**ASSIGNMENT FOR WEEK 4 DAY 3**

**COMPARE AND CONTRAST INTER CONTRACT EXECUTION AND INHERITANCE IN A ONE PAGE REPORT**

Inter-contract execution and inheritance are two distinct features in Solidity, each with its own use cases and benefits. In this report, we will compare and contrast inter-contract execution and inheritance in terms of their definition, advantages, disadvantages, and use cases.

**Inter-Contract Execution:**

Inter-contract execution is the ability for smart contracts to call other contracts and execute their functions. This is made possible by the EVM (Ethereum Virtual Machine) and is one of the key features that enable decentralized applications (dApps) to be built on the Ethereum blockchain.

**Advantages:**

**Encapsulation:** Inter-contract execution allows for better separation of concerns and encapsulation of functionality, enabling developers to create modular and reusable contracts.

**Flexibility:** Inter-contract execution provides flexibility in creating complex applications as it allows for different contracts to interact and communicate with each other.

**Disadvantages:**

**Cost:** Every inter-contract call incurs a transaction fee which can be expensive, especially for complex and frequent interactions between contracts.

**Security:** Inter-contract communication can introduce security vulnerabilities, such as reentrancy attacks, where a malicious contract repeatedly calls a vulnerable contract to drain its funds.

**Use Cases:**

**Token standards**: Inter-contract execution is widely used in the creation and management of token standards like ERC-20, ERC-721, and ERC-1155.

**Composable dApps:** Inter-contract execution is also used in creating composable dApps, where multiple contracts interact to provide a single cohesive user experience.

**Inheritance:**

Inheritance is a key feature of Solidity that allows a contract to inherit properties and behavior from another contract. Inheritance is used to create a parent-child relationship between contracts, where the child contract inherits all the properties and functions of the parent contract.

**Advantages:**

**Reusability:** Inheritance promotes code reusability by allowing developers to create common functionality in a parent contract that can be inherited by multiple child contracts.

**Modularity:** Inheritance enables the creation of modular and extensible contracts that can be easily updated and maintained.

**Disadvantages:**

**Complexity:** Inheritance can introduce complexity in code, especially in cases where multiple levels of inheritance are used.

**Tight Coupling:** Inheritance can lead to tight coupling between parent and child contracts, making it difficult to modify one contract without affecting the other.

**Use Cases:**

**Token standards:** Inheritance is used in the creation of token standards like ERC-20, where child contracts inherit the basic token functionality from the parent contract.

**Library contracts:** Inheritance is also used in creating library contracts that provide reusable functionality for other contracts.

**Conclusion:**

Inter-contract execution and inheritance are two distinct features in Solidity, each with its own benefits and use cases. Inter-contract execution provides flexibility and encapsulation, while inheritance promotes reusability and modularity. Developers should carefully consider the advantages and disadvantages of each feature before deciding on the best approach to use in their smart contracts.

**PART TWO**

* Create a Base Contract and Declare a string variable that would hold a firstName value.
* Declare function that assigns a value to the firstName variable. The function should be able to be overridden by a contract inheriting it.
* Declare a function that retrieves the value of the firstName variable
* Create a Derived Contract and declare an array of strings that would contain firstName variables.
* Declare a function that would override the function in the base contract, to push firstName variables into its array