Lab 18-08-2023

Project Title: Decentralized Marketplace with User Roles

Develop a decentralized marketplace smart contract system that allows participants to **register**, **buy**, and **sell** items. The system includes three user roles: **Premium Sellers**, **Regular Buyers**, and **VIP Buyers**, each with unique features and interactions.

1. Base Marketplace Contract:

- Create a **base contract** named `Marketplace`.
- Implement a `**mapping**` named 'balance' that **maps address** to **uint**, to store the balance of each participant's **address**.
- Define an `**event**` named `Purchase` to log details of a purchase. **e.g.** 'buyer address', 'sender address', and 'value of purchase'.
- Write a function `register` to register **address** of a new participant with an initial balance.
    - Send a relevant 'error' message if 'initial balance' is inputted as zero.
- Implement the `buy` function to enable a 'registered' buyer to purchase an item from a 'registered' seller. It will also send 'cash' taken as input in the function, from buyer **address** to seller **address**.
    - Send a relevant 'error' message if either 'buyer' or 'seller' have not been previously registered.
    - Send a relevant 'error' message if 'cash' inputted is zero.
    - Make sure to deduct the 'cash' from the 'buyer's balance' and add it to the 'sender's balance'.
    - At the end of the function, **emit** the 'sender' **address,** 'buyer' **address** and the 'cash' taken as input.
- Develop the `sell` function for a 'registered' seller to transfer an item (item cannot be free) to a 'registered' buyer.
    - For transferring item, you can just return a **string** statement saying that "Item will be shipped to buyer's location".
    - Send a relevant 'error' message if either 'buyer' or 'seller' have not been previously registered.
    - Send a relevant 'error' message if price of the item being sold is inputted as zero.
    - Send a relevant 'error' message if the 'sender' and the 'buyer' are the same person.

2. PremiumSeller Contract:

- Create a **derived contract** named `PremiumSeller` that **inherits** from the `Marketplace` contract.
- Add a **state variable** `fee` to store the fee percentage.
- Write a **constructor** to set the `fee` when creating a `PremiumSeller` instance.
- **Override** the `sell` function from the **base contract** to deduct the fee from the 'registered' seller's balance and add it to the contract's balance.
    - You can add to contract balance using **address(this)** for contract address.
    - Send an 'error' message if 'balance' of the seller is zero.
    - Include the rest of the 'error' statements mentioned in the 'sell()' function in the 'Marketplace' contract using **super** keyword.

3. RegularBuyer Contract:

- Create a derived contract named `RegularBuyer` inheriting from the `Marketplace` contract.
- No additional features are required for `RegularBuyer`.

4. VIPBuyer Contract:

- Develop a **derived contract** named `VIPBuyer` inheriting from the `Marketplace` contract.
- Add a **state variable** `discount` to store the discount percentage.
- Write a **constructor** to set the `discount` when creating a `VIPBuyer` instance.
- Override the `buy` function from the base contract to apply the discount to the purchase amount for VIP Buyers.
    - Send a relevant 'error' message if either 'buyer' or 'sender' **addresses** are not registered.
    - Make sure to subtract the 'discounted' price from the 'buyer's **address**' and add it to the 'seller's **address**'.
    - At the end of the function, **emit** the 'sender' **address,** 'buyer' **address** and the 'discounted' price paid.

Hint:

- To calculate final 'fee' amount in the 'sell()' function in 'PremiumSeller' **contract**:

```
uint feeAmount = ItemPrice * fee / 100;
```

- To calculate the final 'dicounted' price in the 'buy()' function in the 'VIPBuyer' **contract**:

```
uint discountAmount = purchaseAmount * discount / 100;

uint finalAmount = purchaseAmount - discountAmount;
```