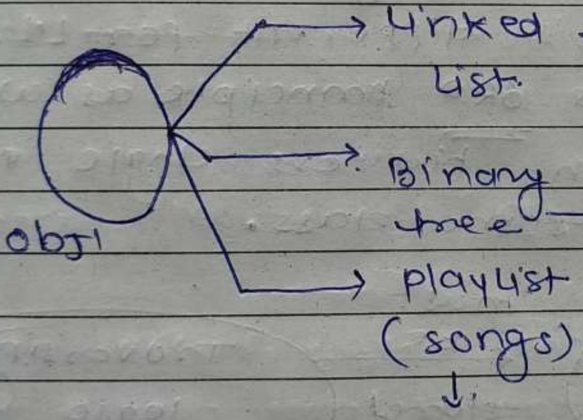
Iterator Design pattern

Standard Defination

Iterator provides a way to access the elements of an aggregate object sequentially without exposing it's underlying Architecture.

- suppose we have an object to traverse through it. traversing is depend on the data structure used to store elements.



Linked List →
```
temp=head;
while(temp!=NULL)
{ temp=temp.next;
}
```

Binary tree →
1) Inorder
2) preorder
3) postorder
4) Levelorder

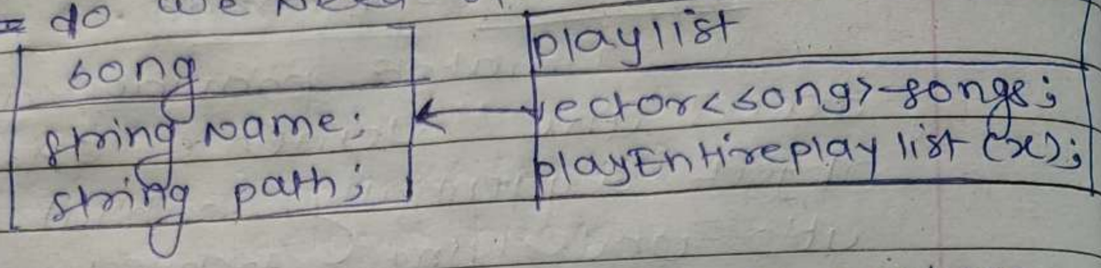
playlist (songs) ↓

List of songs canbe stored using vector (can use forloop to traverse).

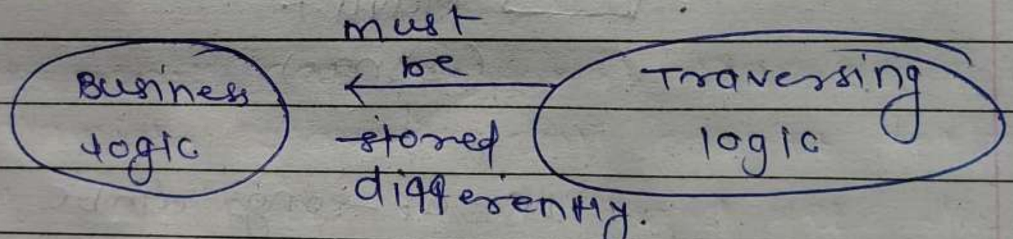
. Along with all these we ~~are~~ have another way of traversing known as "iterator".

↳ list.getIterator → it
- → hasNext() → checks if Next value is there
- → next() → if hasNext() ~~there~~ move to Next value

why ~~[illegible]~~ do we need Iterators ?



∴ Initially we ~~store~~ songs in vector.
so we can iterate using loop. for loop.
– for (songs: song)
song → path.
∴ later, we want to store in Linked list so
we need to ~~[illegible]~~change the traversing
logic as it's different for L.L.
∴ which breaks SRP principle as we are
storing both business logic and traversing
logic in some class.



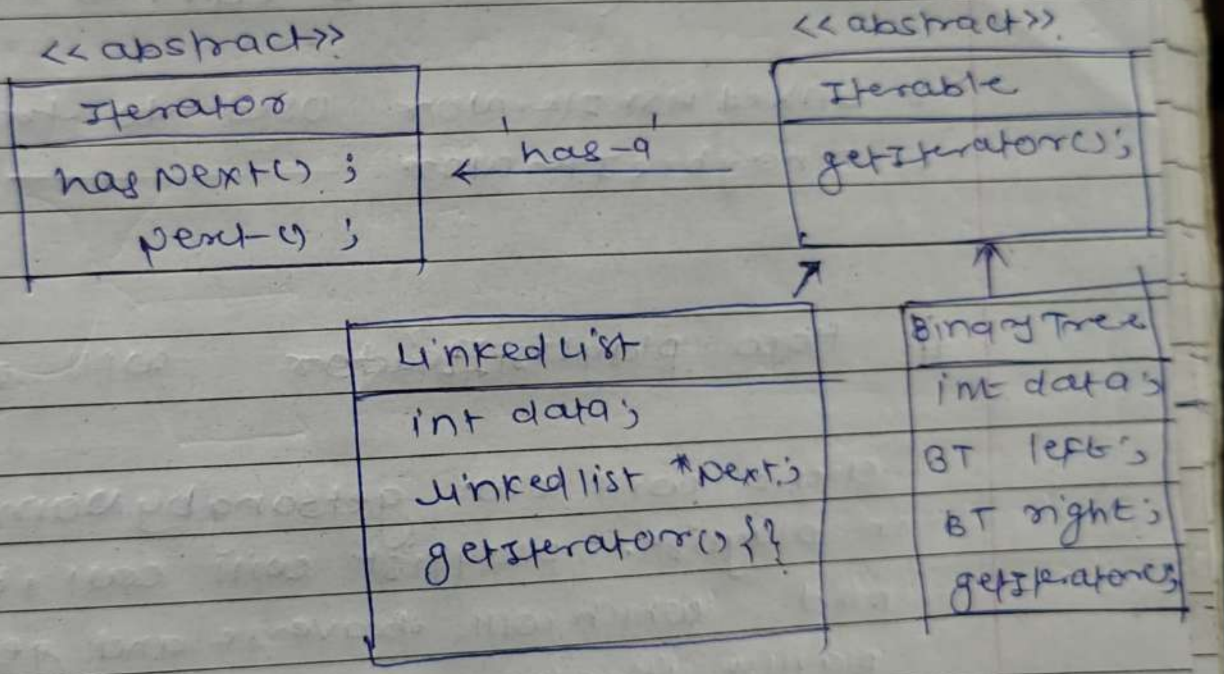
which means playlist should not know
how to traverse it just call one
method and get another song.

↳ we will use Iterator for that
though! more work but to
prevent breaking solid
principles.

using Iterator we will be able to follow SRP. even if we change its underlying. Data structure.

UML Diagram for playlist example

- First Iterator class :- knows how to ~~[illegible]~~ iterator perticular data structure.



∴ Now instead of calling getIterator() mtd banane ke jagah abstract class banadi named as Iterable.

* How LL traversing happens?

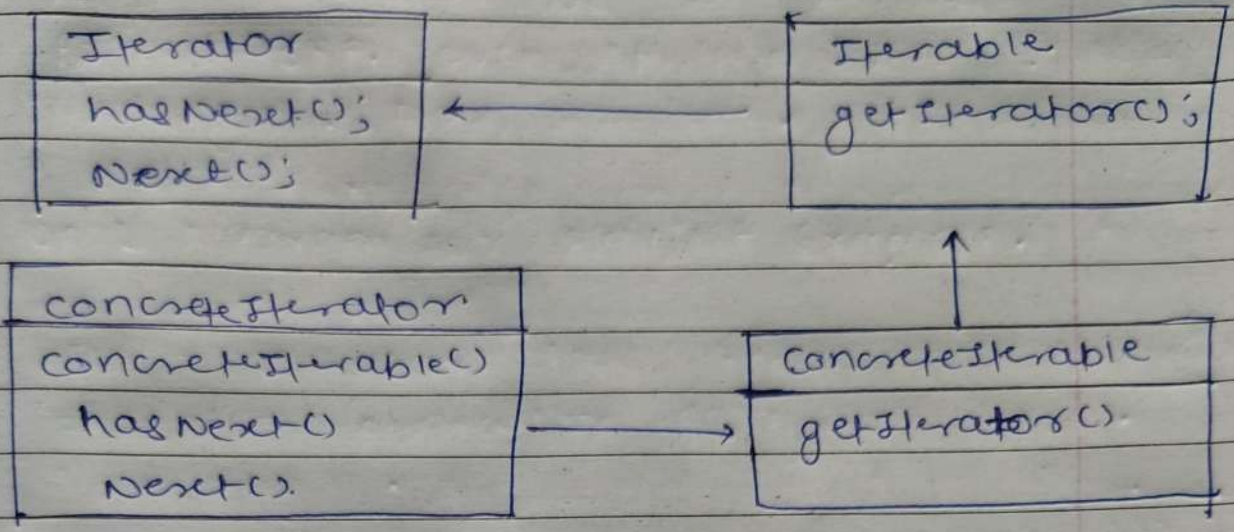
```
hasNext() {
    if (curr → next != null)
            return true;
      else
          return false;
  }
next() {   curr = curr → next;  }
```

* LinkedList Iterator used to traverse LL.
* BinaryTreeIterator ———————— BinaryTree

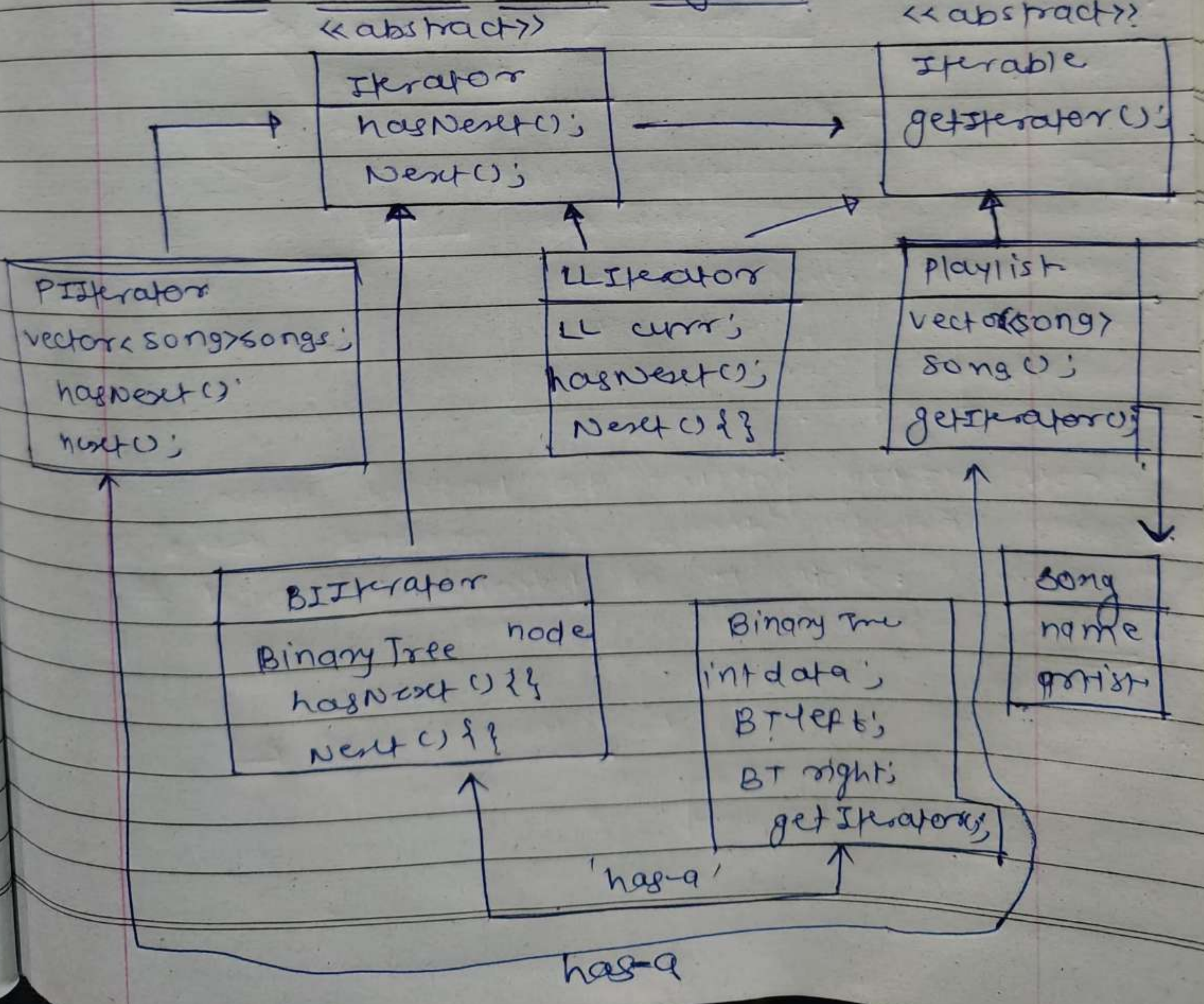
How playList~~Itedor~~ will work?

client will call getSongbyName() method in playlist. playlist will call it getIterator() mtd which will traverse and get's it's playlist iterator. then it will ~~store~~ the traverse the list of songs without knowing it's vector | LinkedList | Binarytree.

standard UML



UML playlist final Diagram:-



<<abstract>>
Iterator
hasNext();
Next();
<<abstract>>
Iterable
getIterator();
PIIterator
vector<song>songs;
hasNext()
next();
LLIterator
LL curr;
hasNext();
Next(){}
Playlist
vector<song>
song();
getIterator();
Song
name
artist
BIIterator
BinaryTree node
hasNext(){}
Next(){}
Binary Tree
int data;
BT left;
BT right;
getIterator();
'has-a'
has-a