Laboration 2: Processimulering

Ofta består ett större system av flera delar som fungerar på ungefär samma sätt. Då vore det enklast att kunna definiera en slags typ som beskriver hur en sådan komponent fungerar och sedan skapa flera instanser av typen när man sätter igång simuleringsprogrammet. I Java kan man låta en sådan systemdel vara en klass som man sedan kan med hjälp av `new` kan skapa tillräckligt många instanser av. Ett exempel på ett sådant program finns på kursens hemsida. I det programmet finns två klasser som kan användas för delarna i ett system, QueuingSystem och Generator.

Uppgift 1

Utgå från programmet på kursens hemsida och gör följande uppgifter.

a) Hur många betjänare har kösystemen som definieras av klassen QS?

b) Hur många buffertplatser finns det i en QS?

c) Är betjäningstiden slumpmässig eller konstant? Hur lång är betjäningstiden i medeltal? Hur stor är en betjäningsintensiteten?

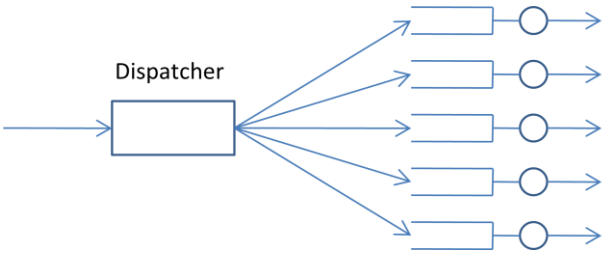
d) Till vad används attributet `sendTo` i klasserna?

e) Simulera ett system som ser ut på följande sätt:

Låt ankomstintensiteten till den första kön vara 8 per sekund. Ändra inte medelbetjäningstiderna som finns i klassen QS? Vad blir medelantal kunder i var och en av kösystemen?

Denna uppgift kan göras genom att bara ändra i MainSimulation.java.

f) Vi ska studera lastdelning, vilket används i många sammanhang, t ex webbservers, datanät och inom logistik. Det finns ett antal kösystem och en fördelare som helt enkelt ska fördela jobb mellan köerna. Fördelaren väljer vilket kösystem som en ankommande kund ska skickas till. Se figuren:



Dispatchern kan använda någon av följande algoritmer för att bestämma till vilket kösystem som den ska skicka ett nytt jobb:

i. Väljer ett kösystem på slump.
ii. Den väljer system1, sedan 2, sedan 3, sedan 4, sedan5, sedan 1 igen osv.
iii. Den väljer systemet med minst antal jobb (om fler system har minst antal jobb så bryr vi oss inte om vilket som väljs). Vi antar att dispatchern alltid kommer åt antalet kunder i kösystemen.

Simulera systemet och avgör vilken algoritm som ger det minsta totala antalet kunder i alla kösystemen. Ändra inte medelbetjäningstiderna i kösystemen och låt ankomstintensiteten till dispatchern vara 45 per sekund.

Gör följande uppgift i mån av tid:

g) Det är ju inte speciellt realistiskt att dispatchern alltid har tillgång till antalet kunder som finns i de fem kösystemen utan fördröjning när man använder algoritm iii ovan. Antag att varje kösystem en gång per sekund skickar antalet kunder som den har till dispatchern. Dessa data används av dispatchern tills den får nya data efter en sekund. Inför möjligheten att göra detta (ytterligare en signal behövs). Hur bra blir nu algoritm *iii)*?