Objetivos

El objetivo de este ejercicio es que el estudiante comprenda y adquiera práctica en:

- El desarrollo de aplicaciones siguiendo un proceso incremental
- El manejo de la persistencia de información en un manejador de base de datos
- La comunicación entre programas a través de sockets
- El desarrollo de interfaces gráficas interactivas
- El desarrollo de pruebas unitarias en junit para las clases del ejercicio.

Preparación

1. Localice el archivo n12_demo.zip, decomprímalo y ejecute el programa .exe que muestra una ejecución del programa. Estudie el funcionamiento esperado del programa.
2. Localice y descomprima el archivo esqueleto.zip.
3. Cree el proyecto en eclipse con el contenido del directorio n12_Ruleta
4. Estudie la documentación del problema disponible en el directorio docs/specs del proyecto
   - La descripción del problema en Descripcion.doc.
   - Los requerimientos funcionales en RequerimientosFuncionales.doc
   - Los requerimientos no funcionales en RequerimientosNoFuncionales.doc
   - El modelo del mundo en modeloConceptualCliente.jpg
   - El modelo de la interfaz en interfazCliente.jpg
   - El modelo de pruebas en pruebasCliente.jpg
5. Asegúrese de tener activado el uso de aserciones para la ejecución del programa. Ver el tutorial en http://cupi2.uniandes.edu.co/cursos/apo2/docs/n7_assert.pdf
6. Defina una configuración de ejecución del proyecto que verifique las invariantes de las clases del mundo. Para esto, debe indicar como parámetro a la JVM la opción –ea, tal como se ha especificado en ejercicios anteriores
7. Realice el mismo proceso en cada una de las configuraciones de Junit

Proceso de desarrollo

En este ejercicio usted deberá implementar completamente el servidor para un juego de ruleta distribuido. Dado que hay un protocolo de comunicación definido, cualquier servidor que siga este mismo protocolo podría ser usado para que los clientes que están dados en el esqueleto se comuniquen entre ellos a través del servidor (conectándose a este).

Siga los siguientes pasos para la elaboración de su ejercicio:

1. Diseñe el servidor. Tenga en cuenta los requerimientos y las recomendaciones que se dan más adelante en este enunciado.
2. Diseñe las pruebas que va a construir para verificar la implementación del servidor. No se espera (para la evaluación del ejercicio) que construya pruebas automáticas para todos los aspectos del servidor, pero usted debe implementar las que le sean de mayor utilidad para apoyar la implementación de su proyecto. Revise

las pruebas del cliente (en especial las de la clase ClienteTest) para tener una idea de la complejidad asociada a las pruebas de aplicaciones multi-hilo con sockets.

3. Implemente los esqueletos de las clases del servidor (creación de las clases, declaración de constantes, atributos, signaturas y documentación de cada método).
4. Implemente las pruebas que definió.
5. Implemente el código del servidor.
6. Ejecute las pruebas automáticas.
7. Pruebe el servidor usando los clientes reales.

Tenga en cuenta los siguientes aspectos para la implementación del servidor:

- El servidor debe estar siempre esperando nuevas conexiones por parte de los clientes.
- El servidor tiene que efectuar varias tareas simultáneamente: lo recomendable es que usted tenga un hilo en el cual se reciben las conexiones de los clientes (hilo principal) y un hilo más por cada cliente conectado. Este último hilo se encarga de recibir los mensajes enviados por el cliente (inicio de sesión de juego, compra de fichas, jugar y desconexión).
- El servidor debe almacenar en una base de datos la información de los jugadores Esta base de datos debe ser Derby DB. Es necesario tener una tabla con los datos de los jugadores: Nombre, fichas disponibles, perdidas, ganadas y jugadas.
- El servidor no debe hacer manejo de errores (recuperación en caso de error). Los posibles errores que se pueden presentar son debidos a problemas de comunicación. Por ejemplo, si un cliente interrumpe su ejecución de manera abrupta o si la conexión falla. En este caso, el servidor debe cerrar a sesión de juego y continuar disponible para los otros jugadores
- En la página de cupi2 se encuentra el ejemplo de la Batalla Naval, que podrá servirle de guía para el uso de sockets y de la base de datos.