11/06/2025

WednessDay

**Abstraction:**

 - key concept of OOP

 - Hiding internal implementation details and showing only the essentail features of an object.

 - *\_Why we need it ?\_*

    1. It reduces complexity

    2. to increase code reusability

    3. focus on what an object does instead of how it does it.

- we can acheive abstraction in java using

    1. Abstract Classes

    2. Interfaces

>> 1. Abstract Class

- declared with  with abstract keyword.

- can have both abstract methods(method without body) and concrete methods(method with body)

- cannot be instantiated

>> 2. Interface

- all methods are implicitly abstract and public

- it supports multiple inheritance

- from java 8 onwards we have

            1. default

            2. static method

|  |  |  |
| --- | --- | --- |
| Feature | Abstract class | Interface |
| Methods | Both abstract and concrete | Only abstract( till java7) |
| Inheritance | Single inheritance | Multiple inheritance |
| Constructor | Yes | No |
| Acess modifier | Can use any access modifier | Methos are public by default |



**Encapsulation**

* It is a fundamental concept in OOP.
* It means binding data (variables) and code (methods) together as a single unit.

Why do we need it?

1. We have to protect data from unauthorized access.
2. We have to improve code maintainability and flexibility.
3. To control how important data is accessed and modified.

Implementation in java

1. Declare class variables as private.
2. Provide public getter and setter to access and update their values.

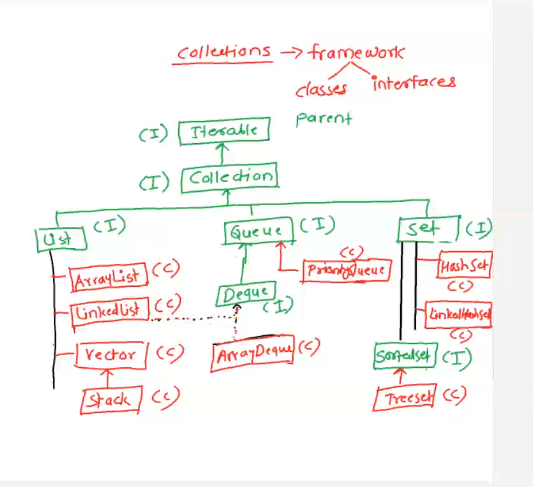
Pojo file : contains only private variables and getter and setter methods

// work

Create a pojo file for teacher class

Id ,name ,salary

**Collections**:



|  |  |  |
| --- | --- | --- |
|  | ArrayList | LinkedList |
| DS | Dynamic | Doubly linked list |
|  | Fast O(1) | Slower O(n) needs to traverse |
| Insertion/deletion(middle) | Slower | Faster O(1) if use iterator |
| (end) | Faster | Fasteer |
| Memory&efficiency | Better than linkedLsit |  |
|  |  |  |

If we want faster access of elements use ArrayList.

When we are dealing with most insertion/deletion at midlle we have to go ahead with linkedLsit.

**Vector:**

* It is similar to ArrayList.
* It is synchronized ( thread -safe);. All methods are synchronized (safe in multi threading
* follows insertion order
* allows duplication
* arraylist is faster than vector

**Set:**

**HashSet:**

* it is a framework class that implements set interface.
* Used to store unique elemets, does not allow dupicates.
* No insertion order.