CMPT 280

Topic 6: Cloning

Mark G. Eramian

University of Saskatchewan

References

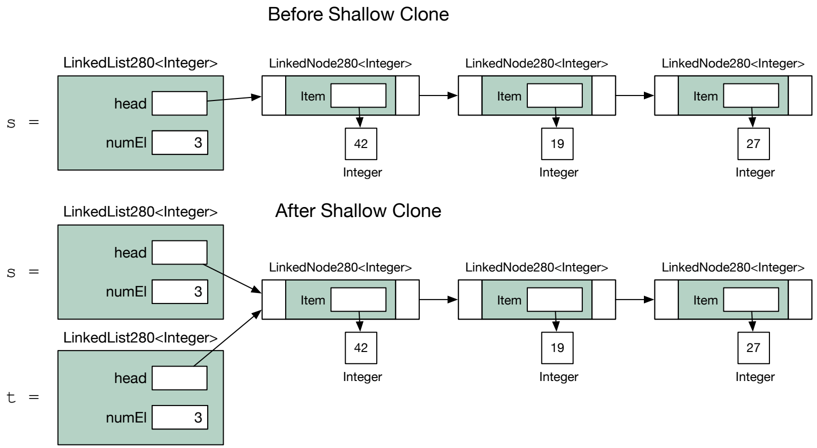
- Textbook, Chapter 6

Shallow vs Deep Clone

- Shallow clone copies only the object's data and references.
- Deep clone copies all of the other objects reference directly or indirectly by the object being cloned.
- Suppose `s` is a reference to a `LinkedNode280<Integer>` object. We want to be able to make a copy of `s` by writing:

```
LinkedNode280<Integer> t = s.clone()
```

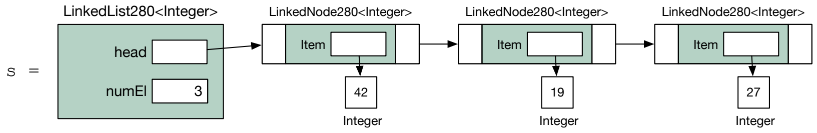
Shallow Clone



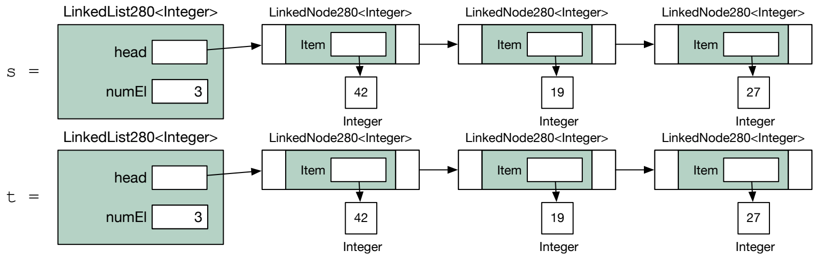
Only the `LinkedList280` object gets duplicated in a shallow clone.

Deep Clone

Before Deep Clone



After Deep Clone



All referenced objects get duplicated in a deep clone.

Cloning in Java

- In Java, the `Object` class has a method called `clone()` which can make a shallow clone of the object.
- It would seem, then, that every Java class supports shallow cloning automatically.
- But let's take a closer look at the documentation for `Object`...

Cloning in Java

- Whoops – clone() is a `protected` method in `Object`! That means another object can't call clone(). So what good is it?
- It is protected because Java insists that you grant explicit permission that it is safe to shallow clone your object by just copying its instance variables.
- This permission is granted by having your objects implement the `Cloneable` interface. Let's take a look...

Cloning in Java

- Classes that implement `Cloneable` must implement a public method called `clone`.
- This can either override the `clone` method in `Object`, or call it explicitly (which is allowed, because protected methods can be called by members of the same class or its descendants!).

Exercise 1

- Make `LinkedNode<I>` cloneable (shallow clone).
- Make `LinkedList<I>` cloneable (shallow clone).

Deep Clones in Java

- Deep clones are enabled in Java in much the same way as shallow clones.
- Instead of calling the protected `clone` method in `Object` from our public `clone` method, we write our own custom code, appropriate to the data structure, to construct a deep clone.

Exercise 2

- Revise the `clone()` methods `LinkedNode<I>` and `LinkedList<I>` so that they are deep clones.
- Verify that the clone is a deep clone by cloning a list, removing a node from the clone, and comparing it to the original list (which should be unchanged).

Next Class

- Next class reading: Chapter 7: Abstract Data Types