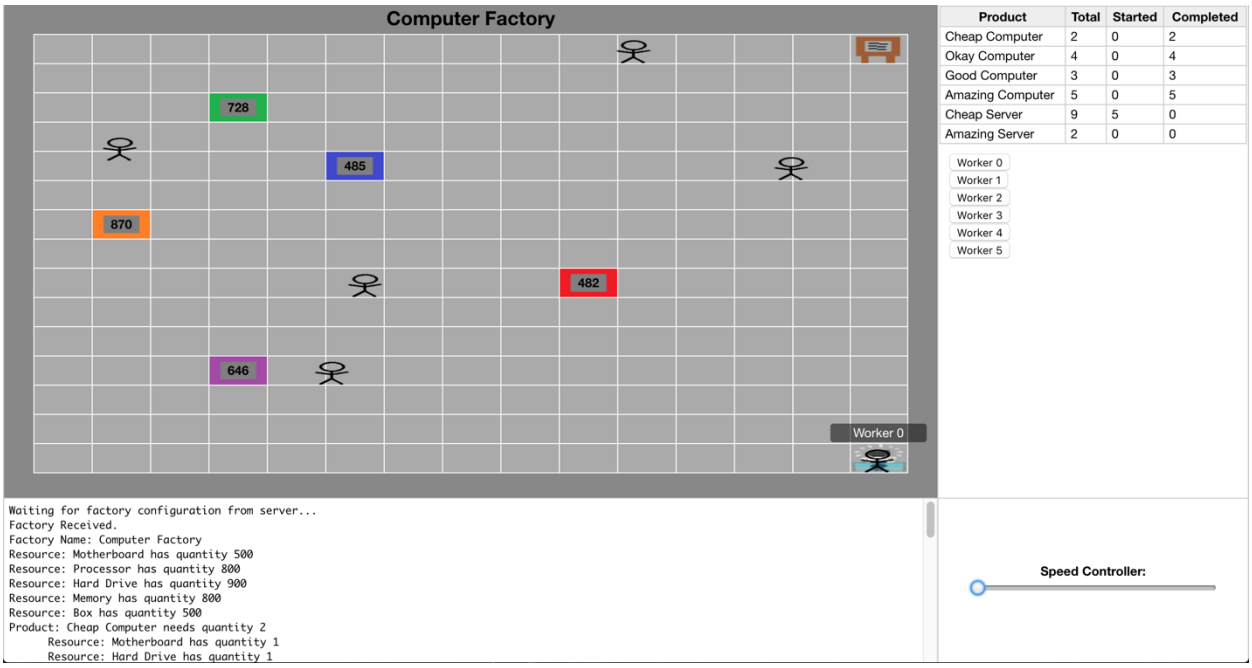

Fig. 1: Worker 0 is at the Manager Office after "Worker 0" button was clicked

## We will test your exam with a different factory.txt file!

Therefore, do not hardcode values or elements. For example, the location of the manager office should depend on the factory data and the buttons should be created dynamically (their names will NOT be "Worker 0," etc.).

| Product | Total | Started | Completed |
|---|---|---|---|
| Cheap Computer | 2 | 0 | 2 |
| Okay Computer | 4 | 0 | 4 |
| Good Computer | 3 | 3 | 0 |
| Amazing Computer | 5 | 3 | 0 |
| Cheap Server | 9 | 0 | 0 |
| Amazing Server | 2 | 0 | 0 |

Worker 0
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5

Fig. 2: Worker buttons (10px margin is optional)

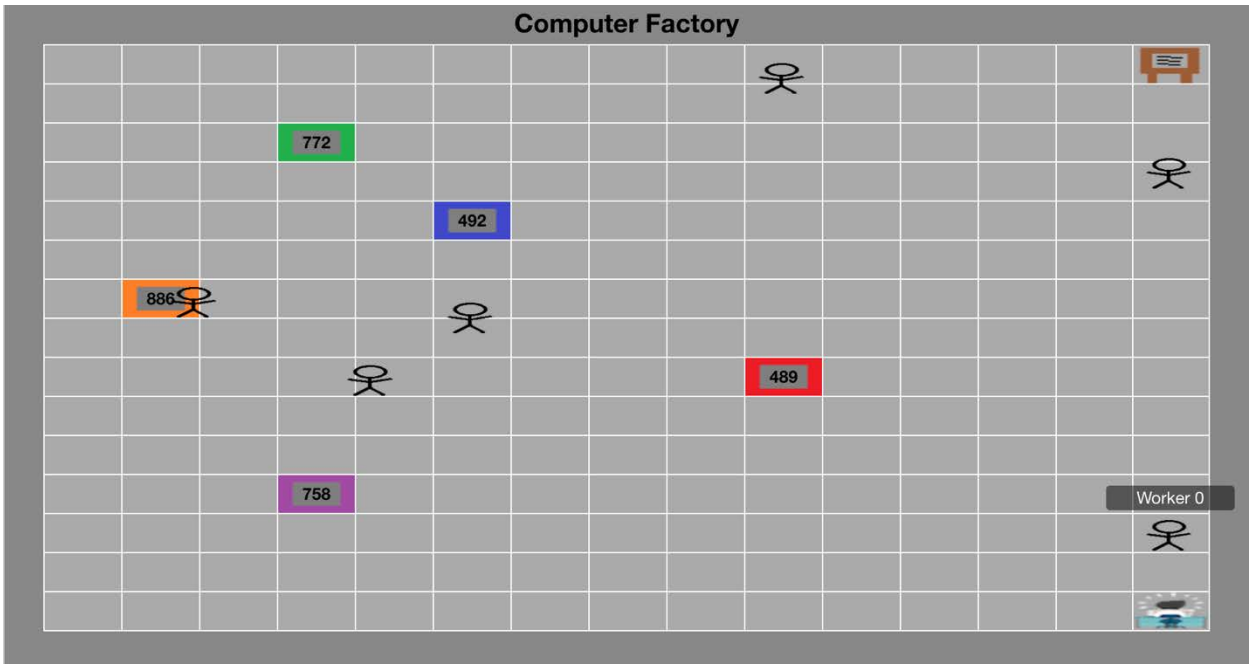


Fig. 3: Worker 0 on their way to the Manager Office

**Things to know:**

**JavaScript function declaration**
Since we didn't cover prototypes in JavaScript, here is how you may declare any of the necessary functions. Know that this is not the only way to name them.

```
Factory.prototype.name_of_function = function(…)
```

**Tip:** just pass in the worker's index into the function that your button will call, not the whole name (e.g. "Worker 0"). Passing the function a parameter with a space causes unwanted behavior that you want to avoid.

**Sending an action**
To send the backend a message that the manager wants to see a given worker, we must use socket.send(…), which we have not covered. Here is the function call which you may copy/edit/paste. It sends the name of an action and the index of the worker that is being requested. You do not have to touch either of the WebSocketEndpoints! See FactoryWorker.js's move(…) function for reference.

```
socket.send(JSON.stringify({
     action: 'Name_Of_Action',
     workerNumber: workerNumber
}));
```

**Synchronization**
After your logic that assigns the worker to go to the manager, add in the following two lines.

```
//backend waits for worker to reach manager in frontend
atLocation.await();
//manager talks to worker for one second
Thread.sleep(1000);
```

**Remember to link any new scripts in your HTML or JSP files!**

# Important:

Testing this exam visually is very "all or nothing." For partial credit, show – via console logs, System prints, etc. – that you implemented things correctly along the way. For example, that your button actually triggers a function. Be sure to comment thoroughly!

Coding style will be graded to some extent, so use proper practices, e.g. create a class where necessary.

**Grading Criteria**

| % of Final Grade | Criteria |
|---|---|
| **2.5%** | There is a Manager Office image at the correct location in the factory |
| **1.5%** | The Manager Office is properly stored in the factory backend |
| **0.75%** | There is a button for each worker displaying the correct name |
| **0.75%** | Each button executes an action specific to the index of the worker it belongs to |
| **1.5%** | Backend receives an action when the manager wants to see a worker |
| **3.0%** | Worker visits Manager Office when appropriate |