Profesor: Lara, Luis

|  |  |
| --- | --- |
| **Alumno** | **Legajo** |
| Castaño, Alexis |  |
| Giordano, Nicolas | 40467 |
| Peretti, Leonardo | 40234 |
| Vigil, Carla |  |

Simulacion

Cola simple m/m/1

**Índice**

Codigo Fuente ………………………………………………… 2

Gráficas ………………………………………………..………… 6

Concluciones…………………………………………………… 8

**Código Fuente**

**package** mm1;

**import** java.io.BufferedWriter;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.text.DecimalFormat;

**import** java.util.ArrayList;

**public** **class** ColaSimple {

//Inicio el VTA (vector tiempo de arribos)

**static** ArrayList<Double> *vector\_tiempos\_arribos* = **new** ArrayList<Double>();

//Defino constante el tiempo de servicio y el tiempo en el que se produce un arribo

**private** **static** **final** **double** ***TIEMPO\_ENTRE\_ARRIBOS*** = 0.5;

**private** **final** **static** **double** ***TIEMPO\_SERVICIO*** = 1;

/\* inicializamos el reloj de la simulacion\*/

**static** **double** *reloj* = 0.0;

/\*iniciamos las variables de estado \*/

**static** **boolean** *estado\_servidor* = **false**;//false: disponible, true: ocupado

**static** **int** *nro\_clientes\_cola* = 0;

**static** **int** *nro\_clientes\_atendidos* = 0;

**static** **double** *tiempo\_ultimo\_evento* = 0.0; //TUE

/\*iniciamos los contadores estadisticos \*/

**static** **double** *demoras\_acumuladas* = 0.0; // Ad

**static** ArrayList<Double> *array\_dd* = **new** ArrayList<Double>();

**static** ArrayList<Integer> *array\_cli\_at* = **new** ArrayList<Integer>();

**static** **double** *area\_cli\_cola* = 0.0; // Aq

**static** ArrayList<Double> *array\_dq* = **new** ArrayList<Double>();

**static** ArrayList<Double> *array\_tiempo\_dq* = **new** ArrayList<Double>();

**static** **double** *area\_uso\_servidor* = 0.0; // Ab

**static** ArrayList<Double> *array\_db* = **new** ArrayList<Double>();

**static** ArrayList<Double> *array\_tiempo\_db* = **new** ArrayList<Double>();

**static** **double** *tiempo\_servidor\_comienza\_ocupado* = 0.0;

/\* iniciamos la lista de eventos \*/

**static** ArrayList<Evento> *lista\_eventos* = **new** ArrayList<Evento>();

**public** **static** **void** main(String[] args) {

**double** tiempo\_arribo = *reloj* + *exp*(***TIEMPO\_ENTRE\_ARRIBOS***);

//Genero el primer evento que será un arribo y lo registro en el LEV

Evento primerEvento = **new** Evento("Arribo", tiempo\_arribo);

*lista\_eventos*.add(primerEvento);

*vector\_tiempos\_arribos*.add(tiempo\_arribo);

//Genero la partida del arribo anterior, el tiempo de esta partida es infinito.

Evento segundoEvento = **new** Evento("Partida", Math.*pow*(2, 32));

*lista\_eventos*.add(segundoEvento);

**while**(*reloj* < 30.0)

{

String tipo\_evento = *tiempos*();

**if**(tipo\_evento.equals("Arribo"))

{

*arribo*();

}

**else**

{

*partida*();

}

}

*reporte*();

}

**private** **static** **void** reporte() {

System.***out***.println("LISTA DE EVENTOS(LEV): CANTIDAD DE EVENTOS: "+*lista\_eventos*.size());

**for**(Evento lista: *lista\_eventos*)

{

System.***out***.println("-----------------");

System.***out***.println(lista.getTipo\_evento()+" tiempo: "+**new** DecimalFormat("#.####").format(lista.getTiempo()));

}

System.***out***.println("-------------------------------------------");

System.***out***.println("Vector Tiempos Arribos(VTA):");

**for**(Double d: *vector\_tiempos\_arribos*)

{

System.***out***.println(" --------- ");

System.***out***.println(" "+**new** DecimalFormat("#.####").format(d));

}

System.***out***.println("Cantidad de arribos:"+ *vector\_tiempos\_arribos*.size());

System.***out***.println("Demora Promedio: "+*demoras\_acumuladas*/*nro\_clientes\_atendidos*+"\n "+"uso promedio del servidor: "+*area\_uso\_servidor*/*tiempo\_ultimo\_evento*+" \n"+" Tamaño promedio de cola: "+ *area\_cli\_cola*/*tiempo\_ultimo\_evento*);

**try** {

String ubicacion = "C:\\Users\\nicolas\\desktop\\Simulacion\\";

BufferedWriter out = **new** BufferedWriter(**new** FileWriter(ubicacion+"demoras\_acumuladas.txt"));

**for** (**int** i=0;i<*array\_dd*.size();i++)

{

out.write(**new** DecimalFormat("#.####").format(*array\_dd*.get(i))+ " \t "+*array\_cli\_at*.get(i)+" \n");

}

out.close();

out = **new** BufferedWriter(**new** FileWriter(ubicacion+"area\_uso\_servidor.txt"));

**for** (**int** i=0;i<*array\_db*.size();i++)

{

out.write(**new** DecimalFormat("#.####").format(*array\_db*.get(i))+ " \t"+**new** DecimalFormat("#.####").format(*array\_tiempo\_db*.get(i))+" \n");

}

out.close();

out = **new** BufferedWriter(**new** FileWriter(ubicacion+"area\_cli\_cola.txt"));

**for** (**int** i=0;i<*array\_dq*.size();i++)

{

out.write(**new** DecimalFormat("#.####").format(*array\_dq*.get(i))+ " \t"+**new** DecimalFormat("#.####").format(*array\_tiempo\_dq*.get(i))+" \n");

}

out.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**private** **static** String tiempos()

{

**double** menorTiempo = Math.*pow*(2, 32);

String tipo\_evento = **null**;

**for**(Evento d: *lista\_eventos*)

{

**if**(d.getTiempo()<menorTiempo && *reloj*<d.getTiempo())

{

menorTiempo = d.getTiempo();

tipo\_evento = d.getTipo\_evento();

}

}

*reloj* = menorTiempo;

**return** tipo\_evento;

}

**private** **static** **void** arribo() {

**if**(*estado\_servidor* == **true**)

{

*area\_cli\_cola* += (*nro\_clientes\_cola* \* (*reloj* - *tiempo\_ultimo\_evento*));//tiempo - tiempo\_ultimo\_evento --> tiempo transcurrido desde el último evento hasta ahora

*array\_dq*.add(*area\_cli\_cola*);

*array\_tiempo\_dq*.add(*reloj*);

*nro\_clientes\_cola* ++;

*vector\_tiempos\_arribos*.add(*reloj*);

}

**else**

{

*estado\_servidor* = **true**;

*tiempo\_servidor\_comienza\_ocupado* = *reloj*;

*nro\_clientes\_atendidos* = *nro\_clientes\_atendidos* + 1;

*array\_cli\_at*.add(*nro\_clientes\_atendidos*);

*array\_dd*.add(*demoras\_acumuladas*);

**double** nuevo\_tiempo\_servicio = *exp*(***TIEMPO\_SERVICIO***);

*lista\_eventos*.add(**new** Evento("Partida", (*reloj* + nuevo\_tiempo\_servicio)));

}

**double** proximo\_arribo = *exp*(***TIEMPO\_ENTRE\_ARRIBOS***);

*lista\_eventos*.add(**new** Evento("Arribo", (*reloj* + proximo\_arribo)));

*tiempo\_ultimo\_evento* = *reloj*;

}

**private** **static** **void** partida() {

**if**(*nro\_clientes\_cola*==0)

{

*estado\_servidor* = **false**;

*lista\_eventos*.add(**new** Evento("Partida", *exp*(Math.*pow*(2, 32))));

}

**else**

{

*area\_cli\_cola* = *area\_cli\_cola*+(*nro\_clientes\_cola* \* (*reloj* - *tiempo\_ultimo\_evento*));

*array\_dq*.add(*area\_cli\_cola*);

*array\_tiempo\_dq*.add(*reloj*);

*area\_uso\_servidor* += (*reloj* - *tiempo\_servidor\_comienza\_ocupado*);

*array\_db*.add(*area\_uso\_servidor*);

*array\_tiempo\_db*.add(*reloj*);

*nro\_clientes\_cola* --;

**int** ultimoTiempo = (*vector\_tiempos\_arribos*.size()-1);

*demoras\_acumuladas* = *demoras\_acumuladas* + (*reloj* - *vector\_tiempos\_arribos*.get(ultimoTiempo));

*array\_dd*.add(*demoras\_acumuladas*);

*nro\_clientes\_atendidos* ++;

*array\_cli\_at*.add(*nro\_clientes\_atendidos*);

**double** nuevo\_tiempo\_servicio = *exp*(***TIEMPO\_SERVICIO***);

*lista\_eventos*.add(**new** Evento("Partida", (*reloj* + nuevo\_tiempo\_servicio)));

}

*tiempo\_ultimo\_evento* = *reloj*;

}

**public** **static** **double** exp(**double** media) {

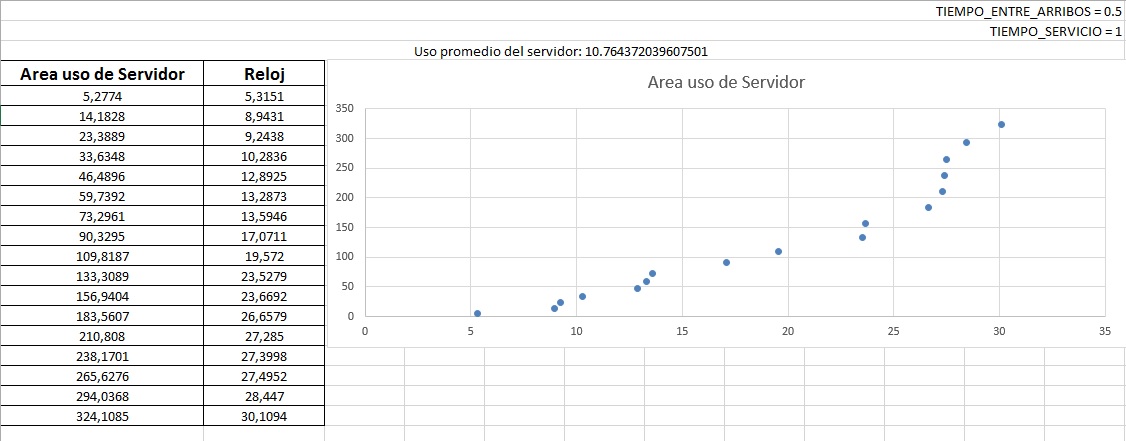
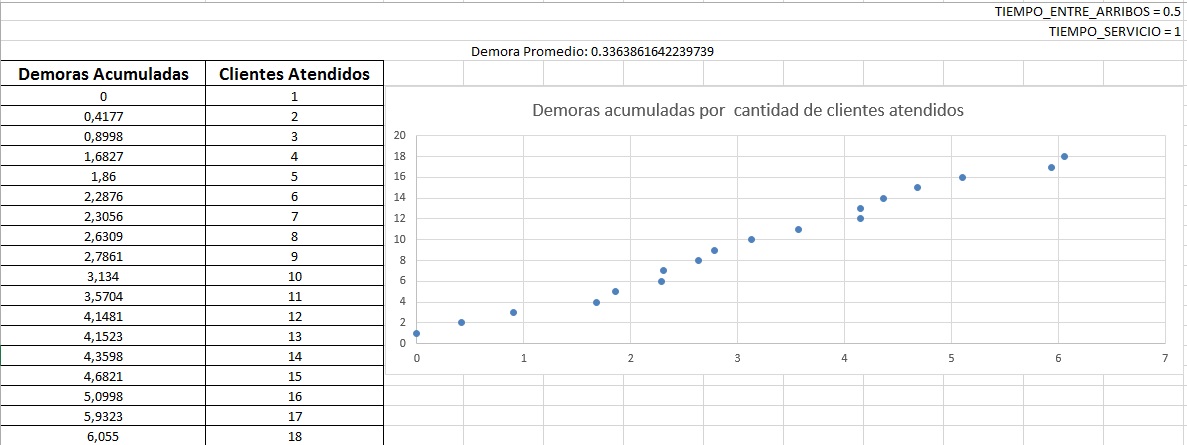
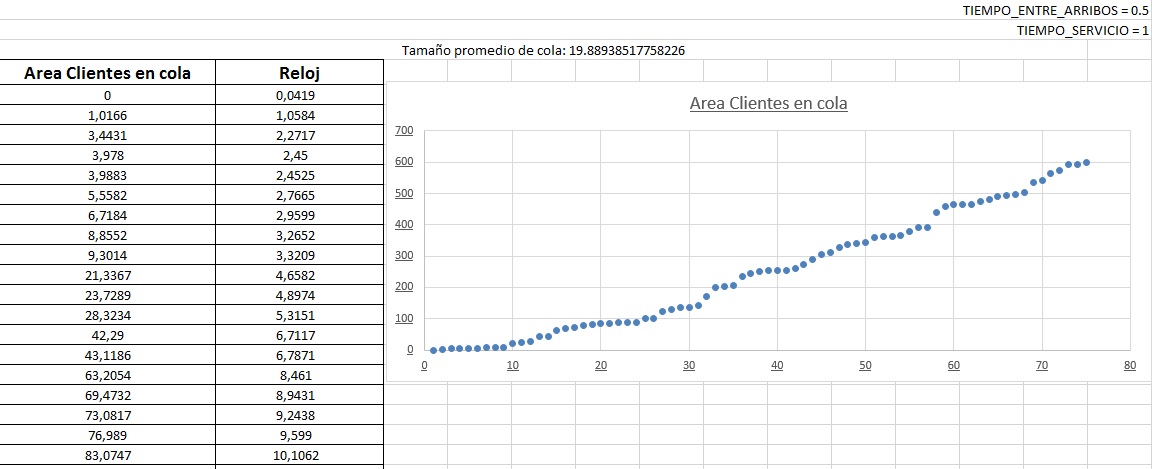
**return** Math.*log*(1 - Math.*random*()) / (-(1 / media));

}

}

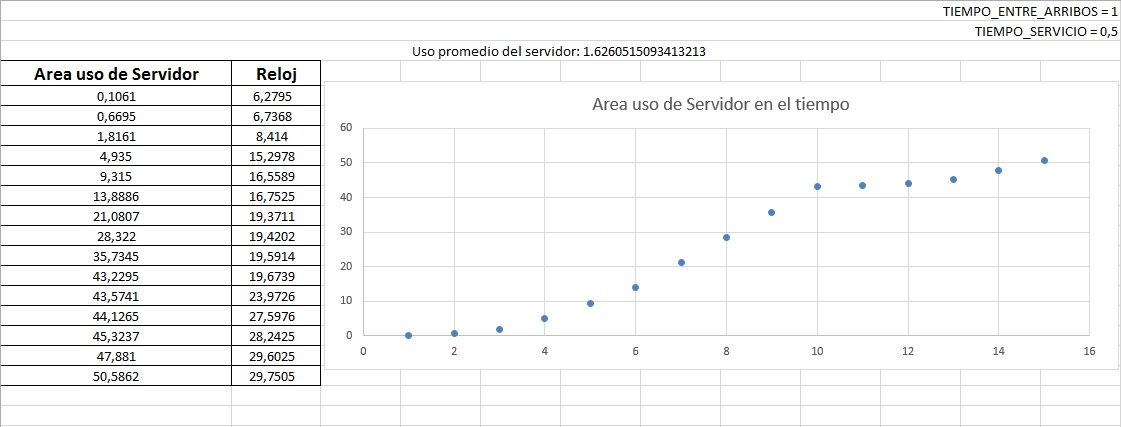
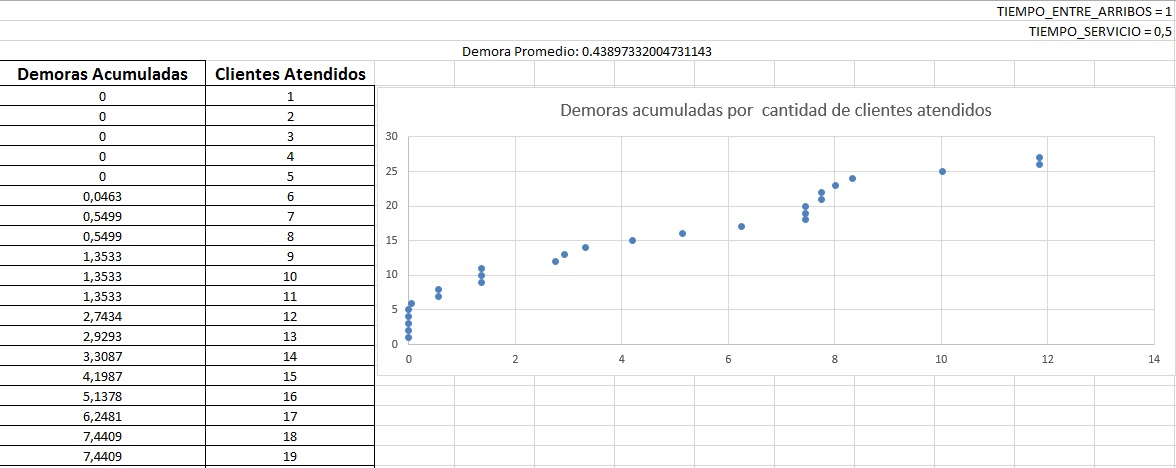
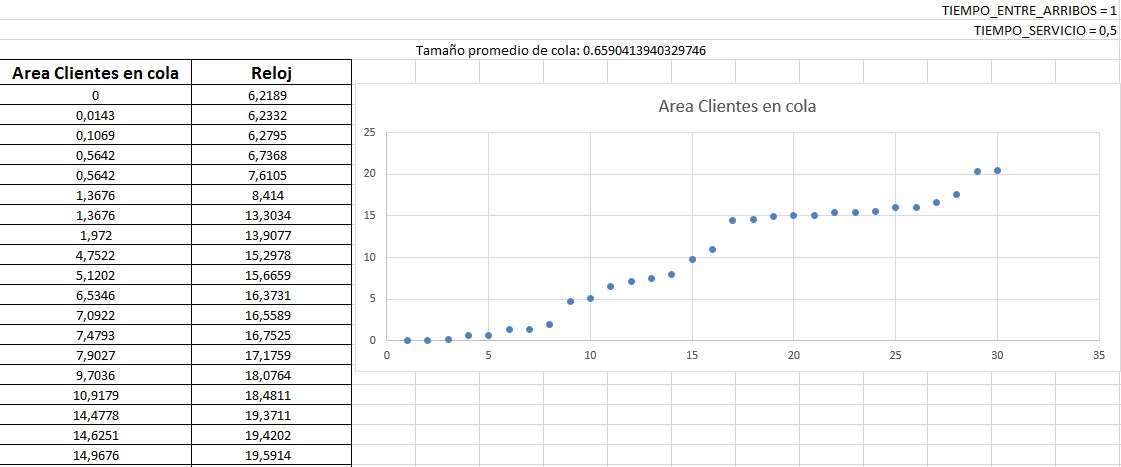
**Gráficas**

Tiempo Entre Arribos = 0.5;

Tiempo de Servicio = 1.0;

Tiempo Entre Arribos = 1.0;

Tiempo de Servicio = 0.5;



**Concluciones**

Cuando el coeficiente entre arribos (lambda) es mayor que el tiempo de servicio(mu)

No se acumula mucha cola, el servidor esta mucho tiempo desocupado y genera tiempo ocioso.

Las demoras no son tan frecuentes, y si ocurren, después generan tiempo ocioso

Cuando el coeficiente entre arribos es menor que el tiempo de servicio ocurre lo contrario

Se genera cola mas largas, el servidor casi que no esta desocupado nunca, y las demoras incrementan con el paso del tiempo