Grado en Ingeniería Informática

Curso 2019/2020 –Convocatoria Ordinaria

Hellín Asensio, Carlos Javier *
Ferreiro Rodríguez, Daniel **

*) lab Profesora Ana Castillo
**) lab 15:00 a 17:00. Profesor Antonio García Cabot

1) Análisis de alto nivel.

Esta práctica se pretende modelar el funcionamiento de un parque acuático. Consta de 2 partes:
1ra parte: Programación Concurrente. 2da parte Programación Distribuida.

Parte1: Programación Concurrente.

Requisitos generales

1-) El parque dispone de una capacidad de máxima de 100 usuarios.
2-) El parque está dividido por zonas: Vestuarios, Piscina de Olas, Piscina de Niños, Piscina Grande, Tumbonas y Toboganes (A,B,C).
3-) El número de usuarios que deben ser atendidos son 5000.
3.1-) Los usuarios tienen una edad entre 1 y 50 años.
3.2-) Los usuarios con edades de entre 1 a 17 años serán niños.
3.2.1) Los usuarios de 1 a 10 años necesita un acompañante mayor de 18 años.
3.3) Los usuarios dispondrán de un ID para su fácil identificación. Estará concatenada la palabra ID junto con un identificador único y su edad. Si es un niño tendrá el ID tendrá concantenado además el identificador de su acompañante, y este el identificador del niño al que cuida.
4-) Habrá 8 monitores, uno por cada zona, se encargarán de comprobar las edades de los usuarios, la capacidad de la zona y ver si se encuentran acompañados para permitirles el acceso o no.
5-) El usuario podrá realizar entre 5 a 15 actividades en el parque.
6-) El usuario debe acceder al vestuario cuando ingresa y termina de hacer sus actividades programadas en el parque. Este acceso es obligatorio tanto para entrar como para salir del parque.
7-) Habrá una cola de espera de tipo FIFO tanto para entrar al parque como en cada una de las actividades.
8-) El acompañante estará dentro de cada zona el mismo tiempo que el niño al que acompaña. (Excepto en la actividad de Piscina de Niños, ya esperará en una cola a que su niño termine.)

Requisitos particulares de cada zona

La zona de los Vestuarios:

- Capacidad: 30 usuarios: 20 adultos y 10 niños.
- Restricción de edad: Ninguna.
- Un acompañante consume una plaza de las disponibles para los niños.

La zona de la Piscina de Olas:

- Capacidad: 20 usuarios.
- Restricción de edad: Los niños de 1 a 5 años no pueden acceder. Los niños de 6 a 10 años deben ir con su acompañante. El resto de usuarios pueden entrar a la actividad sin problema siempre y cuando consigan pareja.
- El acceso a la piscina tiene que ser por parejas. Los usuarios que no tengan acompañantes deben esperar a otro que tampoco tenga. Los niños con sus acompañantes entran a disfrutar la piscina cuando sea su turno y haya capacidad.
- El monitor comprueba que los usuarios se encuentren en parejas y haya hueco.

La zona de la Piscina de Niños:

- Capacidad: 15 usuarios.
- Restricción de edad: Los niños de 1 a 5 años acceden con sus acompañantes. Los usuarios mayores de 11 años no pueden usar esta piscina.Los acompañantes de los niños de 6 a 10 años irán una cola a esperar a que termine la actividad su niño. Los usuarios mayores de 11 años no pueden entrar.

La zona de la Piscina Grande:

- Capacidad: 50 usuarios.
- Restricción de edad: Ninguna.
- Esta tiene 2 accesos: 1 por los toboganes y otra por la piscina.
- No se admitirán nuevos usuarios ni por la piscina ni por los toboganes mientras la capacidad sea 50 (máximo).
- El monitor puede sacar aleatoriamente un usuario de la piscina si esta se encuentra llena para que entre otro.

Toboganes:

- Hay 3 toboganes: A, B y C.
- Hay 1 monitor en cada uno de los toboganes.
- Restricción de edad: Los menores de 11 años no podrán acceder a los toboganes.
  División:
- Tobogán A: para niños de 11 a 14 años.
- Tobogán B: para niños de 15 a 17 años.
- Tobogán C: para usuarios mayores de 18 años.

Tumbonas:

- Capacidad: 20 usuarios.
- Restricción de edad. Mayores de 15 años pueden acceder.
- Al quedar un sitio libre los usuarios compiten por una tumbona y se la termina quedando el más rápido.

Tiempos Usuarios:

Los usuarios se crearán con un intervalo de 400 a 700 milisegundos.
Los usuarios permanecen en el vestuario unos 3 segundos.
Un usuario está en la piscina de olas entre 2 y 5 segundos.
Cada usuario tarda entre 2 y 3 segundos en tirarse por el tobogán.
El usuario permanece en la Piscina de Niño entre 1 a 3 segundos.
Un usuario permanece en una tumbona entre 2 y 4 segundos.
Los usuarios pueden tardar entre 3 y 5 segundos en la Piscina Grande.

Tiempos Monitores:

El monitor tarda 1 segundo en comprobar si se encuentran en parejas los usuarios que quieran entrar a Piscina de Olas.
El monitor tarda entre 1 y 1.5 segundos en comprobar las edades de los usuarios en la Piscina de Olas.
El monitor tarda 1 segundo en comprobar la edad de los usuarios que quieran ir a los Vestuarios.
El monitor de la Piscina Grande tarda 0.5 segundos en comprobar si el usuario puede entrar.
El monitor de la Piscina Grande consume entre 0.5 y 1 segundos en sacar de la piscina a un usuario, y a su acompañante si tuviera, de forma aleatoria de la piscina.
El monitor tardarán entre 400 y 500 milisegundos en comprobar las edades de cada uno de los usuarios que quieran entrar a su respectivo tobogán.
El monitor de las Tumbonas tardará entre 500 a 900 milisegundos en comprobar las edades de los usuarios que quieran acceder a su zona.

Funcionamiento

Una vez que el programa se ejecute los monitores estarán listos para recibir a los usuarios en sus respectivos puestos. Se dispone de una cola de espera en el exterior del parque y en cada una de las actividades disponibles.
Los usuarios van accediendo poco a poco al parque y entran obligatoriamente por el vestuario. Estos esperarán mientras el aforo en el interior de la actividad sea máximo o el monitor se encuentre chekeando.
Cada usuario puede llegar a usar de 5 a 15 actividades, al salir de cada zona elegirá una actividad de forma aleatoria de todas las disponibles pero no sabe si cumple con los requisitos para disfrutar de ella o no. El monitor tiene el trabajo de validar la edad de cada uno de los usuarios que quieran entrar a su respectiva zona. Si un usuario eligió una actividad no disponible para él, el monitor no le dejará pasar por lo que no se le restará 1 al número de sus actividades restantes y escogerá otra sin afectar este valor. Al quedarse sin actividades por hacer el usuario pasará por el vestuario y terminará.
El tiempo de ejecución de la práctica estará fuertemente ligada al tiempo que tarden los hilos en realizar cada una de sus funciones dentro de cada zona y el número de actividades que “decidan” hacer.

Parte2: Programación Distribuida

Está pensada para que un cliente consulte una serie de datos interesantes del comportamiento del parque.

Requisitos generales:

Implementar Sockets o RMI. (Solución escogida: Sockets)
Establecer 4 tipos de controles para un fácil seguimiento de la simulación:
Control de Ubicación: A partir de un ID mostrar la ubicación, el número de actividades y el ID completo del usuario que se pregunta. Si este usuario preguntado no se encuentra en ese momento dentro del parque se mostrará un mensaje que así lo especifique.
Control de Menores: Mostrar el número de menores que han pasado por el parque hasta el momento.
Control de Toboganes: Mostrar el número de usuarios que han accedido a cada unos de los toboganes.
Control de Aforo: Consultar el número de usuarios que se encuentran en ese momento en cada una de las zonas.
Cada módulo de Control tiene un botón “Buscar” que al presionarlo pedirá la información al servidor y la mostrará en el módulo en cuestión.
Requisito añadido: El servidor permitirá el servicio simultáneo a 10 clientes.

Funcionamiento

En el interior de la interfaz Control será nuestro Servidor y Remoto nuestro Cliente. Para un correcto funcionamiento de la aplicación Control deberá estar ejecutada antes que Remoto. Si se hace en diferente orden dará un error ya que el cliente intentará hacer una conexión con un servidor inexistente por el puerto 5000.
Al presionar cada botón “Buscar” el cliente pedirá la información correspondiente al servidor por medio de un identificador.
El servidor recibirá la etiqueta y pedirá a las clases pertinentes los datos solicitados para concatenarlos en un string separados por comas. El cliente que pidió la información la toma y la muestra en la interfaz.

2) Diseño y las herramientas de sincronización utilizados.

Hilos: usuarios, monitores y la atención que reciben los clientes al conectarse con el Servidor.

Las colas de entrada y salida de cada unas de las zonas son ArrayList porque es una estructura que no necesita especificar su capacidad y posee métodos optimizados como el size(), add(), remove(), isEmpty().

Herencias:
La clase Persona.java es padre de las clases Monitor.java y Usuarios.java.
La clase Zona.java es padre de las clases de actividades: PiscinaOlas.java, Tobogan.java, PiscinaNinos.java, Tumbonas.java, PiscinaGrande.java, Vestuario.java.

Mecanismos utilizados en la **sincronización** y asegurarnos un correcto funcionamiento:
Semáforos con contador para poder controlar el aforo de las actividades. Ejemplo: la cantidad de niños permitidos en el vestuario debe ser 10.
Semáforos justos para las colas de espera en las diferentes zonas. El orden de llegada de los usuarios es importante.
Semáforo binario para permitir el acceso por pareja a la Piscina de Olas de aquellos usuarios que no tuvieran acompañantes.
Métodos synchronized se usaron con el objetivo de garantizarnos la exclusión mutua. Ejemplo en la clase Lista.java solo se podía meter o sacar un usuario a la vez en una misma cola evitando así incongruencias en los datos y conflictos.

Tipos de Sincronización observados:
Cooperación: Los hilos que se encuentran en la cola del parque tiene que esperar que finalicen los usuarios que se encuentra dentro, si el aforo es máximo, para continuar su ejecución.
Competición: Una vez que queda una tumbona libre y los usuarios que se encuentran en la cola compiten por acceder al recurso. Lo tomará si cumple con los requisitos de la edad y es el más rápido.

3) Las clases principales que intervienen con su descripción.

Este proyecto consta de 3 paquetes:

1. **Paquete interfaz**

Tiene 4 clases (Cliente,Control, Remoto y Servidor). Control y Remoto son las clases encargadas de mostrar 2 ventanas con información referente a la simulación del parque y al ejecutarlas saldrá un servidor y un cliente respectivamente.

Clase Control.java

Para una fácil manipulación y simplicidad de los constructores de las clases se ha creado varios array de JTextFields que se le pasarán a las clases intermedias.
La clase Lista.java imprimirá en el JTextField correspondiente la lista de usuarios que se encuentran en las colas de espera y usando cada una de las actividades.
Ejemplo:
JTextField [] tFColasAfuera = {tfColaParque,tfColaVestuarios,tfColaPiscinaOlas, tfColaPiscinaNinos,tfColaTumbonas,tfPiscinaGrande,tfColaToboganes};

Clase Remoto.java

El módulo de acceso remoto está dividido por varios controles (ubicación, toboganes, aforo y num Menores).Para separar la interfaz del manejo y escritura de los datos esta clase tiene similitud con la forma de operar de la anterior clase anterior ya que devolverá un array de JTextFields dependiendo del módulo que se necesite.

Clase Servidor.java

El servidor aceptará la conexión de los clientes que se conecten por el localhost y el puerto 5000. Brindará un servicio a un máximo de 10 clientes simultáneamente.
Creará un hilo para cada cliente que se conecte y estará “escuchando” todo el rato mientras tengo conexión con este.
void tratarMensaje(Socket conexion, String mensaje);
Funcionalidad: Verá que control le ha mandado la petición y le devolverá un array con la información solicitada.

Clase Cliente.java

Será la encargada de pedir la información al servidor del módulo de control de acuerdo al botón “Buscar” presionado.
String [] devolverMensajeArray(String mensaje);
Funcionalidad: Como parámetro se recibirá una cadena que contendrá unos datos separados por comas y este método se encargará de retornar un array ocupando en cada posición un dato y eliminará el carácter separador.

void recibirMensaje(String mensaje);
Funcionalidad: Recibirá una cadena, identificará a que control pertenece los datos, por el identificador que está en la primera posición del array, y se le pasará al método escribirDatosControl() un array con datos y un array de los JtextField para que muestre la información por la interfaz.

void enviarMensaje(int tipoControl);
Funcionalidad:Envía un int que significará el tipo de control que se ha pedido la información al presionar el botón sdadas“Buscar” en la interfaz.

void escribirDatosControl(JTextField [] tf, String [] datos);
Funcionalidad: Escribirá los datos en el JTextField correspondiente del control.

2. Paquete simulacion:

Persona.java

Será el padre de los usuarios y los monitores. Tendrá 2 parámetros: identificador y la edad.

int getIdentificador();

Funcionalidad:Retornará el id de esta persona.

int getEdad();

Funcionalidad: Devolverá la edad de la persona.

Monitor.java

Hereda de la clase Persona. Su principal función es dejar pasar a los usuarios que cumplan los requisitos de los actividades: edad o acceso en pareja. Se encuentra comprobando todo el tiempo comprobando la edad de los usuarios y se duerme luego unos 16 milisegundos para no sobrecargar la CPU.

Su constructor recibe 2 parámetros:

1. id: un número del 1 al 8 que será asignado en dependencia de la actividad que realice.
2. una zona en donde efectuará su "trabajo".

Usuario.java

Hereda de Persona. En esta clase se crea un usuario y se le asigna un acompañante mayor de edad si este lo necesita. Entra y saldrá por el vestuario. Dispondrá de entre 5 a 15 actividades.

String mostrarIdentificador();

Funcionalidad: Mostrará el identificador del usuario: (ID3-19) concatenando el identificador y la edad. Si requiere acompañante, dicho identificador se concatenará con el id y edad del niño (ID3-9-4, ID4-23-3).

boolean esAdulto();

Funcionalidad: Devolverá true si la edad del usuario es >= 18

boolean necesitaAcompanante();

Funcionaliad: Devolverá un boolean a true si la edad del usuario está en el rango de 1 a 10 años.

Actividad elegirActividad();

Funcionalidad: Devolverá un valor aleatorio que representa una actividad de las disponibles.

setPareja(Usuario pareja);

Funcionalidad: Le añade una pareja a ese usuario.

Zona.java

Clase en donde se crea cada unos los monitores de las diferentes actividades y se inserta o se saca a los usuarios de las colas de entrada y dentro de una actividad.

void extraer(Usuario usuario, Usuario acompanante);

Funcionalidad: Extraerá tanto al usuario como al acompañante de la colaEntrada de la actividad.

void insertar(Usuario usuario, Usuario acompanante);

Funcionalidad: Insertará en la cola de dentro al usuario y también a su acompañante si es que tiene.

void entrar(Usuario usuario, Semaphore semaforo, int permisos);

Funcionalidad: Insertará en la cola de entrada al usuario y a su acompañante si la actividad no es la Piscina de Niños. Una vez que estén dentro del parque se sacará al usuario y a su

acompañante de la cola de espera y se insertará en la cola de dentro. Si la actividad no es la piscina de Olas los usuarios se paran para que el monitor pueda comprobar la edad.

void salir(Usuario usuario, Semaphore semaforo, int permisos);
Funcionalidad: Extraer al usuario como a su acompañante de la cola de dentro de cada actividad

int getCapacidad();
Funcionalidad: Devolverá la capacidad de la zona en cuestión.

Parque.java
Hereda de la clase Zona. Se instancia la clase Vestuario,lugar por donde el usuario entrará y saldrá, y el conjunto de actividades que ofrece el parque acuático.
void entrar(Usuario usuario);
Funcionalidad: Entrar a el usuario por el Vestuario.

void salir(Usuario usuario);
Funcionalidad: Sale el usuario del Vestuario.

void realizarActividad(Usuario usuario, Actividad actividad);
Funcionalidad: Entra el usuario a una actividad.

Paso.java
Define un cerrojo con un Condition para la variable booleana cerrado que es comprobada por un proceso. Si vale false(abierto) el proceso puede continuar. Si es true(cerrado) el proceso se detiene.

Simulador.java
Hilo encargado de crear los hilos usuarios.
void generarUsuarios();
Funcionalidad: Crea y se inicializan 5000 usuarios. En el total de usuarios se contabilizará los acompañantes de los niños que lo necesitan. Cada se usuario se creará dentro de un rango de 400 a 700 milisengundos.

Lista.java
Tiene un constructor en donde se creará un arrayList de Usuarios y varios métodos para gestionarla.
synchronized void imprimir();
Funcionalidad: Se encargará de recorrer el arrayList y concatenar en un String todos los identificadores de los usuarios que se encuentren en la lista y se lo pasará al JTextField para que se muestre en la ventana de la aplicación.

synchronized void insertar(Usuario usuario);
Funcionalidad: Insertará en el arrayList el usuario que entre por parámetro al igual que a su acompañante si dispusiera de él. Llama al método imprimir().

synchronized void extraer(Usuario usuario);
Funcionalidad: Elimina al usuario que entra como argumento y a su acompañante si tuviera. Llama al método imprimir();

Utils.java
Dispone de varios métodos que devolvieran números aleatorios tanto para las edades como para el tiempo a dormir los hilos ya que su uso está intensificado por todo el programa.
static int azar(int a, int b);
Funcionalidad: Devuelve un entero N al azar al que a <= N <= b.
static void dormir_ms(int milisegundos);
Funcionalidad: Hace dormir al hilo en cuestión el número de milisegundos pasados por argumento.

3. Paquete simulacion.actividades:

Todas estas clases heredan de Zona.java

PiscinaGrande.java
Usuario comprobarEdad();
Funcionalidad: El comportamiento de este método es similar al de su padre. Extraerá de la cola de entrada al usuario y al acompañante, lo introducirá en la cola de dentro y dormirá 0.5 segundos. Si el aforo de la piscina es mayor o igual a la capacidad permitida sacará a un usuario aleatorio de los que se encuentra dentro. Devuelve un usuario.

PiscinaNinos.java
Usuario comprobarEdad();
Funcionalidad:No deja entrar a los usuarios que tengan más de 11 años. Permitirá el acceso a los niños de 1 a 5 años junto con sus acompañantes. Si el niño está en el rango de 6 a 10 años el acompañante se inserta en una lista de espera de adultos. Allí estará hasta que finalice su niño. Tardará entre 1 a 1.5 dentro de la actividad. Retornará el usuario.

void salir(Usuario usuario);
Funcionalidad: Comprobará la edad del usuario que entra por parámetro y si este está en el rango de 6 a 10 años se extraerá a su acompañante de la cola de espera destina para los adultos.

PiscinaOlas.java
Usuario comprobarEdad();
Funcionalidad: No dejará entrar a los niños de 1 a 5 años, lo sacará de la cola de espera. Si el niño dispone de acompañante se sacarán de la cola espera y entrarán al interior de la actividad. Si ha podido conseguir pareja el usuario los dos entrarán en la actividad. Tardarán 1 segundo.

void esperarAcompanante(Usuario usuario);
Funcionalidad: Si el usuario no tiene acompañante y aún no tiene pareja esperará a conseguirla. Una vez conseguida se liberará de la espera y entrarán.

Tumbonas.java
Si hay una tumbona libre, los usuarios que se encuentren en la cola competirán por ese recurso, en caso contrario, el monitor dejará entrar al primero de la cola de espera.

Usuario comprobarEdad();

Funcionalidad: Si la edad del usuario es menor que 15, serán sacados de la cola de espera tanto él como su acompañante. Si son mayores que 15 saldrán de la cola de esperas y entrarán a la cola de adentro. Tardará entre 500 a 900 milisegundos.

salir(Usuario usuario);

Funcionalidad: Si un usuario sale de la actividad queda una tumbona libre y será una comparación entre la cantidad de tumbonas ocupadas y el aforo de la zona. Si sale mayor entonces se liberarán a los usuarios que están esperando para que compitan por la tumbona.

Tobogan.java

Usuario comprobarEdad();

Funcionalidad: Retorna un Usuario. Se preguntará por la edad y el acompañante del usuario que está con el monitor. Se sacará/n de la cola de espera y se dejará pasar si cumple con la edad de este tobogán. Abarca las edad de 11 a 14, 15 a 17 y mayores de 18. Se esperará un tiempo entre 400 a 500 milisegundos.

boolean accion(Usuario usuario);

Funcionalidad: Si ha podido entrar al tobogán hará el usuario duerma entre 2 a 3 segundos. Retornará true si ha podido realizar la entrada.