# Computer Assignment 5: Berths with Two Cranes

## Concepts

* Use of containers
* Passing parameters in waitDelay() call
* Use of interrupt() to implement cancelling edge
* Extending Entity class with attributes that can change
* Using Adapter to connect components

## Description

## A small harbor unloads ships using two cranes mounted on a track. There is room at dockside for two ships. When two ships are present, each crane unloads one of them. When only one ship is present, both cranes work on unloading it, reducing the unloading time by half. If another ship arrives when there is only one being unloaded, then one crane will immediately begin unloading that ship, thus increasing the remaining unloading time of the first ship by a factor of two. Arriving ships wait in a first-in first-out queue if both positions dockside are taken. If a ship completes unloading when there are two present, then if there is a ship waiting, the first one begins unloading with the free crane. If the queue is empty, that crane switches back to unloading the remaining ship, thus decreasing again the remaining unloading time by half.

## Ship Entity

Create an Entity subclass called Ship. This will have one additional attribute, the remaining unloading time (if one crane is working on it). It should also have a method called creditWork(double amount) that decreases the amount of remaining unloading time by the amount passed in. When a ship arrives, its remaining unloading time is initialized according to a specified probability distribution (see the Output section below for the specific values). In the event graph in Figure 3 below, a Ship’s remaining time is denoted ‘r’. In your code, it should be named something more descriptive, such as remainingUnloadingTime. The class diagram for the Ship entity is shown in Figure 1



Figure . UML Class Diagram for Ship

The work(double) method should assume that the Ship’s elapsed time is how much calendar time it has been processing; the argument is the rate at which the work is to be credited. In this model, 1.0 is passed to the work method to credit work when one crane has been unloading on the Ship, whereas 2.0 is passed to the work method to credit work of two cranes unloading on it. Specifically, remainingUnloadingTime is decremented by the argument times the elapsedTime, as follows:

public void work(double rate) {

this.remainingUnloadingTime -= getElapsedTime() \* rate;

}

This of course assumes that the Ship has had its timestamped at the event representing the start of unloading (either with one or with two cranes). Also, it would be a good idea to throw an IllegalArgumentException if the rate argument passed in was not a positive number.

## ShipArrivalProcess

Create a ShipArrivalProcess class; this will be similar to the CustomerArrivalProcess class from a previous computer assignment. It will subclass ArrivalProcess and add a parameter unloadingTimeGenerator of type RandomVariate. The Event Graph for ShipArrivalProcess is as follows. (where {*tA*} = interarrival times of ships and {tU} = unloading times of ships when a single crane is operating).

Table . Parameters for ShipArrivalProcess

|  |  |  |
| --- | --- | --- |
| **Event Graph Name** | **Variable Name** | **Type** |
| tA | interarrivalTimeGenerator | RandomVariate |
| tU | unloadingTimeGenerator | RandomVariate |

Note that the scheduled event with the Ship as argument is called ShipArrival.[[1]](#footnote-1)



Figure . Event Graph for ShipArrivalProcess

## TwoCranesBerth

The primary component that models the berths with the cranes will be called TwoCranesBerth. This will not have any parameters, but will have state variables according to the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Event Graph Name** | **Variable Name** | **Type** | **Initial Value** |
| q | queue | SortedSet<Ship> | empty |
| b | berths | SortedSet<Ship> | empty |
| D | delayInQueue | double | NaN |
| W | timeInSystem | double | NaN |

Table . State Variables for TwoCranesBerth

The Event Graph for TwoCranesBerth is as follows in Figure 3.



Figure . Event Graph for TwoCranesBerth

Note that in the Event Graph, s.r is a ship’s remainingUnloadingTime attribute.

## Listener/Adapter Diagram

Because the event in ShipArrivalProcess with the Ship argument is called ShipArrival, but the TwoCranesBerth component is expecting an event called Arrival, an adapter is required. This is indicated in Figure 4:



Figure . Listener Diagram

To implement an adapter in Simkit, use the Adapter class. Its constructor takes two String arguments: The first is the name of the source event and the second is the name of the adapted event. To create the connection, use the connect method of Adapter, which takes two arguments: the first is the source component and the second is the listener component.

## Output

Run your model for 10 years and collect statistics as shown below. Assume that time between arrival of ships is Exponential with a mean of 0.7 days and that unloading times are uniformly distributed between 0.5 and 1.5 days.

Your output should look like Figure 5.

ShipArrivalProcess.1

unloadTimeGenerator = Uniform (0.500, 1.500)

interarrivalTimeGenerator = Exponential (0.700)

TwoCraneBerths.2

Simulation ended at time 3,650.0

Number of ships arriving: 5,135

Number of ships unloaded: 5,129

Maximum # in queue: 11

Average # in queue: 0.6834

Average # busy berths: 1.1737

Average time in system: 1.3207

Average delay in queue: 0.4857

Figure . Output from Running Model for 10 Years (3,650 Days)

## Implementation Notes

* While this is the most elaborate Simkit implementation so far, all the concepts have been covered in previous computer assignments, except for the use of Adapter.
* Remember the convention for “getters” of container state variables returning a copy.
* Be sure to use instances of the appropriate SimpleStats class when collecting statistics on state variables.
* Be sure to run your model in verbose mode (with a SimplePropertyDumper) as you are developing it.
* It is usually most effective to develop complex models in stages. Write a few events and then run in verbose mode to verify that they are working properly.
* Watch for typos in waitDelay() calls!

## Deliverables

Push your code to Gitlab by COB on the due date.

1. We could have called the rightmost event “Arrival(s)” – the purpose of the name is so that an Adapter will be required. [↑](#footnote-ref-1)