Parcial 2 Programación Básica 1 - Q1 - C64 - TN - 2023

Trading bots

Generaremos un producto software que permita realizar transacciones de compra y venta entre bots. Para esto, debemos ingresar bots por pantalla y luego simular las transacciones de compra y/o venta.

Clase de prueba TradingBots

Deberemos entonces completar los métodos incluidos en la clase TradingBots para que el software pueda funcionar correctamente.

- **public static void main(String[] args)**: Dado el menú, deberemos completar cada caso:
    - **Ingresar bots**: Ingresar la cantidad de bots, debiendo validar el número ingresado para que esté comprendido entre 3 y 20. Agregar la cantidad de bots que fue ingresada anteriormente al sistema Trading. Para ingresar un bot por pantalla, utilizar el método "ingresarBot()". Una vez generado el bot, se deberá agregar al array de bots de la clase Trading mediante el método agregarBot().
    - **Simular transacciones**: se deberá invocar al método simularTransacciones() de la clase Trading. Mostrar un mensaje indicando que se simularán las transacciones y otro que indique que finalizó la tarea.
    - **Bots ordenados por crypto:** Obtener los bots ordenados consumiendo el método obtenerBotsOrdenamosPorCryptoDescendiente() de la clase Trading y mostrarlos.
    - **Promedio de transacciones de venta:** Obtener el promedio del monto de las transacciones de venta consumiendo el método obtenerPromedioDeTransaccionesDeVenta() de la clase Trading y mostrarlo.
- **private static MenuPrincipal obtenerOpcionDeMenuPrincipal(int numeroIngresado):** Tomar del enum el valor correspondiente a la opción ingresada.
- **private static int ingresarNumeroEntero(String mensaje):** Mostrar un mensaje y solicitar el ingreso del dato.
- **private static String ingresarString(String mensaje)**: Mostrar un mensaje y solicitar el ingreso del dato.

Nombre y Apellido DNI FIrma

- **private static double ingresarNumeroConDecimales(String mensaje):** Mostrar un mensaje y solicitar el ingreso del dato.
- **private static Bot ingresarBot(String mensaje):** Muestra un mensaje que solicita el ingreso de los datos del bot y devuelve una instancia de la clase Bot.

Clase Bot

Un bot está conformado por un nombre, saldo (dinero) y cantidad de crypto (moneda). Según el caso, cada bot sabe decirnos si puede comprar o vender según sea necesario. Es importante poder mostrar los datos de un bot.

- **public boolean puedeVender(double cantidadCrypto):** Indica si el bot tiene la cantidad de crypto solicitada para la venta.
- **public boolean puedeComprar(double cantidadTransaccion, double valorCrypto):** Indica si el saldo es suficiente para comprar la cantidad de crypto al valor suministrado.

Clase Transaccion

Una transacción es una operación de compra o venta entre bots. Cada una de ellas posee un id único, un Tipo de Transaccion (COMPRA o VENTA), una cantidad de crypto comprometida en la misma y el valor, basado en la cantidad de crypto y el precio de 1 crypto (constante y presente en la clase Trading).
Esta clase tiene la posibilidad de asignar un nuevo id a cada transacción en cada construcción, debiendo ser los ids contiguos.

Clase Trading

Será la clase que contenga la lógica del producto software. Posee un array de Bot para almacenar los bots con los cuales se generarán las transacciones, un array de transacciones, donde guardaremos las transacciones simuladas y un valor de crypto (que no cambia por el momento), para realizar los cálculos correspondientes.

Debemos entonces completar los siguientes métodos:

- **public void agregarBot(Bot bot)**: Agrega un bot al array de bots solo si no existe otro bot con el mismo nombre. Para esto se deberá consumir el método

buscarBotPorNombre(). En caso de no existir un bot con ese nombre, entonces se deberá agregar al array de bots.

- **private Bot buscarBotPorNombre(String nombre)**: Dado el nombre que llega como parámetro, se deberá buscar en el array de bots. Si no existe, se devuelve null.
- **public void simularTransacciones()**: Genera transacciones con valores aleatorios. Se debe completar cada posición del array de transacciones. Debemos tener un bot vendedor y otro comprador obtenidos desde el array de bots aleatoriamente, para poder transaccionar. El bot comprador no puede ser el mismo que el vendedor. La cantidad comprometida en la transaccion, debera ser aleatoria consumiendo el método obtenerCantidadCryptoParaTransaccion(). Solo si el vendedor puede vender y el comprador, puede comprar, generamos transacciones. Se deberá generar un transacción para registrar la venta y otra para registrar la compra consumiento el metodo generarTransaccion().
  Importante: En esta primera versión, no actualizaremos los datos de cada bot por cada transacción.
- **private void generarTransaccion(TipoTransaccion tipoTransaccion, double cantidadTransaccion)**: Es necesario saber cual es el monto de la transacción. Una vez instanciada la transacción, es necesario agregarla al array de transacciones.
- **private Bot obtenerBotAleatorio()**: Devuelve un número double aleatorio entre 1 y 5.
- **public Bot[] obtenerBotsOrdenamosPorCryptoDescendiente()**: Ordena los bots de manera descendiente por la cantidad de crypto monedas que tiene.
- **public double obtenerPromedioDeTransaccionesDeVenta()**: Obtiene el promedio del monto de todas las transacciones que sean de venta.

Nombre y Apellido DNI FIrma

Lineamientos

- No es posible modificar la firma de los métodos existentes, pero es posible agregar todos los métodos que se consideren necesarios.
- Se pueden aplicar las mejoras que considere necesarias siempre cumpliento el punto anterior.
- Se deberán completar los métodos y código faltante para que el juego funcione correctamente.
- Deberá ajustar el nombre del proyecto suministrado indicando sus apellidos y nombres en donde dice “ApellidosNombres”: PB12023Q1C64P2TN-**ApellidosNombres.**
- La versión de java del proyecto deberá ser ajustada a la del laboratorio.

Condiciones de aprobación

Se considerará desaprobado todo proyecto que:

a. no compile, o,
b. no se identifiquen los atributos necesarios para resolver lo solicitado, o,
c. los tipos de datos elegidos para los atributos no sean los adecuados, o,
d. no cumpla con las condiciones mínimas descritas a continuación.

Se considerará aprobado aquel examen que cumpla con el 70% de puntaje.

Clase de prueba TradingBots	main()	10%
	obtenerOpcionDeMenuPrincipal(int numeroIngresado) y ingresarNumeroEntero(String mensaje) y ingresarString(String mensaje) y ingresarNumeroConDecimales(String mensaje)	5%
	ingresarBot(String mensaje)	5%
Clase Bot	Constructor, getters y setters necesarios, método para mostrar detalles del bot	5%
	puedeVender(double cantidadCrypto) y puedeComprar(double cantidadTransaccion, double valorCrypto)	5%

Nombre y Apellido DNI FIrma

Clase Transaccion	Constructor, getters y setters necesarios, método obtenerSiguienteId()	5%
Clase Trading	constructor, getters y setters necesarios	5%
	agregarBot(Bot bot)	10%
	buscarBotPorNombre(String nombre)	5%
	simularTransacciones()	15%
	generarTransaccion(TipoTransaccion tipoTransaccion, double cantidadTransaccion)	10%
	obtenerBotAleatorio() y obtenerCantidadCryptoParaTransaccion()	5%
	obtenerBotsOrdenamosPorCryptoDescendiente()	5%
	obtenerPromedioDeTransaccionesDeVenta()	10%
Total		100%

Nombre y Apellido DNI FIrma