**Parte 2 del Trabajo 2 (60%) sobre Solidity**

**Enunciado: Tienda “Vanessa Daou”**

**Enunciado**: Sergio Andrés Arboleda Zuluaga.

**Revisión**: Francisco Moreno.

Implementar un contrato inteligente en Solidity para una tienda en la que van a interactuar los siguientes usuarios:

**Dueño**: Se encarga de crear los productos, definir su precio base (unitario), número de unidades en inventario y descripción (ver más adelante). También puede visualizar sus ganancias y otros datos de interés de la tienda (ver más adelante). Solo puede haber un dueño de la tienda (el creador del contrato).

**Cliente**: Puede **comprar** (sin fiar) y **comprar fiado**. Para poder comprar, el cliente primero se debe registrar. Al fiar un producto el cliente contrae una deuda por el valor de la compra, el cliente debe poder consultar el valor de su deuda y pagar **exactamente** este valor para quedar a paz y salvo. Si un cliente tiene deuda no puede comprar más productos hasta que la pague.

**Requisitos**

Cada producto registrado debe tener nombre, número de unidades en inventario, precio base unitario (en ethers, un valor entero mayor que cero) y descripción.

No puede haber dos productos con el mismo nombre.

Solamente el **dueñ**o puede registrar productos.

Un **cliente** se registra con un código (entero positivo), nombre y país de origen. No puede haber dos clientes con el mismo código.

Cuando un cliente va a comprar un producto, el precio **final** de un producto puede cambiar así: si un cliente ha gastado más de 50 ethers en compras de la tienda (esto no incluye el valor de una posible deuda que tenga), se restan 3 ethers del precio base siempre y cuando el resultado de aplicar esta operación no termine en un precio final negativo.

Un **cliente** debe poder consultar (por nombre del producto) el precio **final** que pagaría por un producto si lo fuese a comprar.

Un **cliente** debe poder consultar (por nombre del producto) los datos de un producto.

Un **cliente** solo puede hacer **compras** si está registrado y no tiene deuda (o sea, un fiado pendiente).

La **compra** debe recibir el valor **exacto** (en ethers) del producto a comprar. En una compra, solo se puede comprar **una unidad de un producto a la vez**; la compra requiere que el producto exista y que haya al menos una unidad del mismo. Cuando se hace una compra, se decrementa el número de unidades del producto comprado en una unidad.

La **compra fiada** no recibe dinero (ether), pero registra en el sistema al cliente con la deuda por el valor del producto que compró fiado (al comprar fiado, se decrementa igualmente el número de unidades del producto en una unidad).

Un **cliente** debe poder consultar el valor de su deuda actual y poder pagar este valor (**exactamente el valor)** para quedar a paz y salvo y así poder hacer más compras.

Un **cliente** debe poder consultar el valor total de sus compras (este valor no debe incluir una posible deuda (un fiado) que tenga).

El **dueño** debe poder consultar el valor total de todas las compras (suma del valor de todas las ventas hechas a todos los clientes) y también el valor total de todo el dinero que le deben los clientes (todos los fiados pendientes).

El **dueño** debe poder consultar el valor total de todas las compras hechas por clientes por país, por ejemplo, consultar el valor total de todas las compras hechas por clientes de Colombia.

El **dueño** debe poder destruir el contrato, pero esto requiere un número fijo de intentos (3). Solo al tercer intento se destruye. Para esto se debe apoyar en la función selfdestruct (destruir y enviar todos los fondos al dueño).

**Notas adicionales:**

* Puede usar todas las estructuras de datos que desee.
* **Se garantiza que en las pruebas solo se ingresarán tipos de datos correctos, por ejemplo, en el precio base de un producto solo se ingresarán números enteros positivos correctos.**
* Para entregar por email a *[fjmoreno@unal.edu.co](mailto:fjmoreno@unal.edu.co)*, el martes 11 de mayo **hasta las 6 pm**. Solo se califican trabajos enviados a ese correo.
* **No se reciben trabajos en hora posterior**. No se reciben versiones “mejoradas”. No se califican trabajos enviados “por accidente” a otros correos.
* **Se debe incluir un informe breve donde se describa cómo se solucionó cada punto.** Este informe hace parte de la calificación del trabajo (máximo 2 hojas). **No enviar los datos de prueba que usted usó para probar sus códigos.**
* Grupos de **tres** personas.
* Los trabajos deben ser independientes entre los grupos. Trabajos copiados **así sea en un SOLO punto** se califican con 0 (cero) en su totalidad para todos los integrantes. Las soluciones presentadas deben ser originales. El trabajo debe ser desarrollado por los integrantes del grupo no por personas ajenas a él.
* El monitor les puede ayudar con aspectos técnicos pero su función **no** es hacerles la práctica **ni está autorizado** para **cambiar las condiciones del trabajo**.
* **Si trabaja con otros lenguajes**, así su trabajo funcione y sea “espectacular”, el trabajo **NO** será calificado.
* Cualquier duda consultarla con el profesor o con el monitor.
* En este trabajo el estudiante Sergio Andrés Arboleda Zuluaga (*saarboledaz@unal.edu.co*) les puede también ayudar a resolver algunas dudas técnicas **puntuales** mediante correos. Él está autorizado para aclararles aspectos del enunciado, pero **NO** para cambiar las condiciones del trabajo. Tampoco es responsabilidad de él hacerles parte del trabajo ni prestarles asesoría en horarios específicos. Es un colaborador.
* Solamente se califican las funcionalidades solicitadas, si incluye funcionalidades adicionales, estas **NO** serán tenidas en cuenta para la calificación, sin importar lo espectaculares o útiles que sean.

Abril 22 de 2021