Assignment 1b

The purpose of this assignment was to simulate a coffee house that's open from 8:00am - 8:00pm with one service register. This was implemented in java using discrete event simulation and next-event time advance mechanism.   
To run our simulation in text mode from a terminal type the following line:

“java Main seed replications queue (replication time)”  
  
Required  
seed: long to use for random number generation.  
replications: integer for the number of replications.  
queue: integer for places in the queue.  -1 for infinite places.  
Optional:  
replication time: integer for the duration of each replication in minutes. If this is not used the time will be set to 720 minutes.   
  
This is our result for the different simulations and questions

#### Question 1: What is the average queuing time of the customers? Do your results match those obtained in Assignment 1a, Question 1?

Result of Java simulation: “java Main 1 1 -1 1000000”  
###########################################  
# Average of: 1 replications with #  
# #  
# Seed: 1 #  
# Replications: 1 #  
# Queue: -1 #  
# Replication length: 1000000 min #  
# #  
###########################################  
# Total average customers: 1.9958e+05 #  
# Total average percent rejected: 0.0000% #  
# Total average average queue time: 15.933 min #  
###########################################  
  
Yes the results match! The calculated results from 1a, question 1 was: 16 min average queuing time. This is obtained in our Java simulation by running 1 replication for a long time, this will simulate that the CC is open 24/7 for 694,4 days (1000000 min).

**Result comparison of AnyLogic and our Java simulation 8:00 am - 8:00 pm.**  
  
By setting CC open hours from 8:00 am - 8:00 pm the average queueing time in the AnyLogic simulation in assignment 1.a was 13.654 minutes. In our Java simulation this is obtained by running multiple replications, each representing one day of 720 min (8:00 am - 8:00 pm). The result of this was: 13.253 minutes. This is lower than the result of 24/7 CC because everyday at 8:00am the queue will be empty when the shop opens. This will affect the result by lowering the average queueing time.  
  
Result of Java simulation: “java Main 1 1000 -1”  
  
###########################################  
# Average of: 1000 replications with #  
# #  
# Seed: 1 #  
# Replications: 1000 #  
# Queue: -1 #  
# Replication length: 720 min #  
# #  
###########################################  
# Total average customers: 143.28 #  
# Total average percent rejected: 0.0000% #  
# Total average average queue time: 13.253 min #  
###########################################

#### Question 2: CC wants to service at least 95% of the customers, meaning that maximal 5% of the customers should be rejected because of a full queue. How many places should the queue have to achieve these requirements if there is only one service desk?

To achieve this CC needs minimum of 6 places in the queue which gives us a average of 4.6517% rejected customers.  
  
The result of the Java simulation:  
  
Result of Java simulation: “java Main 2 1000 6”  
  
###########################################  
# Average of: 1000 replications with #   
# #  
# Seed: 2 #  
# Replications: 1000 #  
# Queue: 6 #  
# Replication length: 720 min #  
# #  
###########################################  
# Total average customers: 143.68 #  
# Total average percent rejected: 4.6517% #  
# Total average average queue time: 7.8586 min #  
###########################################

#### Question 3: In the DES algorithm a new Arrival event should only be generated when a Customer arrival event was consumed. In theory one could also generate all Arrival events directly at the beginning of the execution. Why should event generation not be done in this way?

If all arrival events are generated at the beginning, the future event list would be very big when starting the simulation. This increases the time of the simulation because insertion of departures will need a lot of time to be sorted in place. By only generating arrival events when another arrival is consumed the future event list will be very short and this decreases the time of the simulation.