Progreso

Reporte de Proyecto de simulación de Transporte de Autobuses

[[Cronologia del Trabajo Realizado]]

1 : Definicion del Problema :

->La Identificacion de las problematicas y/o inconformidades en el servicio de los autubuses “Arteaga” : Largas esperas y falta de asientos techados

->El analisis de las quejas de los usuarios en horas pico y temporadas con clima inclemente

->Objetivos y Plan del proyecto :

->Objetivo : Desarrollar un programa de simulacion en Java para evaluar solcuciones y respuestas

->Preguntas clave a responder : Reduccion del tiempo de espera, alternativas para dismunuir dicho tiempo y su efecto en la saturacion de asientos con techo

->Distrubucion de tareas entre los miembros del equipo para la recoleccion de datos, validacion, anailsisi y desarrollo del modelo

2 : Conceptualizacion del Modelo :

->Abstraccion de caracteristicas clave del problema y su entorno

->Definicion de entidades(clientes, autobuses), atributos, actividades , recursos y eventos relevantes.

->Creacion de un diagrama del modelo para visualizar la interaccion entre las entidades y actividades

3 : Recoleccion de datos :

->La Obtencion de informacion sobre la llegada de clientes y la puntualidad de los autobuses

->Analisis de datos para modelar la llegada de clientes(distrubucion exponencial) y puntualidad de los autubuses(distrubucion normal)

->Identificacion de parametros fundamentales para la simulacion : tasa de llegada de clientes, tiempos de espera y desviacion estandar de llegada de autobuses.

3 : Desarrollo del codigo en Java :

->Impelentacion del esqueleto del programa de la simulacion en Java

->Definicion de parametros de simulacion(duracion, frecuencia actual y nueva de autobuses)

->Generacion de llegadas de clientes y autobuses según distrubuciones establecidad

->Desarrollo de logica para atender a los clientes y registrar estadisticas clave(tiempo de espera promedio, maximo de clientes esperando)

[[Decisiones Tomadas y Logros Alcanzados]]

: Decisiones :

->Uso de distrubuciones exponenciales y normales para modelar la llegada de clientes y autobuses

->Establecimiento de una duracion de simulacion de 2 horas(120 minutos) para capturar patrones a lo largo del tiempo

-> de una logica para atender a los clientes según la disponibilidad de autobuses

: Logros :

->Creacion de un codigo funcional que simula la interaccion entre llegadas de clientes y autobuses

->Registro inicial de estadisticas como tiempo de espera promedio y maximo de clientes esperando

->Integracion exitosa de generadores de llegadas basados en distrubuciones definidad

[[Cambios Realizados y Puntos Importantes]]

: Cambios Realizados :

->Ajuste de la logica para considerar la nueva frecuancia de autobuses duranta la simulacion

->Incorporacion de la actualizacion de la frecuencia a mitdad de la simulacion para reflejar cambios propuestos en tiempo real.

->Con base a nuestro primer codigo a lo terminado se ha realizado lo siguiente :

[[[Codigo Previo]]]

```
package simulacion;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Random;

public class Simulacion {

   public static void main(String[] args) {
      // Parámetros de la simulación
      int duracionSimulacion = 120; // Duración de la simulación en minutos
      int frecuenciaActual = 20; // Frecuencia actual de los autobuses en minutos
      int nuevaFrecuencia = 15; // Nueva frecuencia de los autobuses en minutos

      // Inicializar la simulación
      Queue<Double> llegadasClientes = generarLlegadasClientes();
      Queue<Double> llegadasAutobuses = generarLlegadasAutobuses();

      int clientesEsperando = 0;

      // Simulación
      for (int tiempo = 0; tiempo < duracionSimulacion; tiempo++) {
         // Verificar si llega un cliente
         if (!llegadasClientes.isEmpty() && llegadasClientes.peek() <= tiempo) {
            llegadasClientes.poll();
            clientesEsperando++;
         }

         // Verificar si llega un autobús
         if (tiempo % frecuenciaActual == 0) {
            // Cambiar a la nueva frecuencia cuando corresponda
            if (tiempo >= duracionSimulacion / 2) {
               frecuenciaActual = nuevaFrecuencia;
            }
            // Atender a los clientes
            int clientesAtendidos = Math.min(clientesEsperando, 8);
            clientesEsperando -= clientesAtendidos;
            System.out.println("Tiempo: " + tiempo + " - Llegó un autobús. Atendidos: " +
clientesAtendidos +
                  ", Esperando: " + clientesEsperando);
         }
      }
   }

   // Generador de llegadas de clientes con distribución exponencial
```

```
    private static Queue<Double> generarLlegadasClientes() {
        Queue<Double> llegadas = new LinkedList<>();
        Random random = new Random();
        double lambda = 0.5; // Tasa promedio de llegada de clientes por minuto

        double tiempo = 0;
        while (tiempo < 120) { // Simular durante 2 horas
            double llegada = -Math.log(1 - random.nextDouble()) / lambda;
            tiempo += llegada;
            llegadas.add(tiempo);
        }
        return llegadas;
    }

    // Generador de llegadas de autobuses con distribución normal
    private static Queue<Double> generarLlegadasAutobuses() {
        Queue<Double> llegadas = new LinkedList<>();
        Random random = new Random();
        double media = 4; // Media de llegada de autobuses en minutos
        double desviacionEstandar = 1; // Desviación estándar de llegada de autobuses en minutos

        double tiempo = 0;
        while (tiempo < 120) { // Simular durante 2 horas
            double llegada = Math.max(0, random.nextGaussian() * desviacionEstandar + media);
            tiempo += llegada;
            llegadas.add(tiempo);
        }
        return llegadas;
    }
}
```

[[[Nuevo Codigo]]]

```
package simulacion;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Random;

public class Simulacion {

    private static double tiempoTotalEspera = 0;
    private static int clientesAtendidosTotal = 0;
    private static int maxClientesEsperando = 0;


    public static void main(String[] args) {
        // Parámetros de la simulación
```

```
        int duracionSimulacion = 120; // Duración de la simulación en minutos
        int frecuenciaActual = 20; // Frecuencia actual de los autobuses en minutos
        int nuevaFrecuencia = 15; // Nueva frecuencia de los autobuses en minutos

        // Inicializar la simulación
        Queue<Double> llegadasClientes = generarLlegadasClientes();
        Queue<Double> llegadasAutobuses = generarLlegadasAutobuses();

        int clientesEsperando = 0;

        // Simulación
        for (int tiempo = 0; tiempo < duracionSimulacion; tiempo++) {
            // Verificar si llega un cliente
            if (!llegadasClientes.isEmpty() && llegadasClientes.peek() <= tiempo) {
                llegadasClientes.poll();
                clientesEsperando++;
            }

            // Verificar si llega un autobús
            if (tiempo % frecuenciaActual == 0) {
                // Cambiar a la nueva frecuencia cuando corresponda
                if (tiempo >= duracionSimulacion / 2) {
                    frecuenciaActual = nuevaFrecuencia;
                }
                // Atender a los clientes
                int clientesAtendidos = Math.min(clientesEsperando, 8);
                clientesEsperando -= clientesAtendidos;
                // Llamada a la función de atención a los clientes
                atenderClientes(tiempo, clientesAtendidos, clientesEsperando);

                System.out.println("Tiempo: " + tiempo + " - Llegó un autobús. Atendidos: " +
clientesAtendidos +
                        ", Esperando: " + clientesEsperando);
            }
        }
    }

    private static void atenderClientes(int tiempo, int clientesAtendidos, int clientesEsperando) {
        // Determinar el tiempo de espera promedio de los clientes atendidos en esta ronda
        if (clientesAtendidos > 0) {
            double tiempoEsperaPromedio = tiempoTotalEspera / clientesAtendidosTotal;
            System.out.println("Tiempo de espera promedios: " + tiempoEsperaPromedio);
        }

        // Actualizar estadísticas globales
        tiempoTotalEspera += clientesEsperando;
        clientesAtendidosTotal += clientesAtendidos;
```

```
            // Actualizar el número máximo de clientes que esperan
            if (clientesEsperando > maxClientesEsperando) {
                maxClientesEsperando = clientesEsperando;
            }
        }

        // Generador de llegadas de clientes con distribución exponencial
        private static Queue<Double> generarLlegadasClientes() {
            Queue<Double> llegadas = new LinkedList<>();
            Random random = new Random();
            double lambda = 0.5; // Tasa promedio de llegada de clientes por minuto

            double tiempo = 0;
            while (tiempo < 120) { // Simular durante 2 horas
                double llegada = -Math.log(1 - random.nextDouble()) / lambda;
                tiempo += llegada;
                llegadas.add(tiempo);
            }
            return llegadas;
        }

        // Generador de llegadas de autobuses con distribución normal
        private static Queue<Double> generarLlegadasAutobuses() {
            Queue<Double> llegadas = new LinkedList<>();
            Random random = new Random();
            double media = 4; // Media de llegada de autobuses en minutos
            double desviacionEstandar = 1; // Desviación estándar de llegada de autobuses en minutos

            double tiempo = 0;
            while (tiempo < 120) { // Simular durante 2 horas
                double llegada = Math.max(0, random.nextGaussian() * desviacionEstandar + media);
                tiempo += llegada;
                llegadas.add(tiempo);
            }
            return llegadas;
        }
    }
```

Se han agregado funciones para registrar estadísticas sobre el tiempo de espera de los clientes y el número máximo de clientes esperando en la parada durante la simulación. Aquí están los cambios detallados:

Nuevas Variables Estáticas:

tiempoTotalEspera: Almacena la suma acumulada del tiempo que los clientes han esperado.

clientesAtendidosTotal: Contabiliza el total de clientes atendidos durante la simulación.

maxClientesEsperando: Registra el número máximo de clientes esperando en la parada en un momento dado.

Nueva Función atenderClientes:

Recibe los parámetros tiempo, clientesAtendidos, y clientesEsperando.

Calcula el tiempo de espera promedio de los clientes atendidos en la ronda actual.

Actualiza las estadísticas globales (tiempoTotalEspera, clientesAtendidosTotal, maxClientesEsperando).

Llamada a atenderClientes:

Dentro del bucle principal, después de atender a los clientes, se llama a la función atenderClientes para actualizar las estadísticas y mostrar el tiempo de espera promedio.

En resumen, la diferencia clave entre los dos códigos radica en que el segundo realiza un seguimiento y registro de estadísticas importantes sobre el tiempo de espera de los clientes y la capacidad máxima de la parada en un momento dado, lo que proporciona una visión más detallada del rendimiento del sistema durante la simulación.

: Puntos Importantes :

->Validacion constante de la logica de simulacion para representar fielmente el comportamiento del sistema real.

->Monitorizacion y analisis continuo de las estadisticas generadas para evaluar la eficacia de las soluciones propuesras