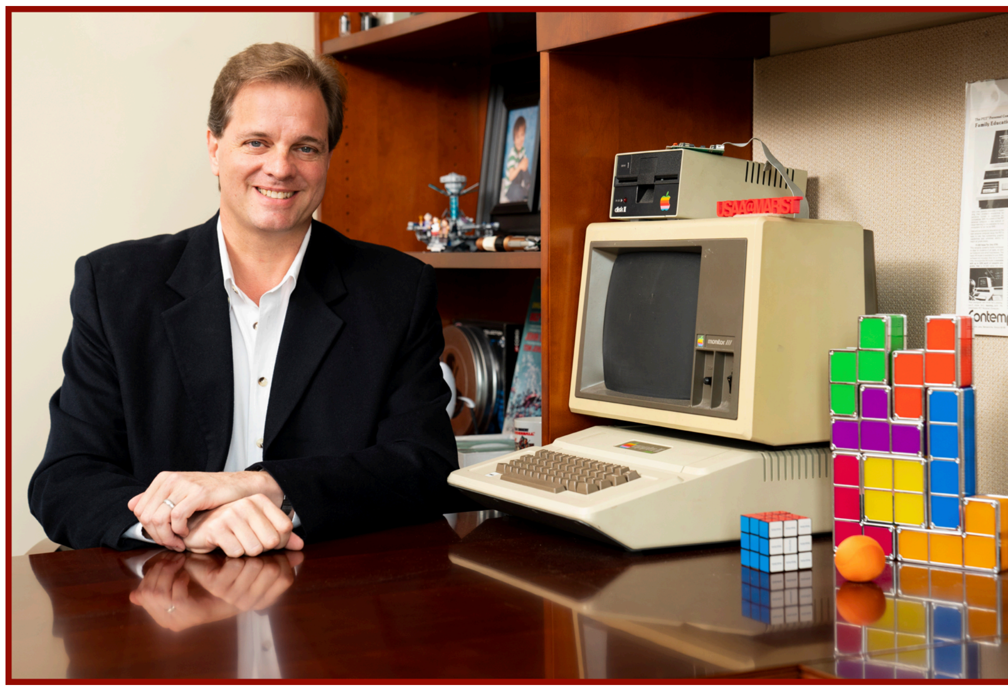
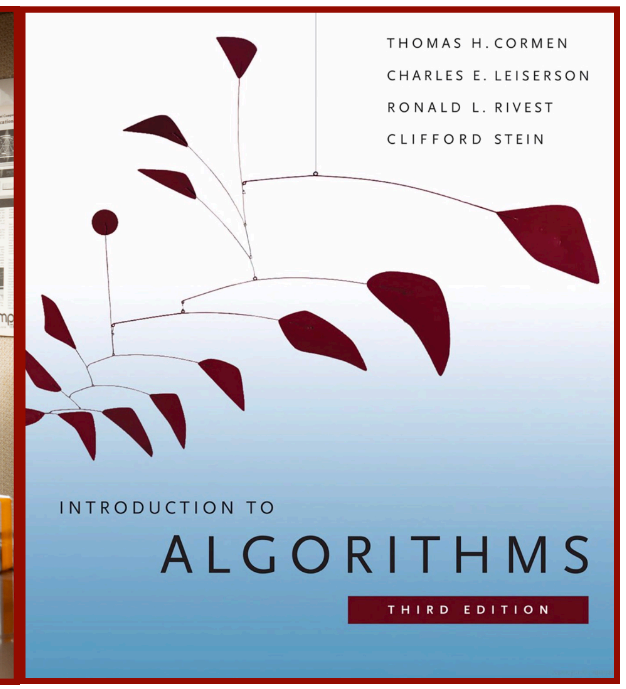
Elementary Data Structures





Alan G. Labouseur, Ph.D.
Alan.Labouseur@Marist.edu

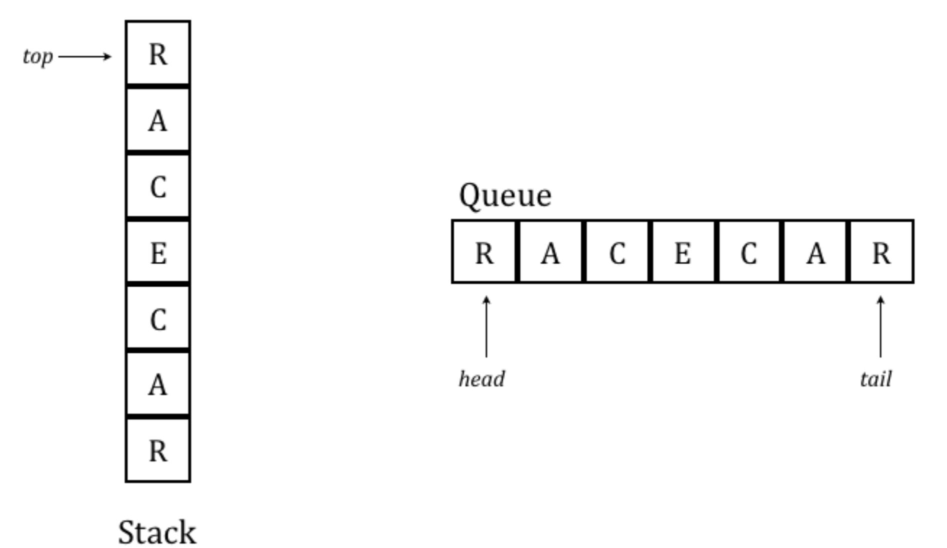
Algorithms :: Elementary Data Structures

Algorithms are general recipes for solving problems
not specific of any language or platform.

To study algorithms, we need something for them to act upon.
That brings up **Data Structures**.

As with algorithms themselves, data structures present the same challenges:

Challenges:
- Correctness
- Efficiency
- Applicability

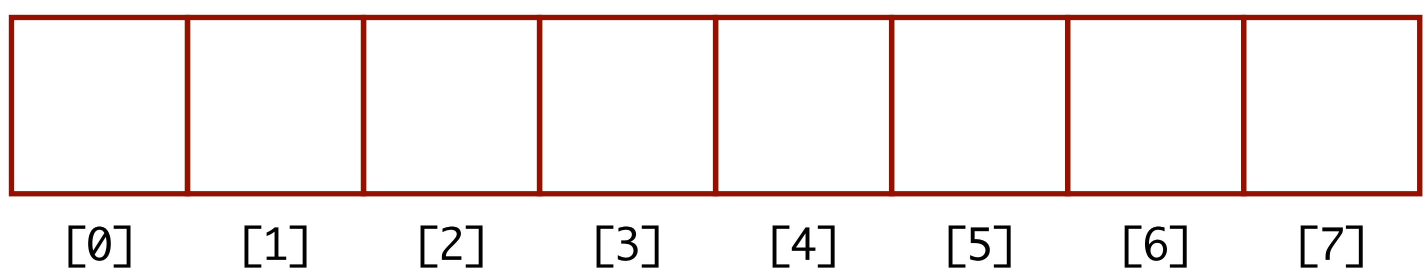


Algorithms :: Elementary Data Structures

Elementary Data Structures

- Lists
- Stacks
- Queues
- ?

But before we get to those, consider the **array** data type.



This is an array of **length 8** indexed from **0** to **7**.
We could fill this array with data, for example ...

Arrays

We could fill this array with eight James Bond movie names.

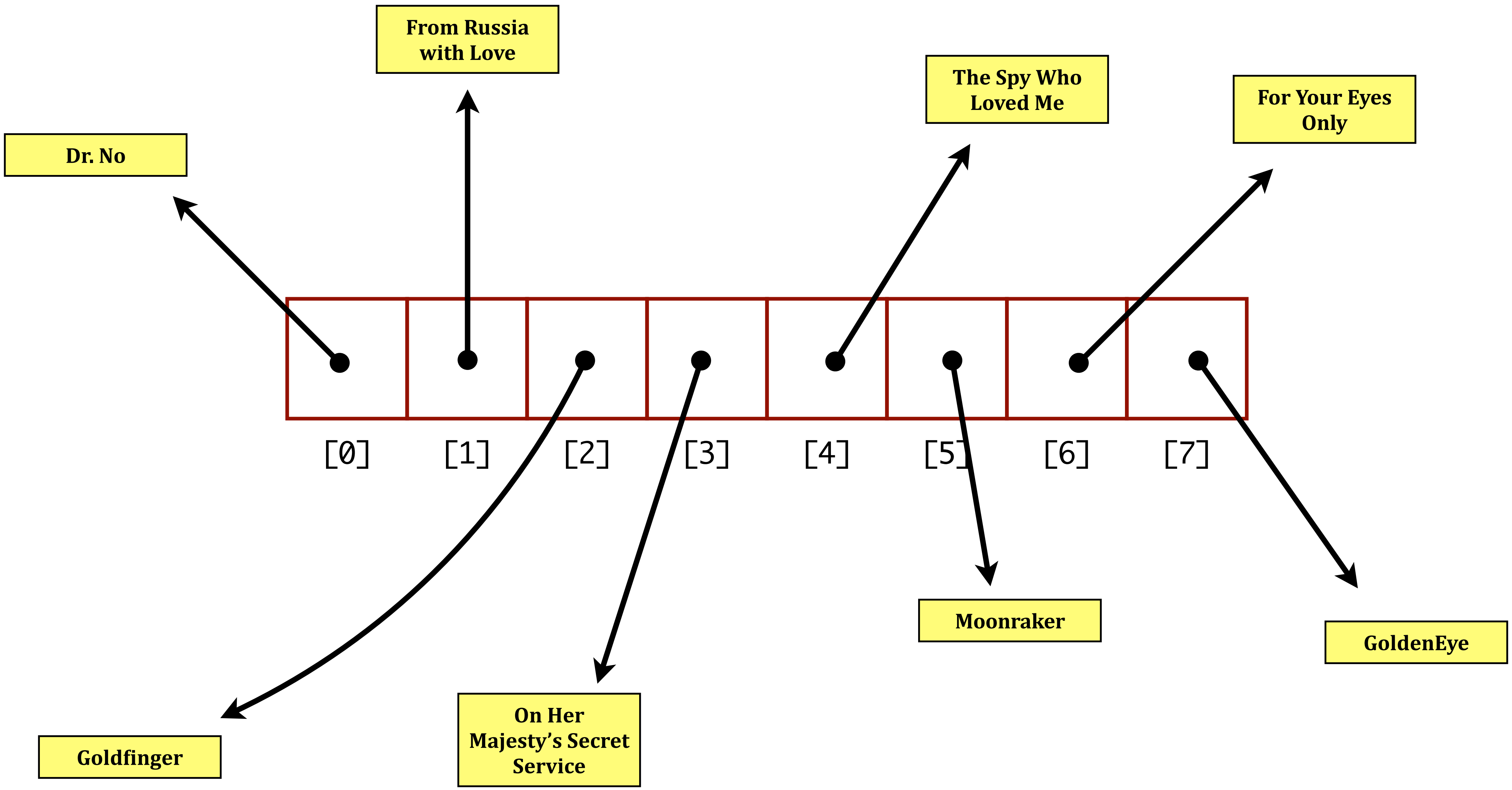
```
string bondFilms[8] = { };

bondFilms[0] = "Dr. No";
bondFilms[1] = "From Russia with Love";
bondFilms[2] = "Goldfinger";
bondFilms[3] = "On Her Majesty's Secret Service";
bondFilms[4] = "The Spy Who Loved Me";
bondFilms[5] = "Moonraker";
bondFilms[6] = "For Your Eyes Only";
bondFilms[7] = "GoldenEye";
```

Dr. No	From Russia with Love	Goldfinger	On Her Majesty's Secret Service	The Spy Who Loved Me	Moonraker	For Your Eyes Only	GoldenEye
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

In this example, we have an array of string (text) values. We can have arrays of other data types, including pointers.

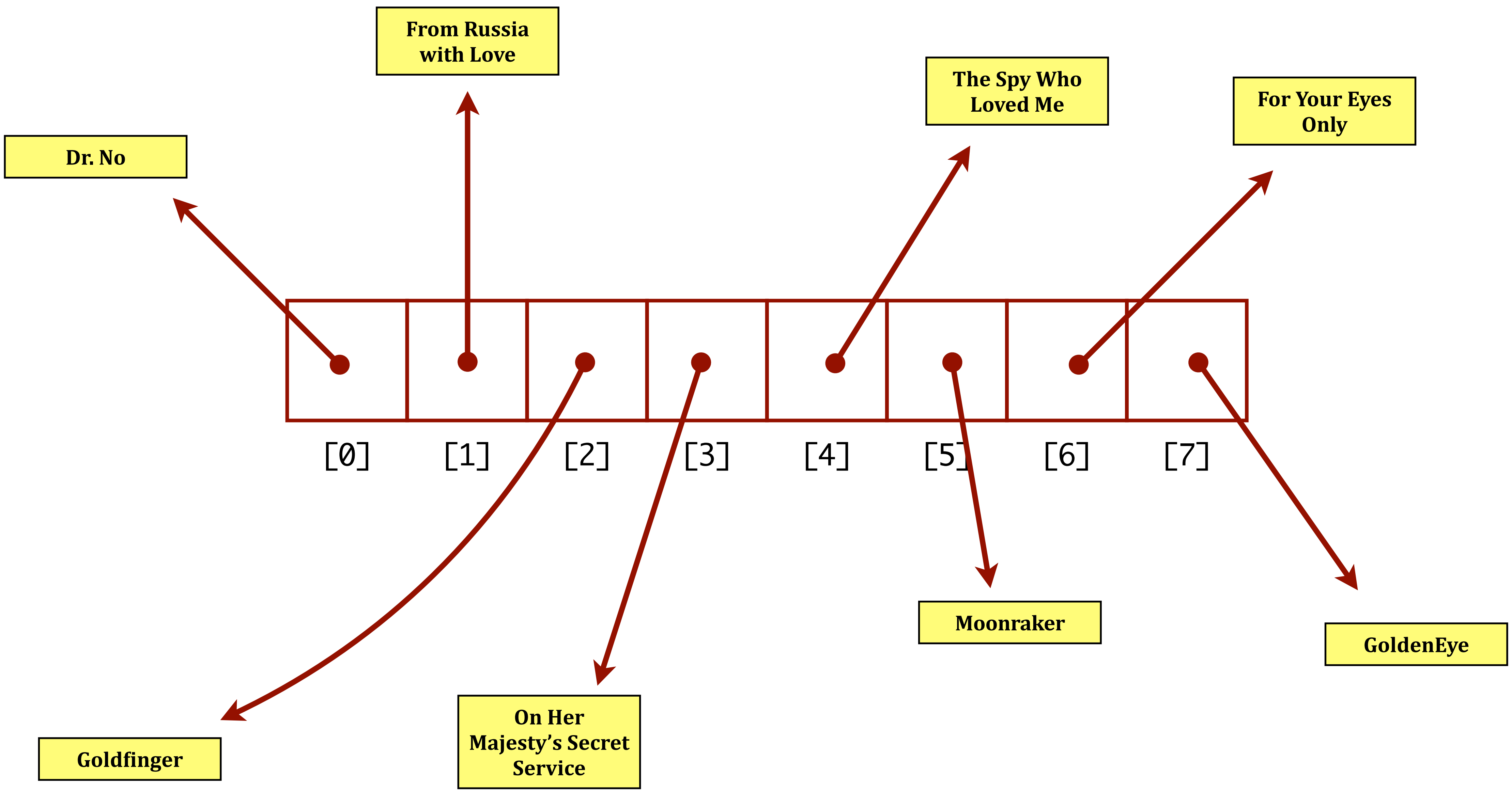
Arrays

We could fill this array with pointers to objects representing eight James Bond movie names.



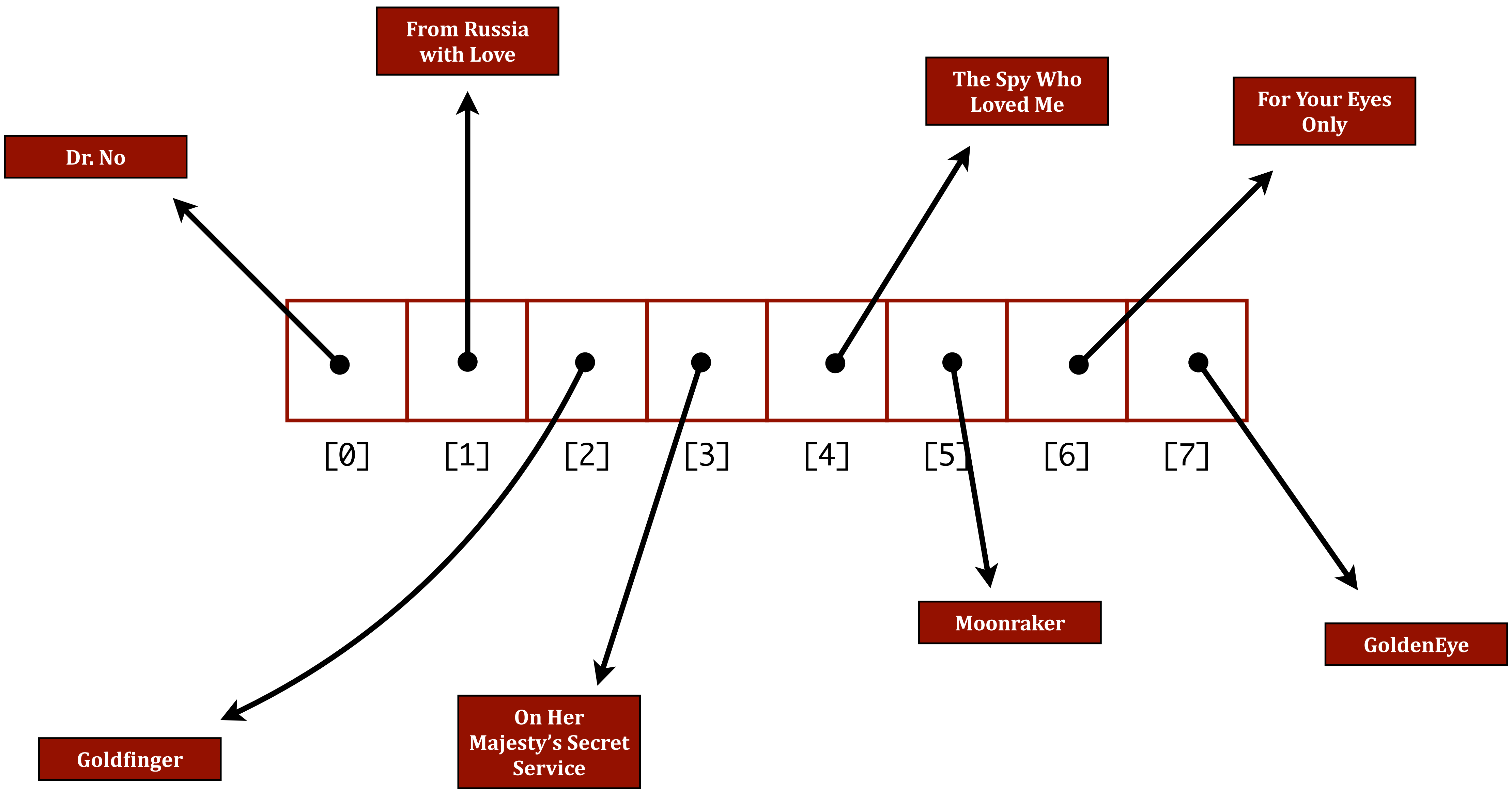
Arrays

We could fill this array with **pointers** to objects representing eight James Bond movie names.



Arrays

We could fill this array with pointers to **objects** representing eight James Bond movie names.

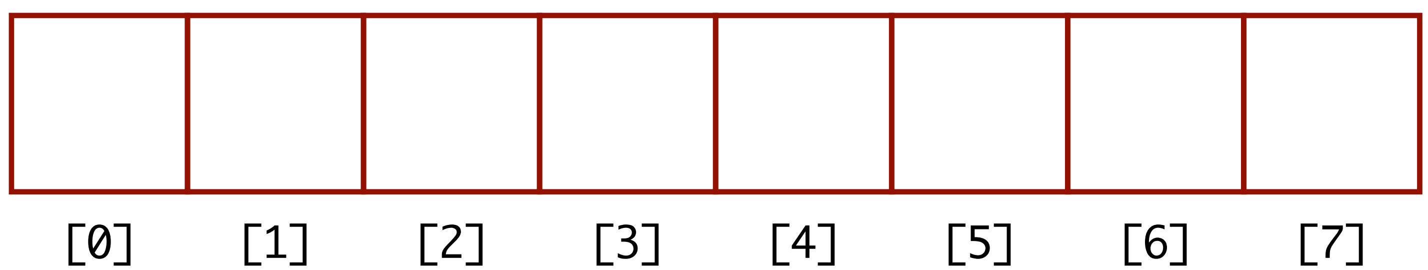


Lists

Elementary Data Structures

- Lists
- Stacks
- Queues
- ?

Q: Does an array work for a list?



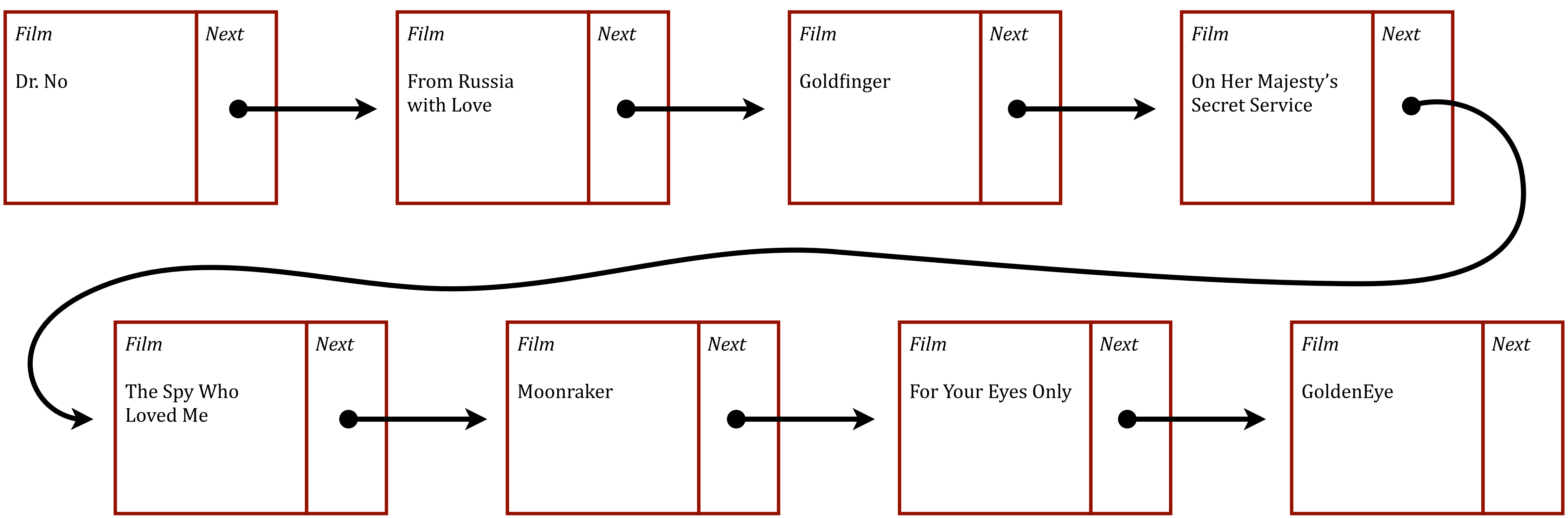
A: It certainly can.
Especially if we know in advance how many items we want to store in the list.

But what if we need to be more flexible?

Linked Lists

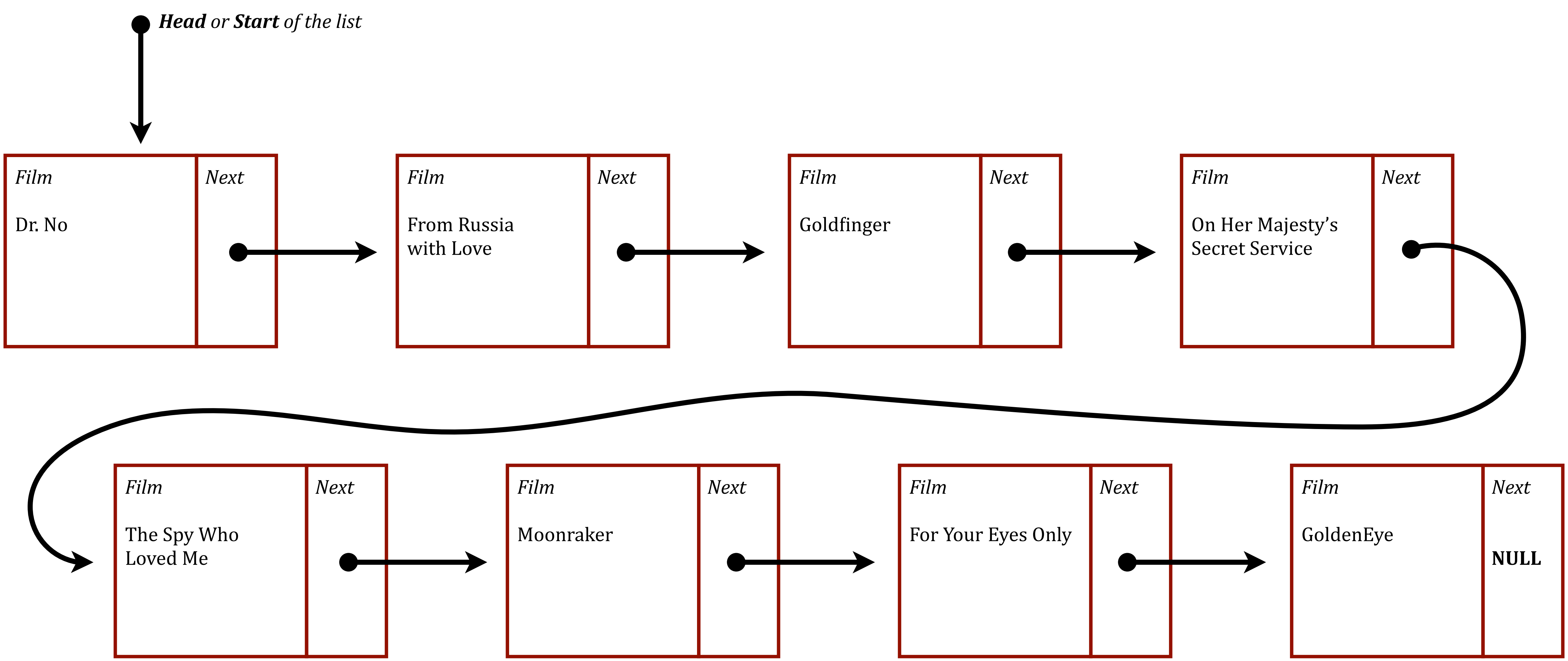
For flexibility, we need a linked list.

It's a **list**… of **linked** objects.



Linked Lists

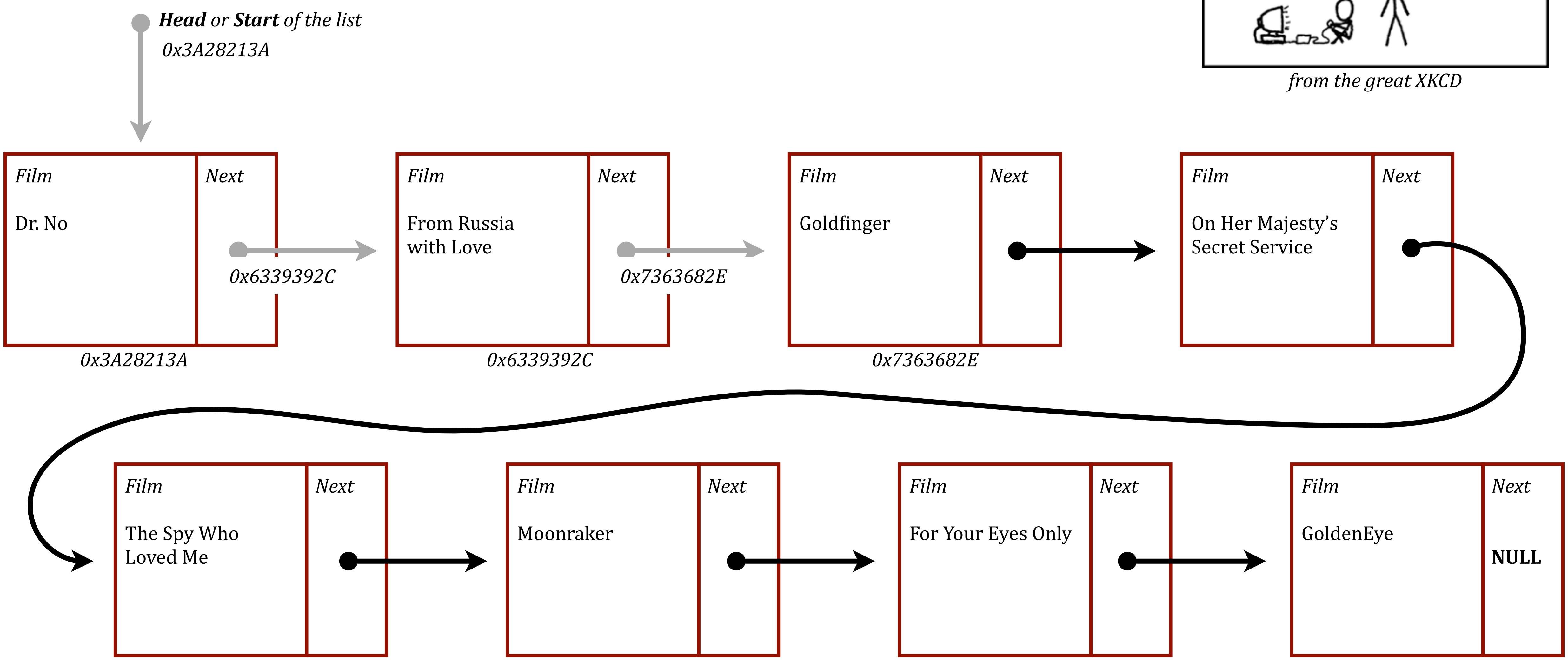
We need to know the **head** or **start** of the list.
We also need to denote that the last item has no *Next* pointer.



Linked Lists

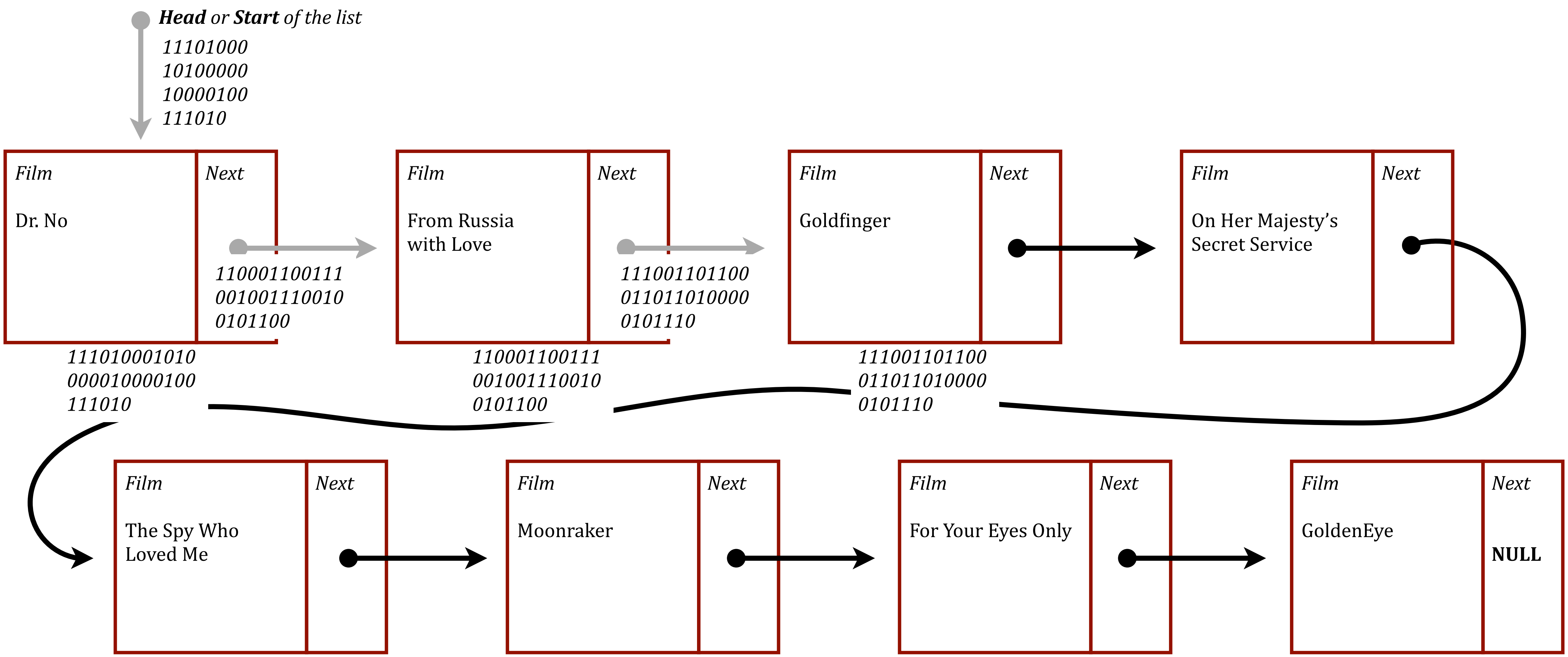
Pointers are really addresses in memory.

from the great XKCD



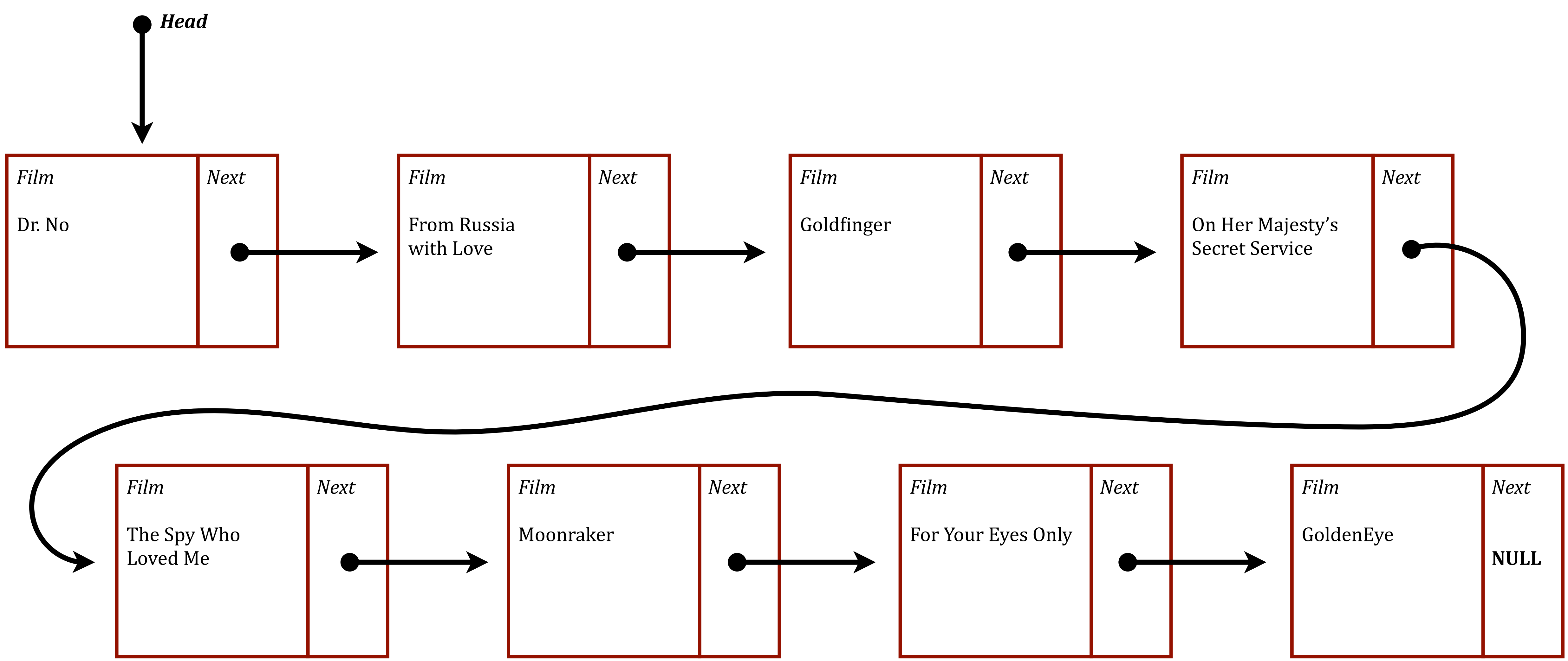
Linked Lists

Pointers are really addresses in memory.
Those hexadecimal addresses are really just shorthand for binary.



Linked Lists

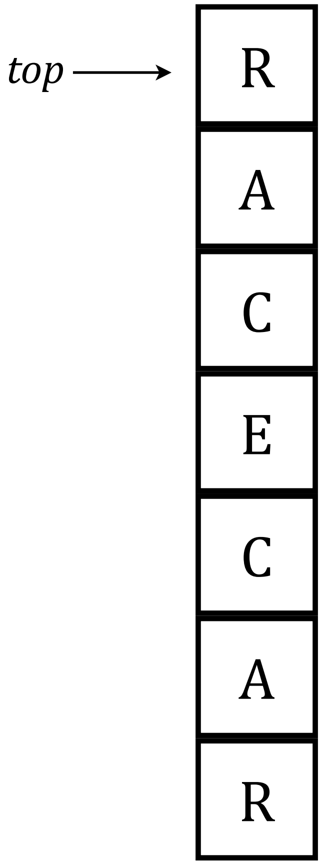
We can use linked lists to build elementary data structures.



Stacks

Elementary Data Structures

- Lists
- Stacks
- Queues
- ?



Stack

Stack

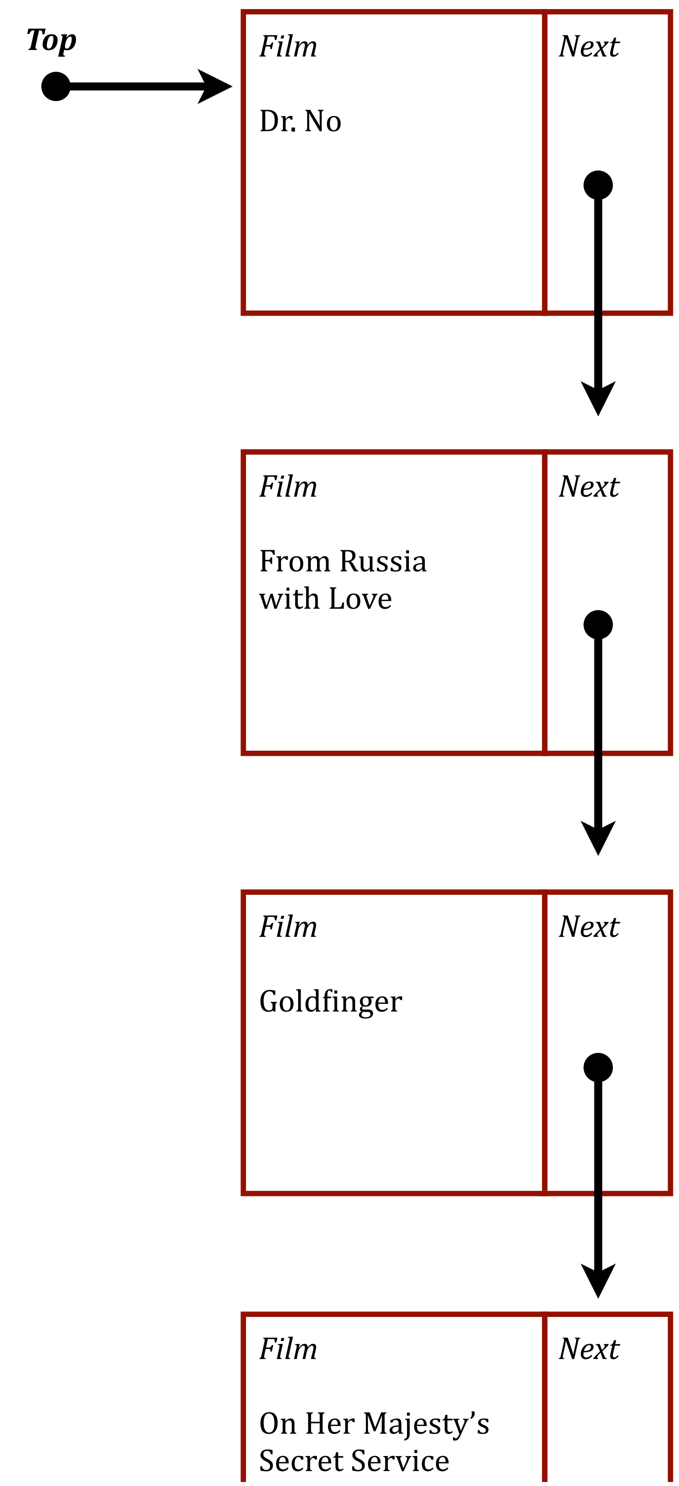
A stack is an abstract data type that supports the following operations:

- Push - add an element to the top of the stack
- Pop - remove an element from the top of the stack
- isEmpty - check to see whether or not the stack is empty

Stack

Operations:

- Push
- Pop
- isEmpty

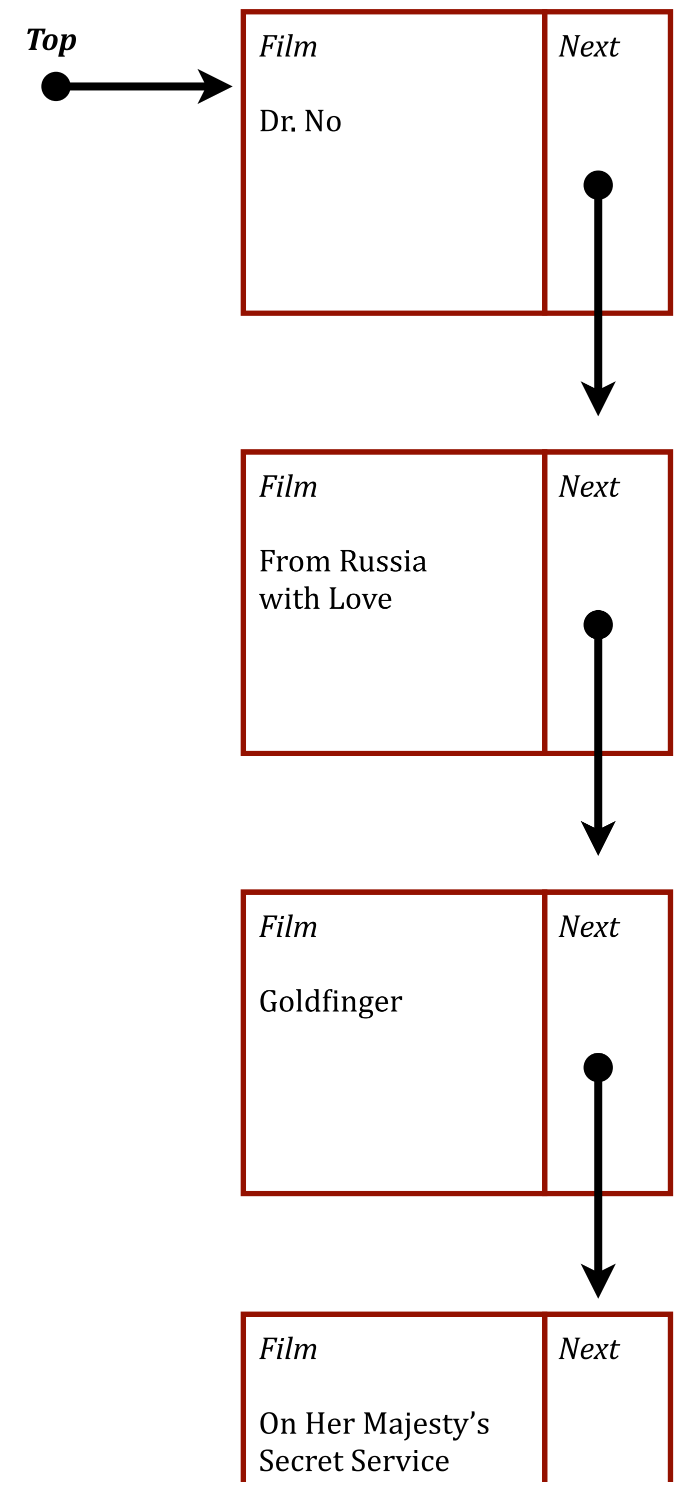


Stack

Operations:

- Push
- **Pop**
- isEmpty

```
m = stack.pop()
```

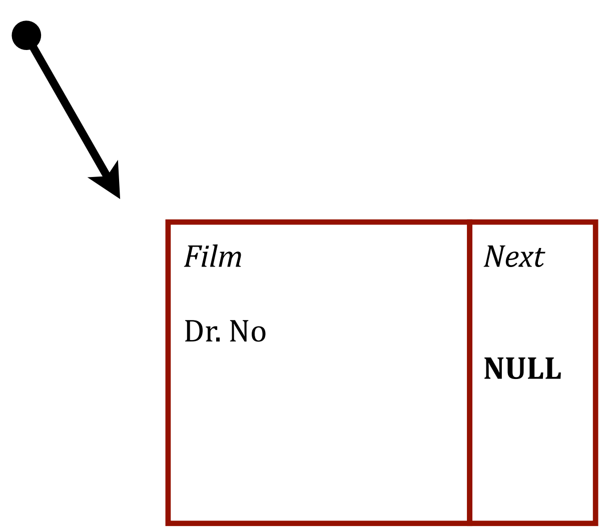


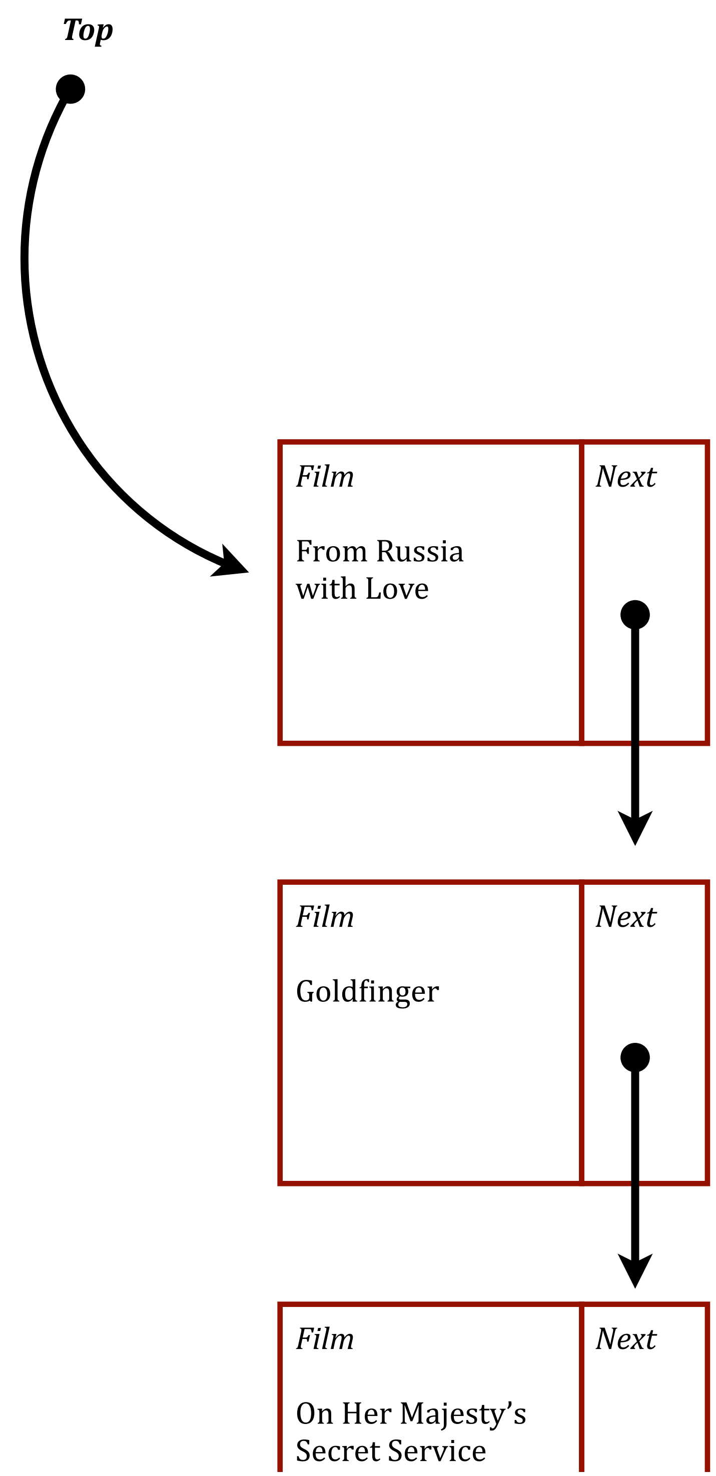
Stack

Operations:

- Push
- **Pop**
- isEmpty

```
m = stack.pop()
```



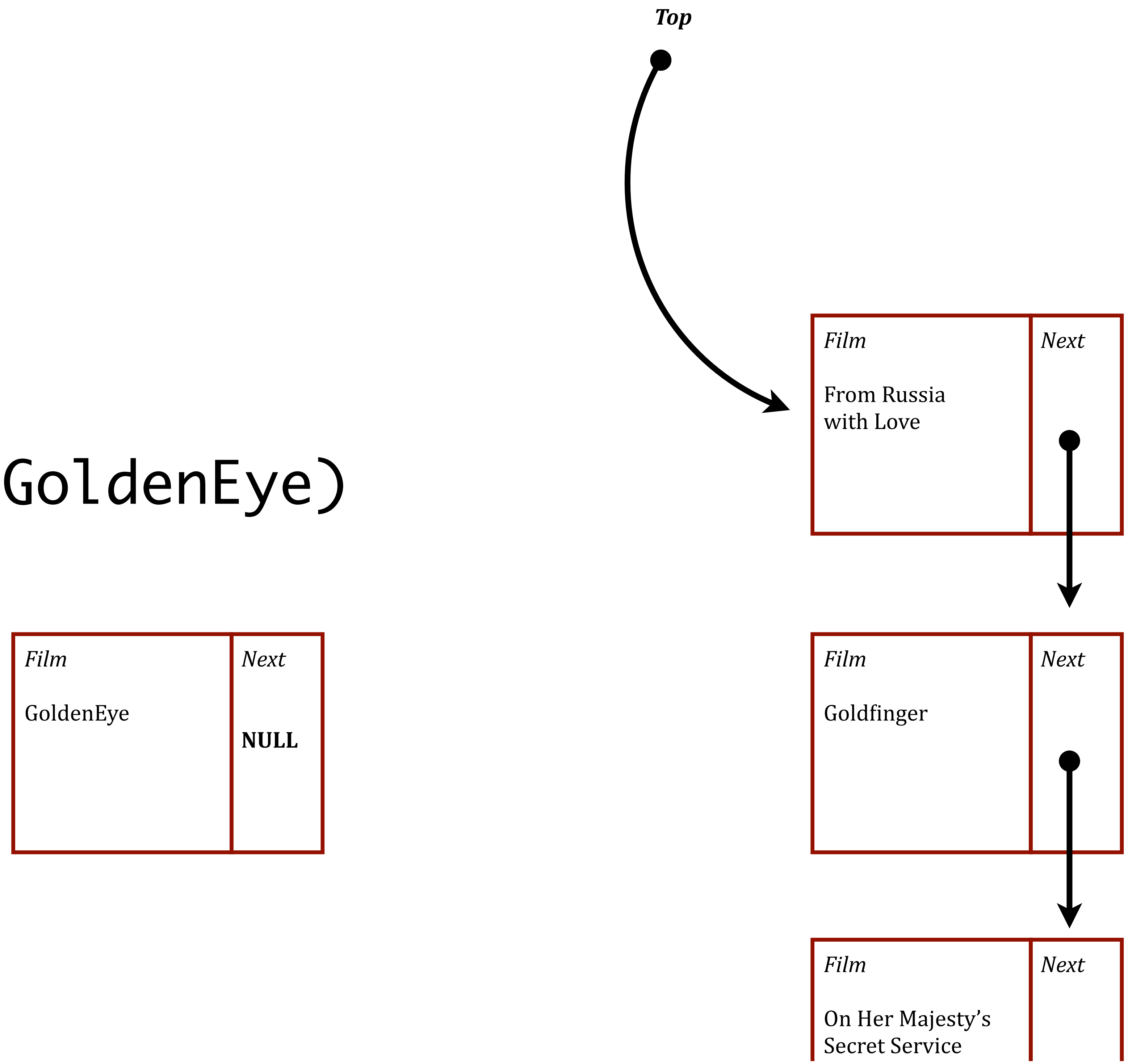


Stack

Operations:

- **Push**
- Pop
- isEmpty

```
stack.push(GoldenEye)
```

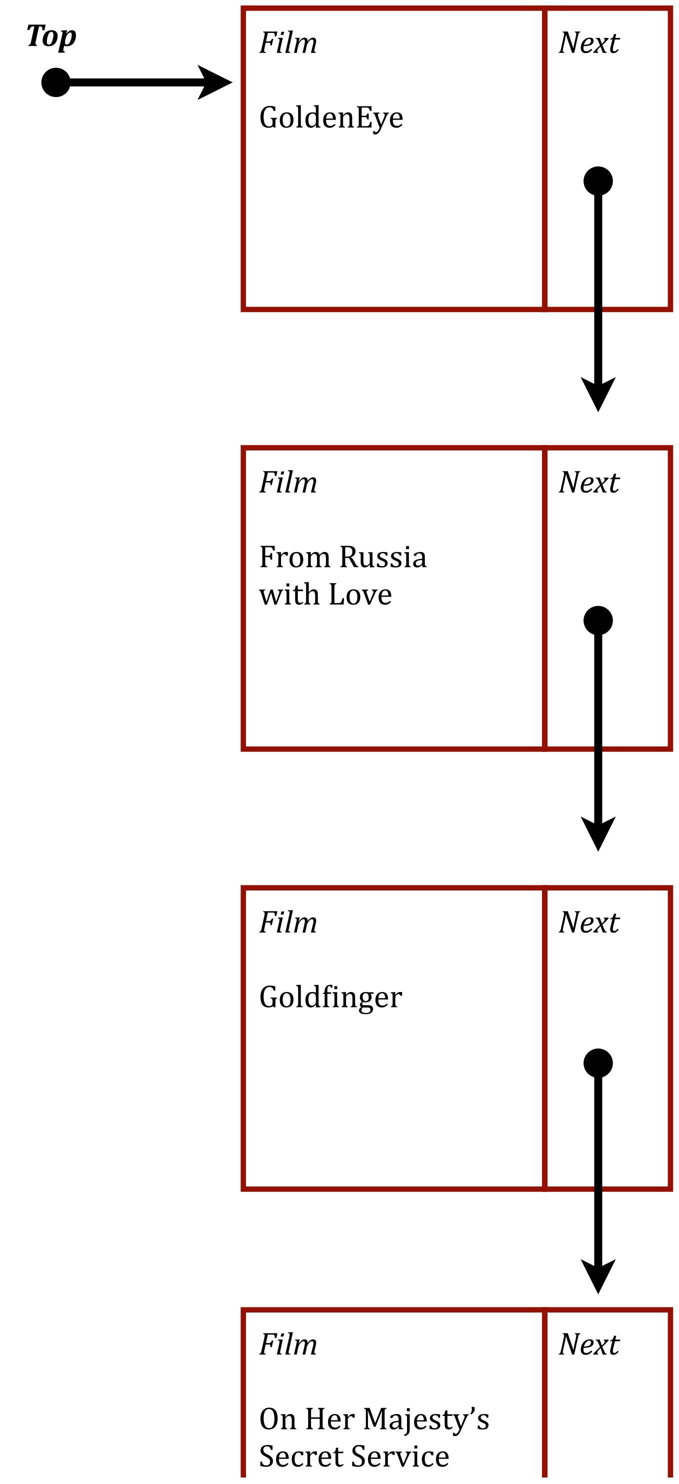


Stack

Operations:

- **Push**
- Pop
- isEmpty

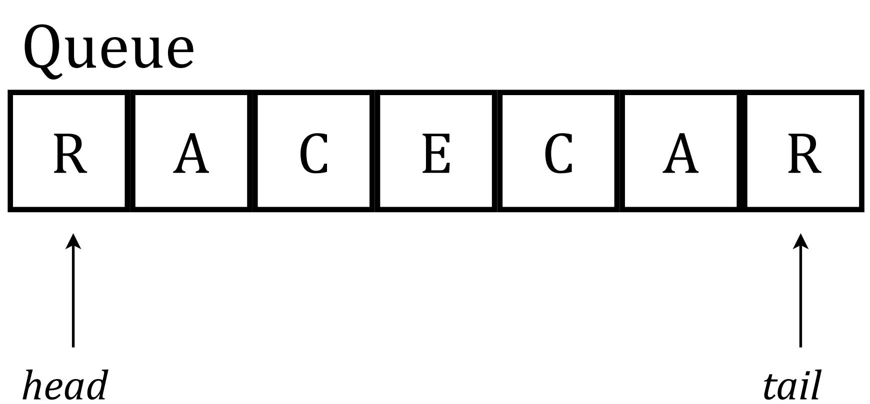
```
stack.push(GoldenEye)
```



Queues

Elementary Data Structures

- Lists
- Stacks
- Queues
- ?



Queue

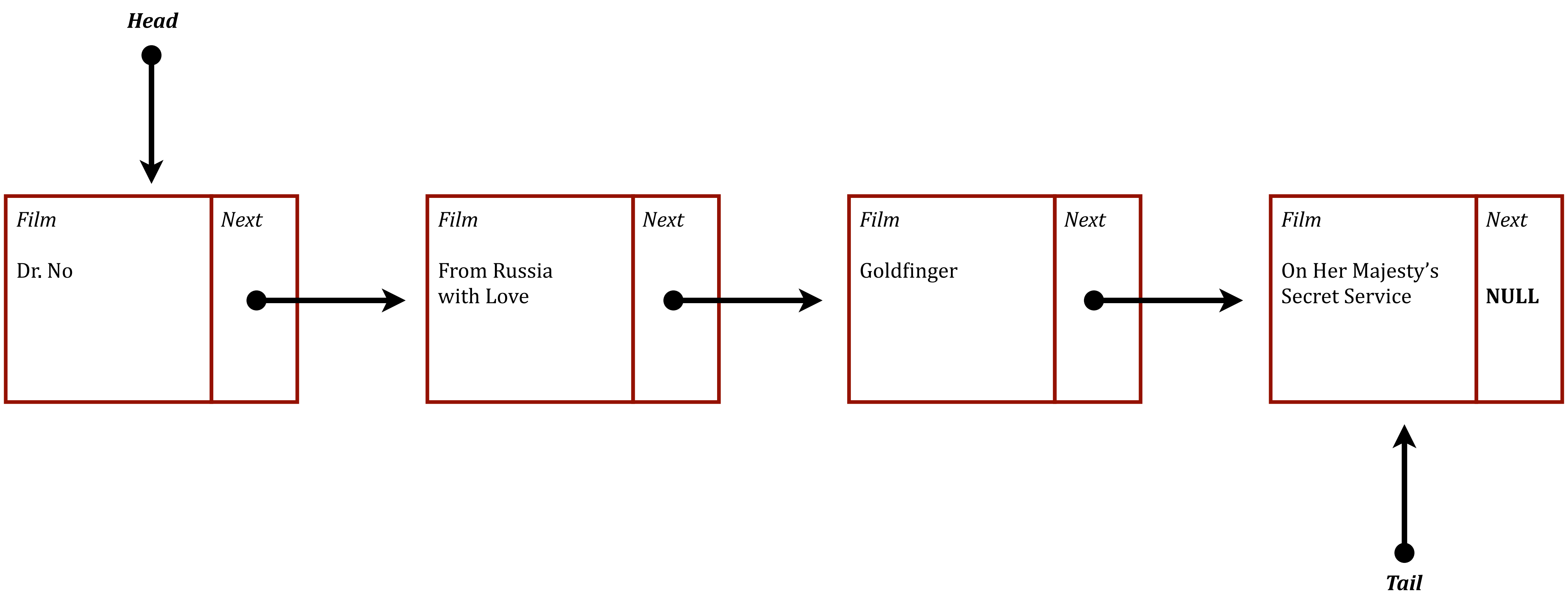
A queue is an abstract data type that supports the following operations:

- Enqueue - add an element to the back of the queue
- Dequeue - remove an element from the front of the queue
- isEmpty - check to see whether or not the queue is empty

Queue

Operations:

- Enqueue
- Dequeue
- isEmpty

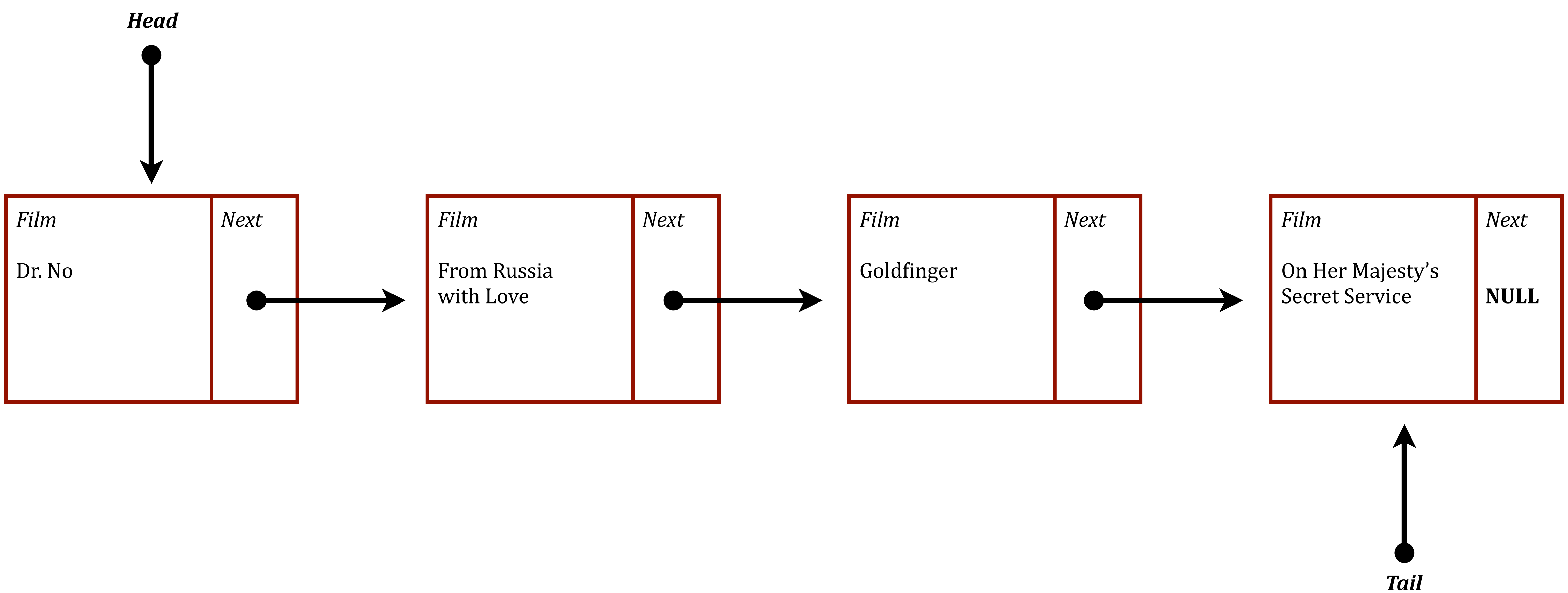


Queue

Operations:

- Enqueue
- **Dequeue**
- isEmpty

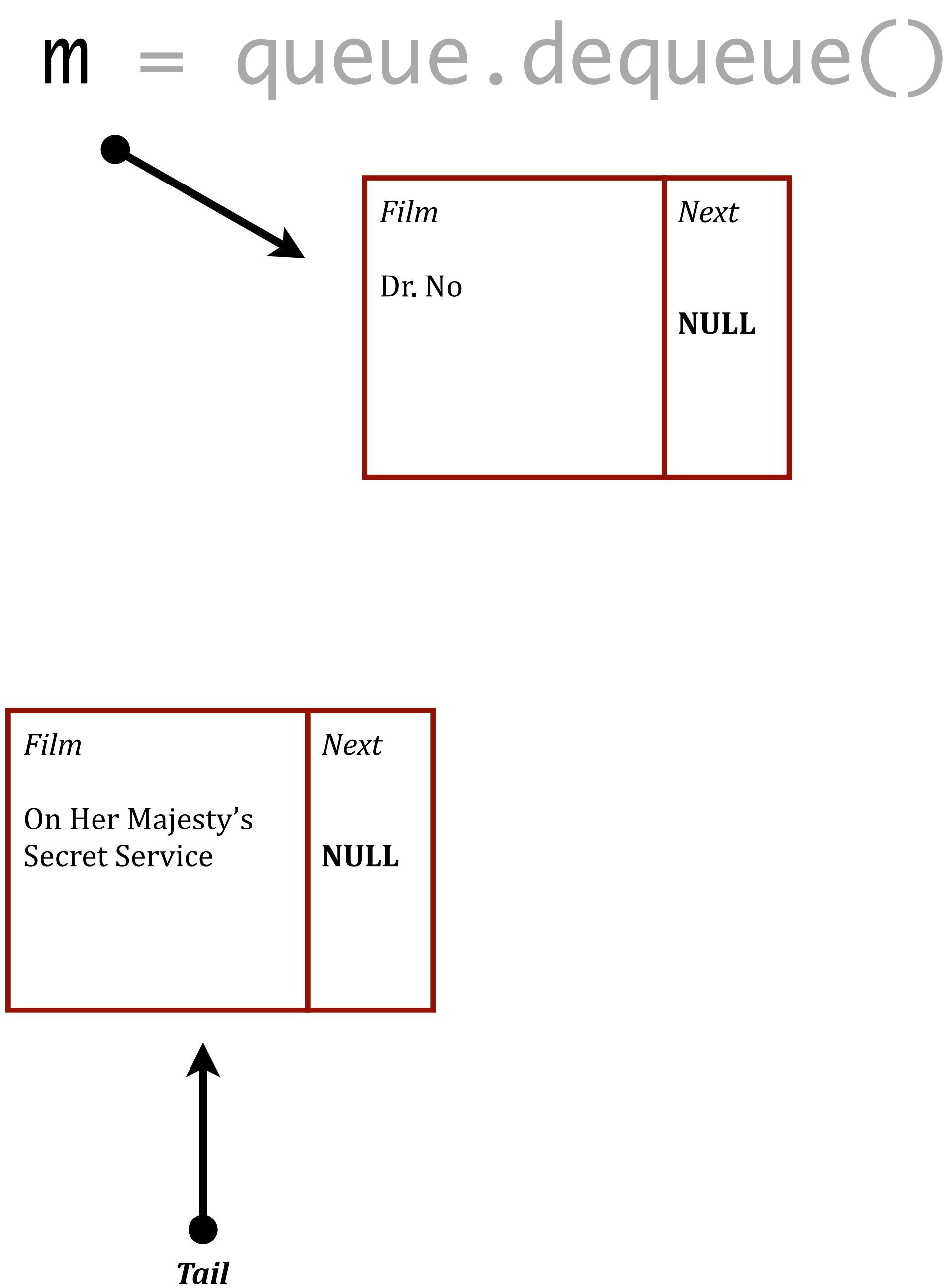
```
m = queue.dequeue()
```



Queue

Operations:

- Enqueue
- **Dequeue**
- isEmpty

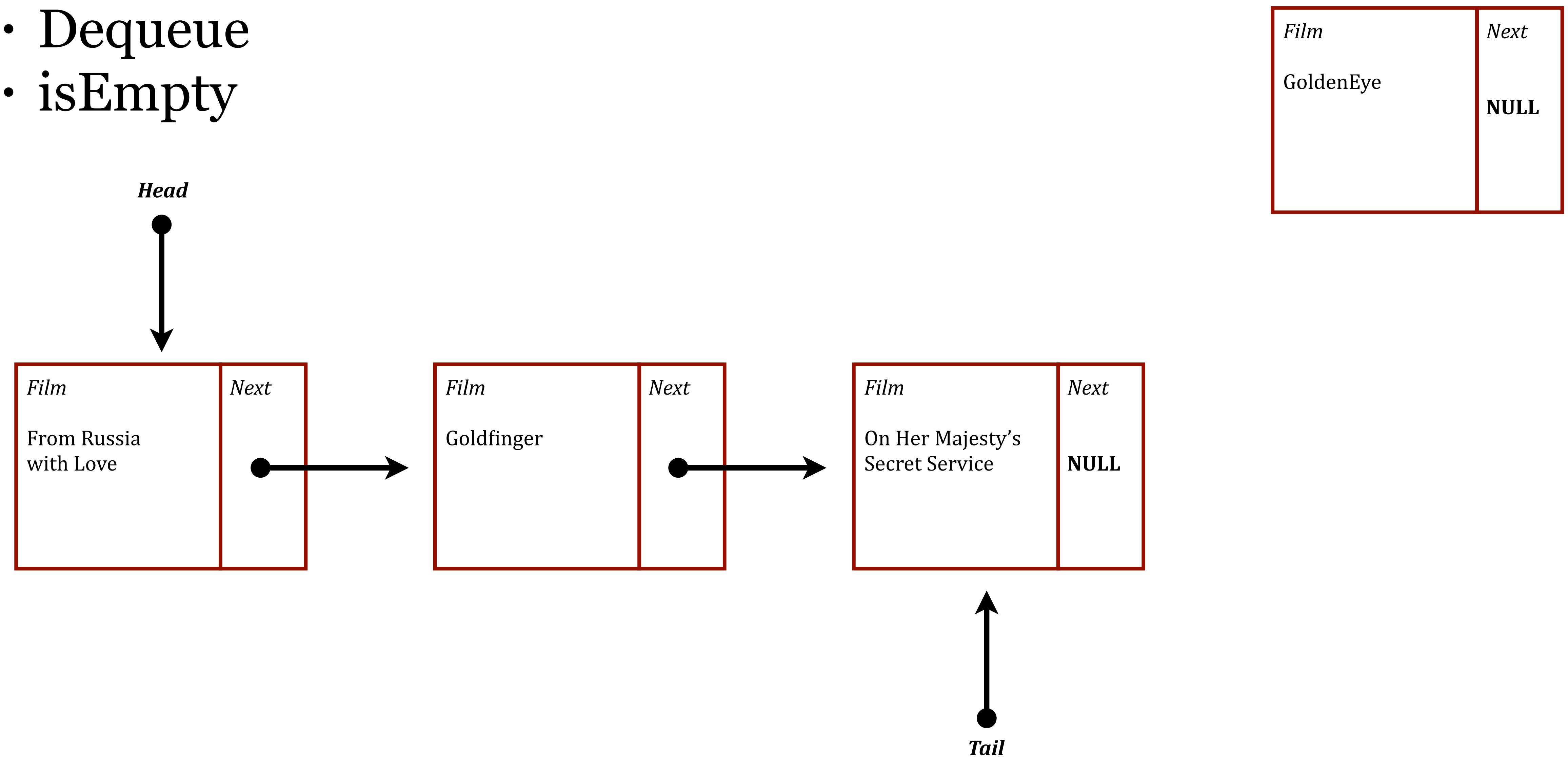


Queue

Operations:

- **Enqueue**
- Dequeue
- isEmpty

`queue.enqueue(GoldenEye)`

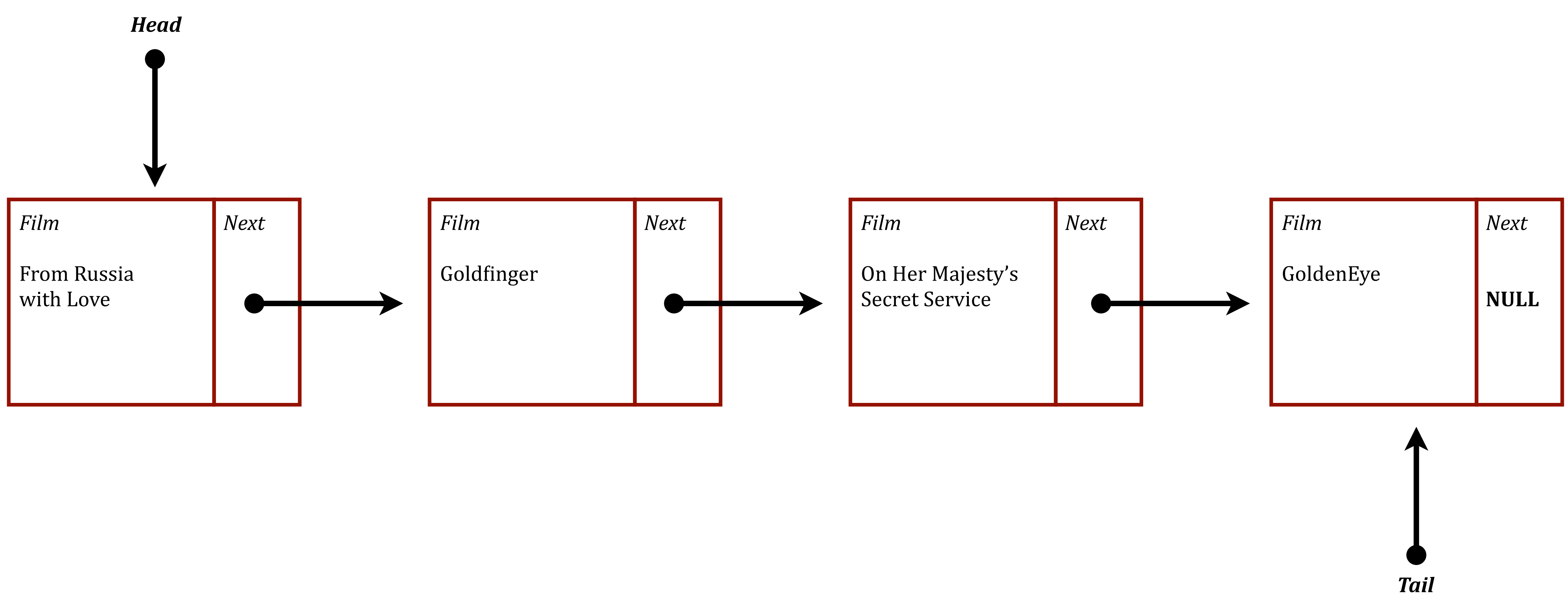


Queue

Operations:

- **Enqueue**
- Dequeue
- isEmpty

`queue.enqueue(GoldenEye)`



Algorithms :: Performance Characteristics

What's the "Big Oh" of these operations?

- Lists
  - add
  - remove
- Stacks
  - push
  - pop
  - isEmpty
- Queues
  - enqueue
  - dequeue
  - isEmpty

Algorithms :: Wait… more?

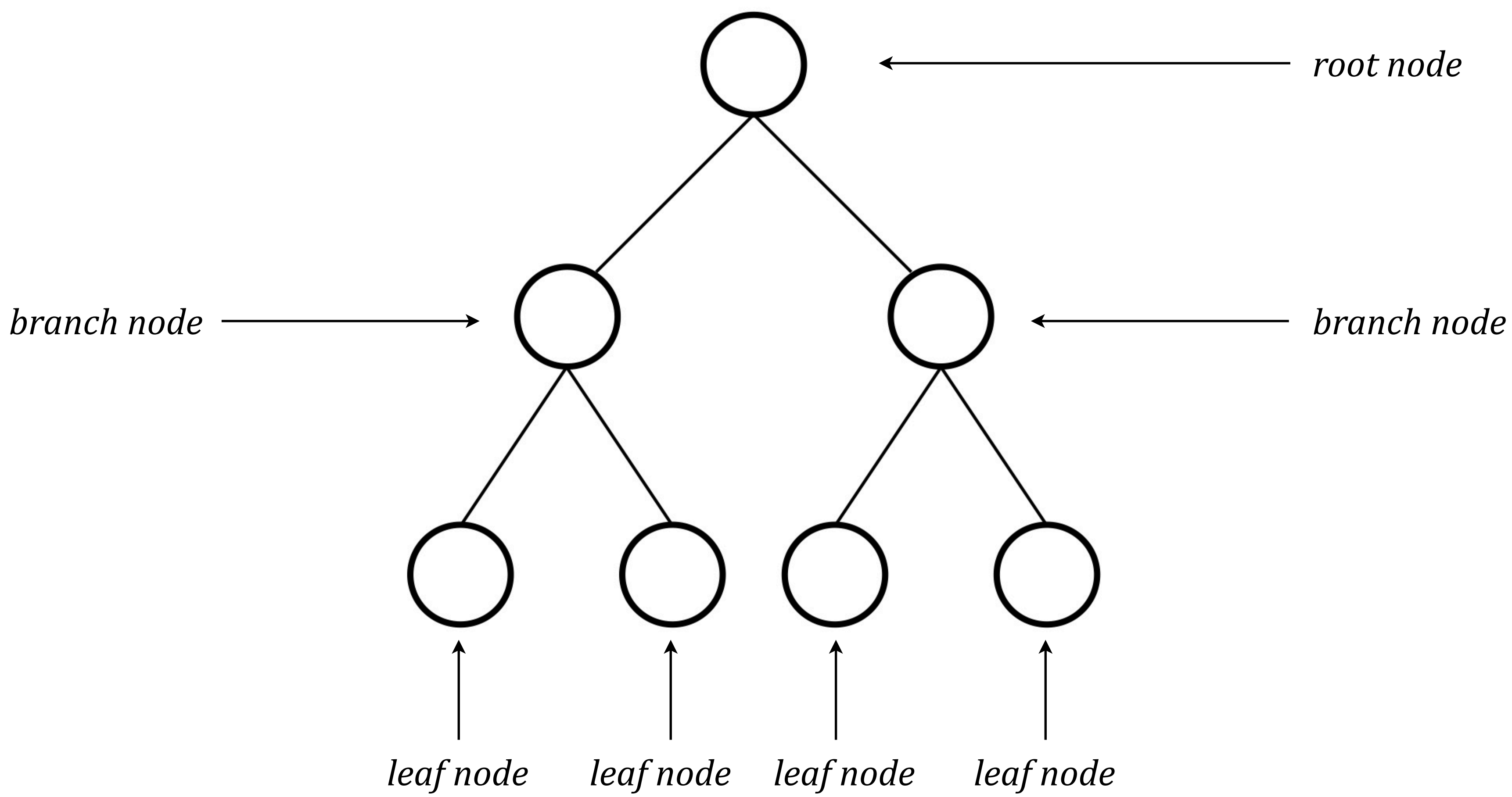
Elementary Data Structures

- Lists
- Stacks
- Queues
- ?

Algorithms :: More?

Elementary Data Structures

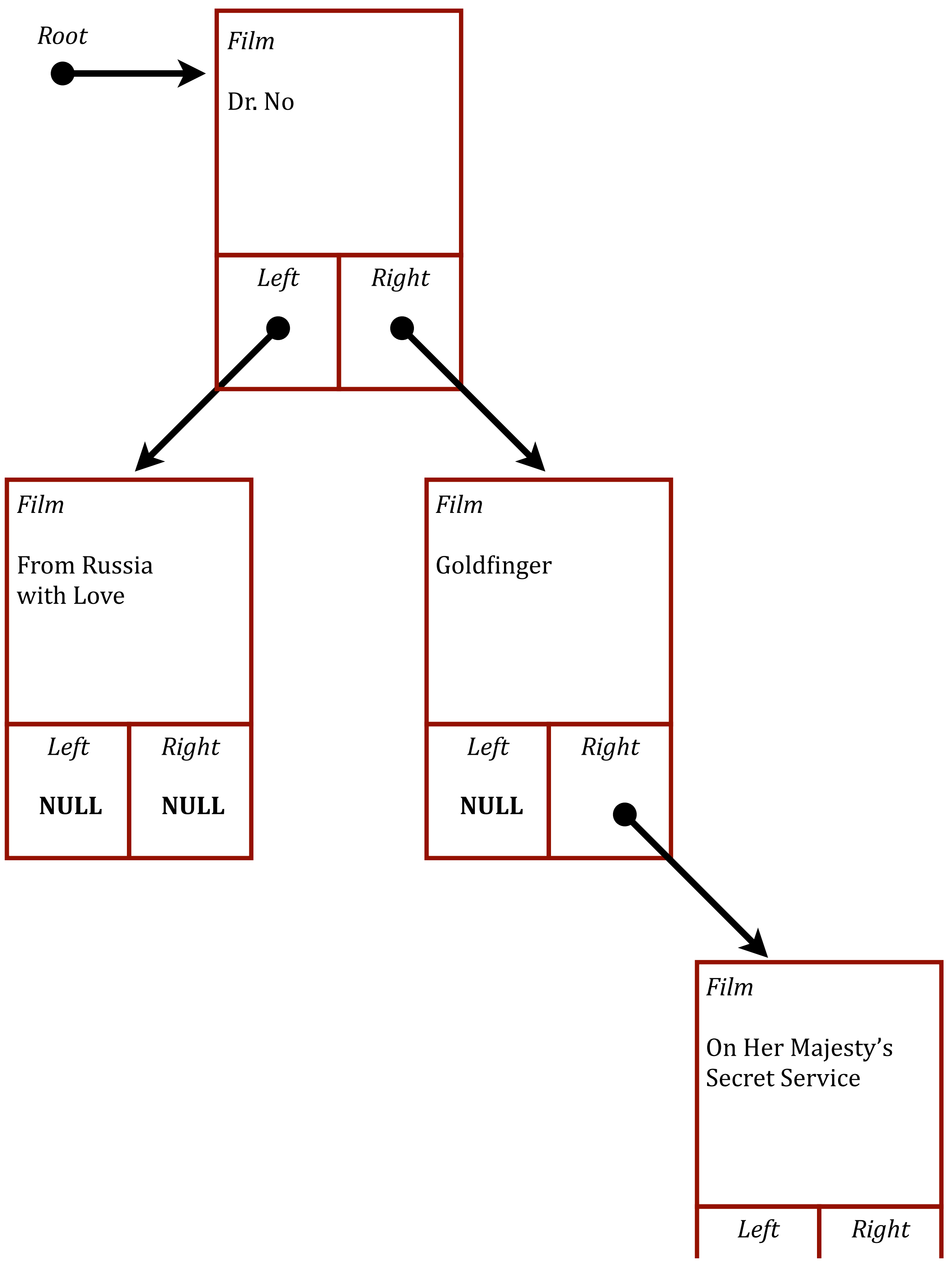
- Lists
- Stacks
- Queues
- Trees



Tree

Operations:

- Insert
- Delete
- Traverse
- isEmpty



Tree

Operations:

- Insert
- Delete
- Traverse
- isEmpty