Data structures:

Array, ArrayList, LinkedList

Dr. Sophea PRUM
sopheaprum@gmail.com

Review: array

- Array

An *array* is a container that holds a fixed number of values of a single type. An *array* is an indexed data structure, which means *array* elements can be accessed by their index numbers using the *subscript operator* [ ].

Review: array

- Create an array object

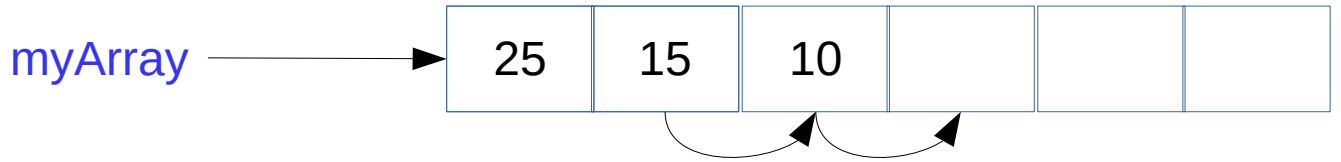
```
int myArray[] = new int[6];
```

- Assign value

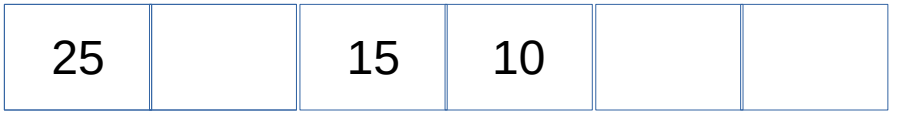
```
myArray[0] = 25;
myArray[1] = 15;
myArray[2] = 10;
```

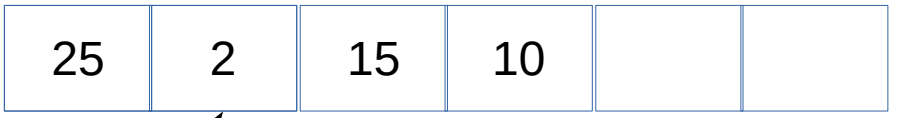
Review: array

- Insert an element



1. Shift elements to the right

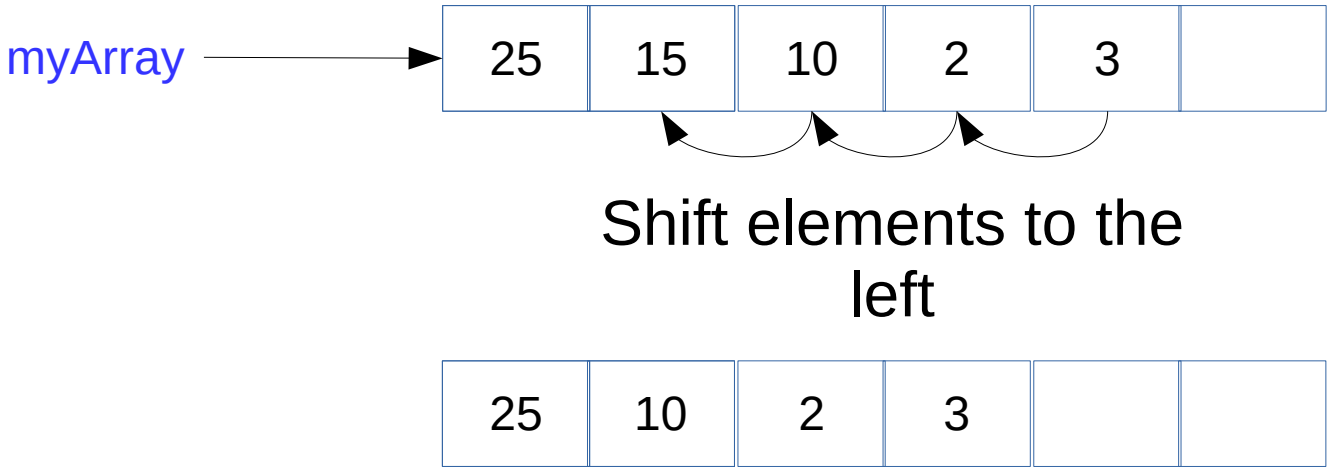




2. Assign value
myArray[1] = 2;

Review: array

- Delete an element
  - Example: delete an element at index 1



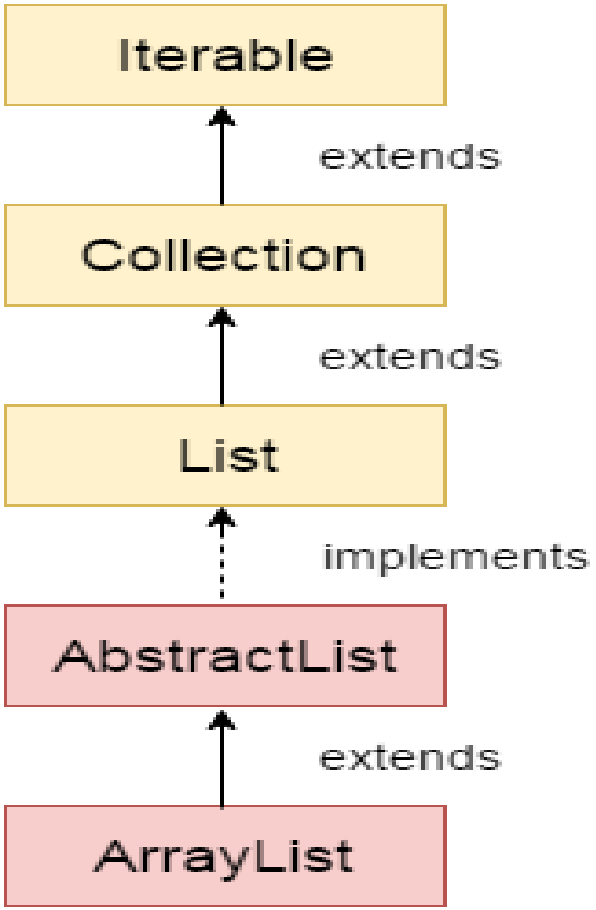
Review: array

- You can’t do the following with an array object:
  - Increase or decrease its length, which is fixed
  - Add an element at a specified position without shifting the other elements to make room.
  - Remove an element at a specified position without shifting the other elements to fill in the resulting gap.

Review: ArrayList

ArrayList provides methods to do the operations that cannot be done with array object: add, insert, delete element(s) and more.

However, ArrayList use array object as data structure to store the data. So the operation such as insert and delete element(s) is a time consuming process.



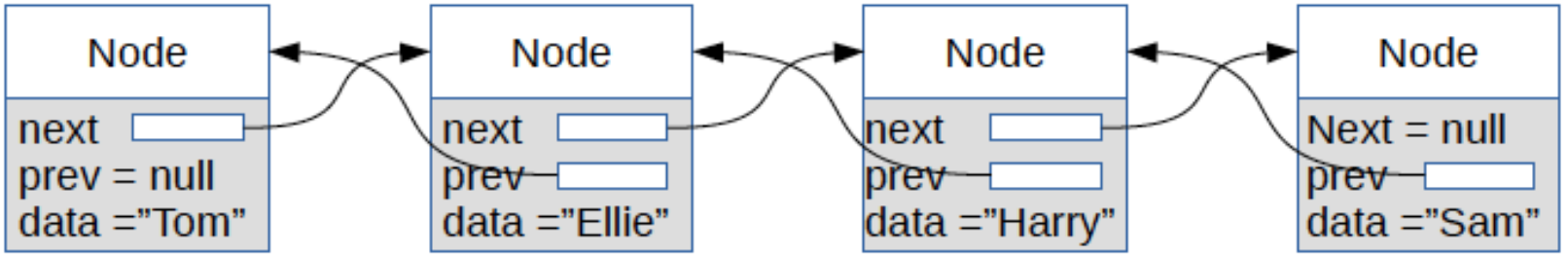
LinkedList

- A LinkedList can be created by connecting different objects using pointers (references)
  - Object is called node

```
/** A Node is the building block for a double-linked list. */
private static class Node<E> {
    /** The data value. */
    private E data;
    /** The link to the next node. */
    private Node<E> next = null;
    /** The link to the previous node. */
    private Node<E> prev = null;

    /** Construct a node with the given data value.
        @param dataItem The data value
     */
    private Node(E dataItem) {
        data = dataItem;
    }
}
```

LinkedList

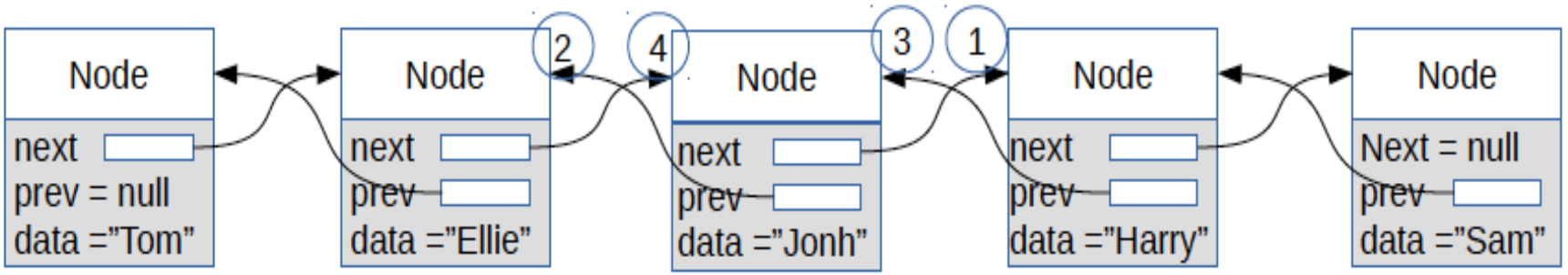
```
Node<String> tom = new Node<>("Tom");
Node<String> ellie = new Node<>("Ellie");
tom.next = ellie;
ellie.prev  tom;

Node<String> harry = new Node<>("Harry");
ellie.next = harry;
harry.prev = ellie;

Node<String> sam = new Node<>("Same");
harry.next = sam;
same.prev = harry;
```



LinkedList

- Insert an element
  - Insert a new element after the node “ellie”

```
Node<String> jonh = new Node<>("Jonh");
jonh.next = ellie.next; //step 1
jonh.prev = ellie; //step 2
ellie.next.prev = jonh; //step 3
ellie.next = jonh; //step 4
```

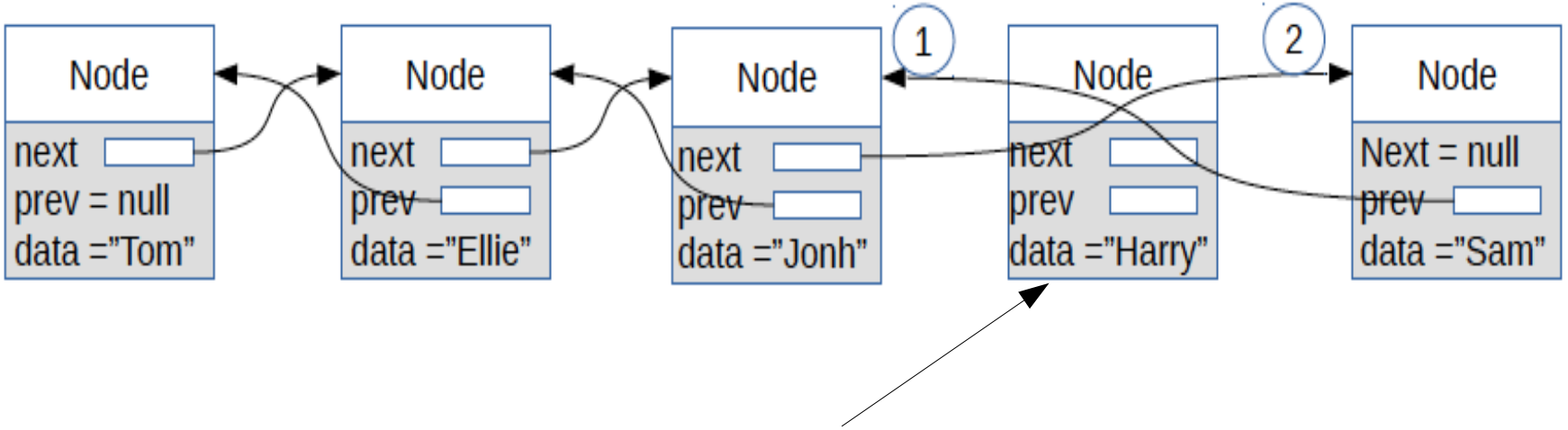


LinkedList

- Delete an element
  - Remove element “harry” from LinkedList

    harry.prev.next = harry.next; // Step 1

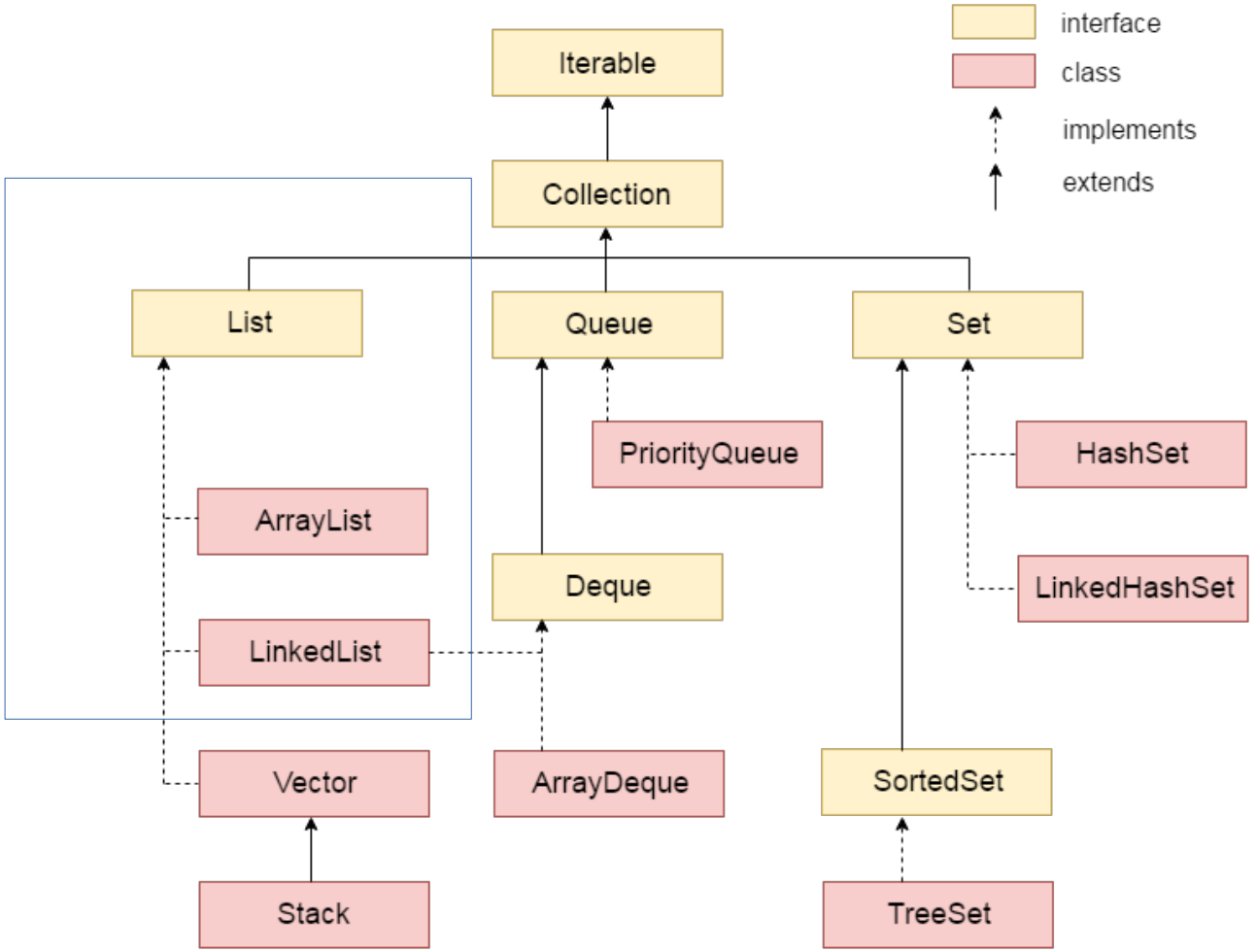
    harry.next.prev = harry.prev; // Step 2



Node “harry” Doesn't belong to the linkedList anymore

LinkedList Class

Like ArrayList, LikedList implement List interface



LinkedList Class

- Create a LinkedList of String

```
LinkedList<String> linkedlist=new LinkedList<String>();
linkedlist.add("Apple");
linkedlist.add("Orange");
linkedlist.add("Mango");
linkedlist.add("Banana");
...
```

LinkedList Class

- Get element at index ”i”

  String fruit = linkedlist.get(i);

  But what really happen when we want to access to an element by index?

  - In reality, LinkedList does not provide the possibility to access to a random element by index.
  - If we assume that the method get begins at the first list node (head), each call to method get must advance a local reference (nodeRef) to the node at position index using a loop such as:

```
Node<E> nodeRef = head;

for (int j = 0; j < i; j++) {
      nodeRef = nodeRef.next;
}
```

LinkedList

- How to loop LinkedList in Java

For loop:

```
for(int num=0; num<linkedlist.size(); num++)
{
    System.out.println(linkedlist.get(num));
}
```

For each:

```
for(String str: linkedlist)
{
    System.out.println(str);
}
```

Using Iterator:

```
Iterator i = linkedlist.iterator();
while (i.hasNext()) {
    System.out.println(i.next());
}
```

LinkedList

- Conclusion
  - Add or delete an element in an ArrayList is slower than LinkedList
  - However, access to an element at index i in an LinkedList is a time consuming process

References

- https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
- https://www.tutorialspoint.com/java/java_linkedlist_class.htm
- http://beginnersbook.com/2013/12/how-to-loop-linkedlist-in-java/