**Universidad de los Andes**
Ingeniería de Sistemas y Computación

ISIS 1205 - Algorítmica y Programación 2

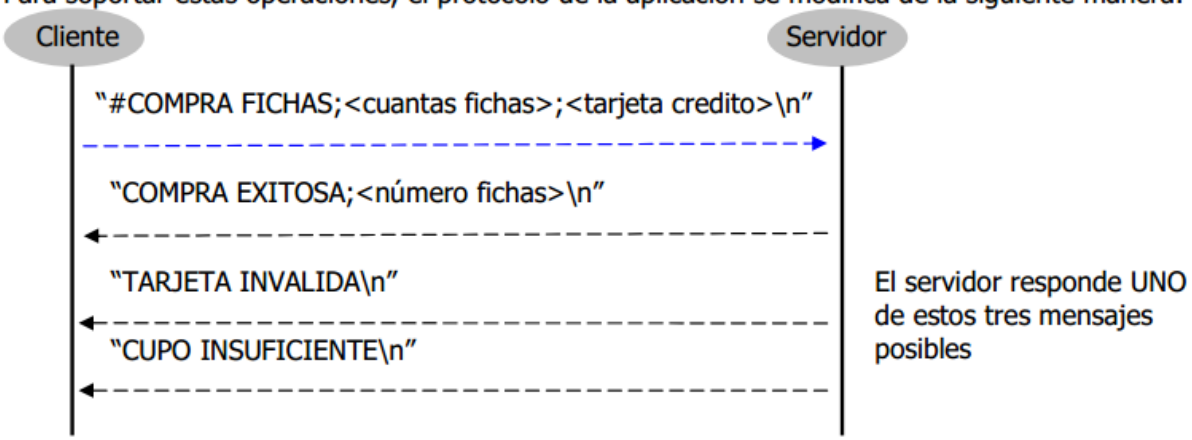
Taller teórico nivel 12

**Cupi2**

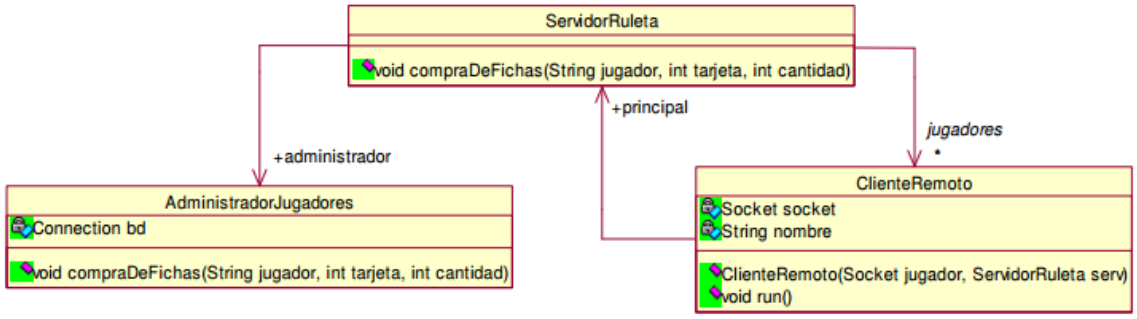
Una vez desarrollada la parte básica del Servidor de Ruleta, se desea completar la aplicación, tanto para el cliente, como para el servidor, para que implemente el manejo de tarjeta de crédito Casino Cupi2. Las reglas de manejo son:

- El identificador de las tarjetas es un entero positivo de 4 dígitos
- Las tarjetas Casino Cupi2 funcionan en un esquema de prepago.
- Son adquiridas y recargadas por fuera de la aplicación
- El cupo inicial y de recarga es único, con valor de 1000 fichas
- La tarjeta Casino Cupi2 es personal e intransferible
- Un jugador tiene derecho a tener una única tarjeta
- Cada compra de fichas disminuye el cupo disponible de la tarjeta en esa misma cantidad
- La aplicación rechaza cualquier intento de compra de fichas por fuera de su cupo

Para soportar estas operaciones, el protocolo de la aplicación se modifica de la siguiente manera:




El modelo siguiente muestra los aspectos más relevantes para satisfacer esta modificación



## 1. [40%] Manejo de Bases de Datos desde Java – JDBC

Suponga que la tabla de jugadores ya ha sido creada, con los siguientes campos.

| TablaJugadores | | | | | | |
|---|---|---|---|---|---|---|
| Nombre | FichasGanadas | FichasPerdidas | FichasJugadas | FichasDisponibles | Tarjeta | CupoDisponible |
| varChar | Int | Int | Int | Int | Int | Int |

En la clase `AdministradorJugadores`, que maneja la base de datos, complete el método que registra la operación de compra de fichas por parte de un cliente (el servidor las vende). Debe cumplir el siguiente contrato:

```
/** Actualiza la información del jugador que quiere comprar una cierta cantidad de fichas.
  Pre: El jugador existe en la Base de Datos
  Post: El cupoDisponible para el jugador en la base de datos disminuyó en cantidad de fichas; las
   fichas disponibles del jugador aumentan en la cantidad comprada.
  @param nomJugador - Cadena con el nombre del jugador -
  @param tarjeta - El número de la tarjeta Casino Cupi2 del jugador - Mayor que cero
  @param cantidad - La cantidad de fichas solicitadas por el jugador - Mayor que cero
  @throws TarjetaInvalidaException - Cuando la tarjeta no corresponde a la que el jugador tiene
   registrada
  @throws CupoInsuficienteException - Cuando la cantidad solicitada es mayor que el cupoDisponible
*/
public void compraDeFichas (String nomJugador, int tarjeta, int cantidad) throws
   TarjetaInvalidaException, CupoInsuficienteException
```

## 2. [60%] Programación Cliente/Servidor

Complete el método run de la clase `ClienteRemoto`, que se encarga de la comunicación con el cliente, de manera que responda al protocolo descrito. Desarrolle **UNICAMENTE LO RELATIVO A LA COMPRA DE FICHAS**

Suponga que la clase `ServidorRuleta` ya está implementada y que, en particular, ya existe el método compraDeFichas, cuya implementación es:

```
public class ServidorRuleta
{
   …
   public void compraDeFichas (String nomJugador, int tarjeta, int cantidad) throws
   TarjetaInvalidaException, CupoInsuficienteException
   {
        administrador.compraDeFichas (nomJugador, tarjeta, cantidad);
   }
   …
}
```

```
PUNTO 1 - 40%
public class AdministradorJugadores
{  …
   private Connection bd;
   public void compraDeFichas (String nomJugador, int tarjeta, int cantidad) throws
   TarjetaInvalidaException, CupoInsuficienteException
   {
      // Traer la información del jugador (12%)
      // Verificar la tarjeta (8%)
      // Verificar el cupo (8%)
      // Si todo va bien, actualizar el cupo disponible (12%)
```

```
   }
}
```

```
PUNTO 2 - 60%
public class ClienteRemoto extends Thread
{  private Socket socket; // El canal de comunicación con el jugador remoto
   private ServidorRuleta principal;    // La referencia a la clase principal del servidor
   private String nombre;               // El nombre del jugador

   public ClienteRemoto (Socket canal, ServidorRuleta servidor)
   {   // YA IMPLEMENTADO
   }

   public void run ()
   {   boolean fin = false;
       // Abrir los flujos de entrada y salida de comunicación con el cliente  (10%)


       while (!fin)
       {
              // Leer mensaje del cliente                                    (5%)
              // Procesar mensaje - UNICAMENTE PARA LA COMPRA DE FICHAS      (20%)
              // Enviar respuesta                                            (20%)

       }
       // cerrar los flujos de datos    (5%)


   }
}
```