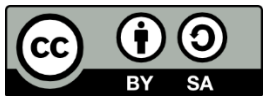
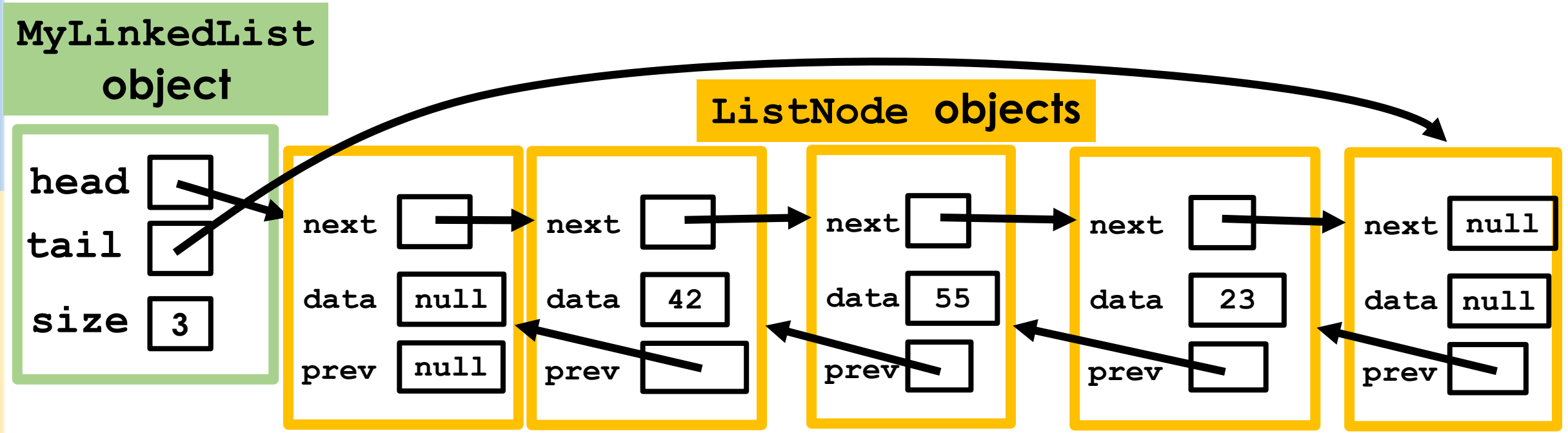
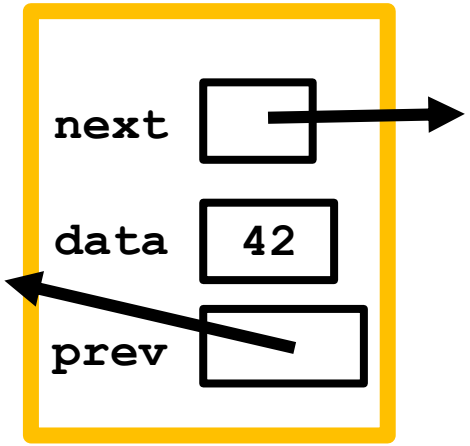
Generics and Exceptions



Implementing a Linked List in Java



```
class ListNode<E> {
    ListNode<E> next;
    ListNode<E> prev;
    E data;
```



What is E??

A parameterized type
Our ListNode is "generic"

Example: Parameterized types

```
public class RememberLast<T> {
    private T lastElement;
    private int numElements;

    public RememberLast ()
    {
        numElements = 0;
        lastElement = null;
    }

    public T add( T element )
    {
       T prevLast = lastElement;
       lastElement = element;
       numElements++;
       return prevLast;
    }
    …
```

```
// Somewhere else…
RememberLast<Integer> rInt =
    new RememberLast<Integer>();
RememberLast<String> rStr =
    new RememberLast<String>();

rInt.add(3);
rStr.add("Happy");
```

Handling bad Input

```
public class RememberLast<T> {

    public T add( T element )
    {
        if (element == null) {
            <<WHAT GOES HERE?>>
        }
        T prevLast = lastElement;
        lastElement = element;
        numElements++;
        return prevLast;
    }
    …
```

Handling bad input

```
public class RememberLast<T> {

    public T add( T element )
    {
        if (element == null) {
            return -1;
        }
        T prevLast = last
        lastElement = element;
        numElements++;
        return prevLast;
    }
    …
```

Doesn't work! Must return a T

Throw exceptions to indicate fatal problems

```
public class RememberLast<T> {

    public T add( T element )
    {
        if (element == null) {
            throw new NullPointerException("RememberLast object cannot
store null pointers.")
        }
        T prevLast = lastElement;
        lastElement = element;
        numElements++;
        return prevLast;
    }
    …
```

Checked exceptions must be declared

```
public class RememberLast<T> {

    public T add( T element ) throws NullPointerException
    {
        if (element == null) {
            throw new NullPoir
store null pointers.")
        }
        T prevLast = lastElement;
        lastElement = element;
        numElements++;
        return prevLast;
    }
    …
```

Not required, since NPE is unchecked, but OK.