EventList
  time
  object
  argString

NOW 0

Now 0

flip 1

25%

2

Q1

Build 1

flip 2

75%

4

Q2

Build 2

flip 1 + 2

flip 2

25%

TestQ

Test

flip 2

25%

Recycle

Ship

Entity

id

in

out

<table>
<thead>
<tr>
<th>flip</th>
<th>coin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>10</td>
</tr>
<tr>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>
Game rules

1. Update event times for servers on game board (use post-its)
2. Create 2 events on post its using template on game board and place them in the EventList: `[10, maker, 'maker'], [400, SimModel,'Stop']`
3. Take turns being the program. As group write psudeo method descriptions for Simulation's simulate() and schedule(…), and server's activity().
   Do this on separate sheet of paper ("CRC card")
1. Turn starts by taking first event from event list and invoke its object activity event.
2. Now is the current time. Initially 0. Update post it on Now stack to show the current time.
3. To get a random time for an object flip one or two coin (different worth).
   Determine time by flip table on game board and server flip instruction.
4. Maker creates new entities and re-schedule itself to make more entities.
5. To create an Entity, use a post it. id ++ (id initially 0), in is now.
6. To create a new Event use a post it. All scheduled times are now + flip time.
7. Place Entity in queue list, or on server square to show its place during the game. If the server following the queue is idle put entity in that server and schedule server.
8. When Entity is done write its out time and place in Done stack.
9. If a queue is full the server trying to insert into the queue is blocked. Put a “blocked” post it on the server (or generator). Blocked servers hold entity until there is space in the output queue. Unblock as space is available.
Component Responsibility Collaborator (CRC) uses index cards to capture a what / who design process. (I use sheets of paper)

Front of card:

<table>
<thead>
<tr>
<th>Component Name (Class name)</th>
<th>Collaborators</th>
</tr>
</thead>
<tbody>
<tr>
<td>list / describe component’s responsibilities (methods)</td>
<td>list of other components</td>
</tr>
</tbody>
</table>

Back of card have data members (instance variables).

I often have brief notes on important method algorithm on additional cards – labelled `<ClassName> :: <type> <MethodName(type args)>`. 