Cámara Argentina de Comercio y Servicios
INSTITUTO PRIVADO

Examen Final - Programación II

Profesor: Jonathan Pepe

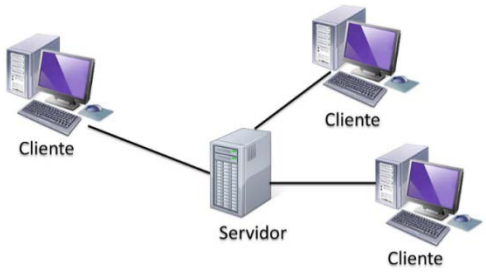
Fecha: Comisión/Turno: Apellido y Nombre:

1	2	3

Calificación

Aprobado	Insuficiente

Arquitectura Cliente – Servidor



Introducción (*Fuente Wikipedia*)

La *arquitectura Cliente - Servidor* es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados *Servidores*, y los demandantes, llamados *Clientes*.

Un *Cliente* realiza *peticiones* a otro programa, el *Servidor*, quien le da respuesta.

Algunos ejemplos de aplicaciones computacionales que usen el modelo *Cliente - Servidor* son el Correo electrónico, un Servidor de impresión y la World Wide Web.

En la arquitectura *C/S* el remitente de una *solicitud* es conocido como *Cliente*. Sus características son:

- Es quien inicia *solicitudes* o *peticiones*.
- Espera y recibe las respuestas del *Servidor*.
- Por lo general, puede conectarse a varios *Servidores* a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Al receptor de la solicitud enviada por el *Cliente* se conoce como *Servidor*. Sus características son:

- Al iniciarse esperan a que lleguen las *solicitudes* de los *Clientes*.
- Tras la recepción de una *solicitud*, la procesan y luego envían la respuesta al *Cliente*.
- Por lo general, acepta las conexiones de un gran número de *Clientes*.

Ejercicio 1 (Obligatorio)

Una máquina utilizada como *Servidor* recibe *peticiones* de las máquinas *Cliente* conectadas a él. Dichas *peticiones* se dividen en dos categorías: *críticas* y *no críticas*.

Cada *petición* está identificada por un *ID unívoco* y posee una duración estimada [*ciclos de reloj*] que necesita para ser procesada en su totalidad. Además, las *peticiones críticas* cuentan con un número de prioridad ϵ [1; 10], donde 1 la prioridad más alta.

El *Servidor* obedece la siguiente política: primero atiende las *peticiones críticas* según su prioridad y, luego, las *no críticas* según el orden de arribo.

Especificar e implementar el *TDA* que gestione las *peticiones* que arriban al ***Servidor***. La clase ***Servidor*** debe proveer métodos que permitan:

1. Recibir nuevas *peticiones*, tanto *críticas* como *no críticas*.

2. Informar cuántas *peticiones críticas* quedan por atender, conjuntamente con el tiempo (*ciclos de reloj*) necesario para procesarlas a todas.

3. Ídem anterior, pero para las *peticiones no críticas*.

4. Conocer qué *petición* será la próxima en ser atendida.

5. Atender las *peticiones* durante un tiempo dado, medido en *ciclos de reloj*, y respetando la política antes mencionada.

   La *petición* podrá ser atendida de forma completa o, podría suceder que no sea posible esto último. En dicho caso, la *petición* no será atendida de forma parcial (pues una vez iniciado el proceso no puede interrumpirse) y se procederá a buscar la siguiente *petición* que pueda ser atendida en su totalidad en el tiempo que resta.

(*Para este ejercicio, no es necesario implementar el TDA Cliente, sólo el Servidor quien recibe peticiones*)

Ejercicio 2 (Opcional)

Cada máquina *Cliente* debe enviarle, en primera instancia, una *solicitud* al *Servidor* para poder establecer una conexión de aquí en más con el mismo. La *solicitud* requerirá el ingreso de una *password de acceso* que el *Servidor* debió establecer oportunamente.

En este contexto, una máquina *Cliente* sólo puede estar vinculada con una máquina *Servidor*, pero un *Servidor* puede prestar servicios a varias máquinas *Cliente*.

Cuando una *petición* enviada por un *Cliente* sea procesada exitosamente por el *Servidor*, será responsabilidad del *Servidor* enviarle un '*aviso*' a la máquina correspondiente indicando este evento. *(¿Qué implica esto último?)*

Especificar e implementar el *TDA* ***Cliente***. La clase ***Cliente*** debe proveer métodos que permitan:

1. Establecer conexión con un *Servidor*.
2. Enviarle al *Servidor* las peticiones que fueran necesarias, tanto *críticas* como *no críticas*.
3. Recibir el '*aviso*' de que la *petición* fue procesada exitosamente por el *Servidor* e imprimir por pantalla "*Petición [ID] procesada de forma exitosa*".
4. *Desvincularse* del Servidor actual.

Consideraciones

En ambos casos se deben realizar los *diagramas de clases* propios del UML y ofrecer una especie de simulación que permita probar y verificar el correcto funcionamiento de esta *arquitectura Cliente – Servidor*.

Utilice C++ como lenguaje para implementar los *TDA*. Se debe indicar, obligatoriamente, pre y post condiciones de cada método, argumentos recibidos y tipo de retorno.

Conjuntamente con el código del programa, deberá hacer entrega de un *informe* que explicite la estrategia de resolución llevada a cabo y aclare todos los supuestos bajo los cuales trabaja. El informe deberá incluir los *diagramas de clases* UML.

El trabajo es individual, aunque esto no impide que pueda consultar con sus compañeros y debatir diferentes propuestas de diseño e implementación. **La entrega del mismo deberá ser, a más tardar, 48 horas previas a rendir el examen final de la materia**. La entrega será vía *e-mail* adjuntando tanto los archivos fuentes como el informe en un .rar rotulado "*FINAL_PROGRAMACION2_APELLIDO_NOMBRE*". Además, en el *main* debe estar comentado su apellido, nombre y DNI.

La *calificación final* de la materia surgirá del *promedio ponderado* entre la nota cuatrimestral y la nota del examen final, donde esta última resultará de la suma entre:

a. Calificación ejercicio 1 -*obligatorio*- (***4 puntos***).
b. Calificación ejercicio 2 -*opcional*- (***3 puntos***).
c. Calificación examen final -*presencial*- (***3 puntos***).

Como su nombre lo indica, el *trabajo obligatorio* es una condición necesaria para presentarse a rendir el examen final. Para realizar el *trabajo opcional*, previamente debe compilar y cumplir con todas las funcionalidades el *trabajo obligatorio*. Deberá hacer una defensa de los trabajos presentados.

El *examen presencial* será modalidad *teórico – conceptual* sobre los temas abordados en la materia.

En caso de detectarse plagio en el código, el examen será desaprobado sin oportunidad de recuperarlo. Sea creativo y original. Todo aporte que mejore el trabajo sin disminuir su complejidad, será altamente valorado.

Traer impreso este documento conjuntamente con el informe al momento de presentarse a rendir el examen final. Los trabajos que no cumplan los requisitos indicados no serán aceptados.

Firma: ……