# Practica HazelCast II

## Parte 1: Event Listener sobre cache de órdenes

Se implementan los métodos entryAdded y entryUpdated como parte del listener los cuales controlan las entradas de ordenes de mercado que se agregan u alteran.

public class VolumeListener

            implements EntryAddedListener<String, MarketOrder>,

        EntryUpdatedListener<String, MarketOrder>, Serializable {

    private String instrumentoAControlar;

    private int volumenAcumulado=0;

    private HashMap<String, MarketOrder> orderMap;

    public VolumeListener(String instrument) {

        this.instrumentoAControlar=instrument;

        this.orderMap = new HashMap<String, MarketOrder>();

    }

    /\*\*

     \* Escuchar entradas que se añaden y sumarlo al volumen/imprimir alerta si llegamos a 30000

     \* @param entryEvent

     \*/

    @Override

    public void entryAdded(EntryEvent<String, MarketOrder> entryEvent) {

        MarketOrder order = entryEvent.getValue();

        if (order.getInstrument().equals(this.instrumentoAControlar)) {

            this.volumenAcumulado += order.getVolume();

            this.orderMap.put(entryEvent.getKey(), order);

        }

        verifyOrderVolumn();

    }

    /\*\*

     \* Escuchar entradas que se añaden, restar valor antiguo y

     \* sumar el nuevo al volumen/imprimir alerta si llegamos a 30000

     \* @param entryEvent

     \*/

    @Override

    public void entryUpdated(EntryEvent<String, MarketOrder> entryEvent) {

        MarketOrder currentOrder = this.orderMap.get(entryEvent.getKey());

        MarketOrder order = entryEvent.getValue();

        if (order.getInstrument().equals(this.instrumentoAControlar)) {

            this.volumenAcumulado += (order.getVolume() - currentOrder.getVolume());

            this.orderMap.put(entryEvent.getKey(), order);

        }

        verifyOrderVolumn();

    }

    private void verifyOrderVolumn(){

        if (this.volumenAcumulado > 30000){

            System.out.println("\nAlerta: 30000 unidades acumuladas\n");

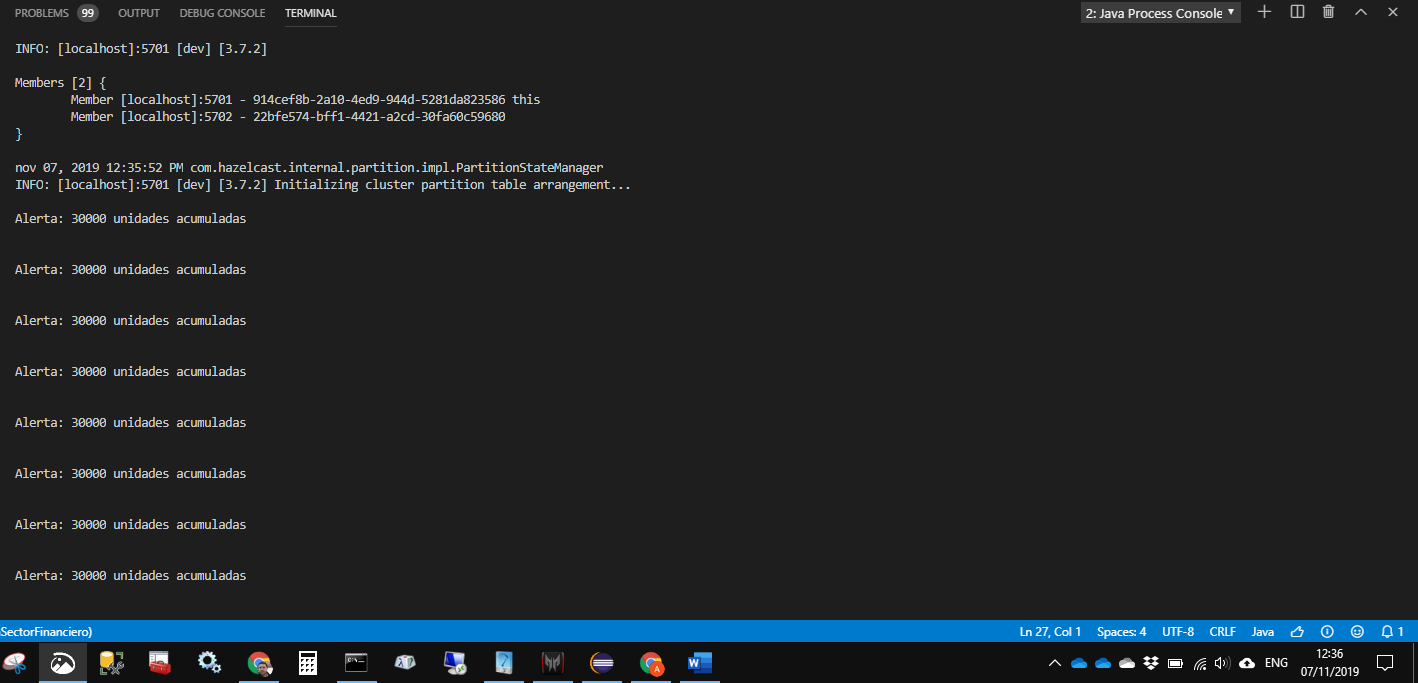
            this.volumenAcumulado = 0;

        }

    }

}

Como resultado en consola y ejecutando inicialmente el cliente en P1VolumenControl y posterior el cliente que agrega las ordenes P1VolumeAdder:



## Parte 2: Búsquedas y modificaciones con EntryProcessor

Se implementa el método process de la clase OrderProcessor el cual retorna el valor actual de la orden y cambia su volumen a 0

public class OrderProcessor

        extends AbstractEntryProcessor<String,MarketOrder> implements Serializable {

    /\*\*

     \*

     \*/

    private static final long serialVersionUID = 1L;

    /\*\*

     \* Metodo que debe procesar cada entrada y cambiar su volumen a 0, y devolver el

     \* antiguo

     \*

     \* @param entry Hazelcast llamará a este metodo para cada entrada de la cache

     \* @return Integer con el volumen que existia

     \*/

    @Override

    public Object process(Map.Entry<String,MarketOrder> entry) {

        MarketOrder order = entry.getValue();

        Integer currentValue = order.getVolume();

        order.setVolume(0);

        entry.setValue(order);

        return currentValue;

    }

}

Como resultado se halla el total del volumen de ordenes y se procesan sus volúmenes a 0

