Generics (cont.)

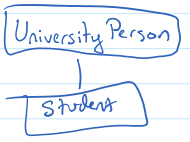
Thursday, September 3, 2015 11:31 AM

Outline
- Generics
- Interfaces (revisited)
- Object References
- Lists → The ArrayList

Lab "BinarySearcher.java"

Subclass or inheriting class

Subclasses "extend" a class



Student extends University Person

Interfaces

Cannot instantiate an interface

```
public interface Number... {
    :
}
```

~~Number num = new Number()~~

```
Number num = new Integer();

public class Integer implements Number, ... {
    :
}

public class Box<E extends Integer> {
    :
}
```

The interface Comparable<E>:

```
public integer compareTo(E that) {
    : ?
```

Integer implements Comparable<Integer>, Comparable<Double>, ...

↑ must be comparable

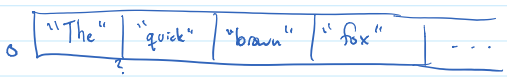
Problem: We want to compare two boxes (their contents)

```
public class Box<E extends Comparable<E>> {
    :
    E obj
```

obj.compareTo(...)

}

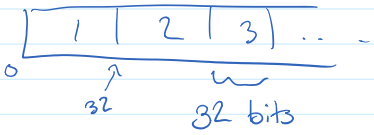
Arrays in Detail



Arrays in general, are not contiguous

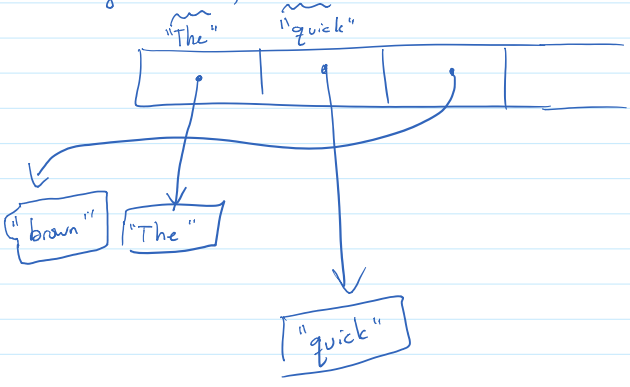
Arrays are only contiguous when they are over primitive types

int[]

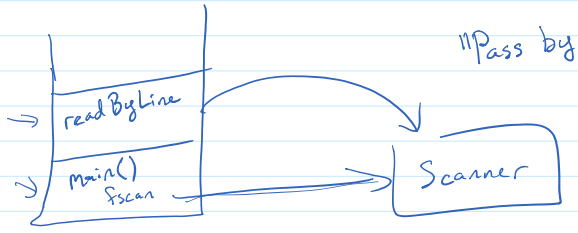



```
int[] arr;
:
arr[0]  // compiler converts this into
        // the first mem addr. of the array

arr[2] // 2*sizeof(int)
```

String[] arr;




```
main() {
    Scanner fscan = ...
    :
    readByLine(fscan);
    :
```



Alternatively "Pass by Object"

References or Pointers

The List (ArrayList)

Problem: size

A list is like an array in that:
- 1-dimensional
- All elements must be of the same type

Guarantees of Arrays, and not of Lists:
- Random Access $O(1)$ "constant time"
- Fixed size

An empty [ ] list

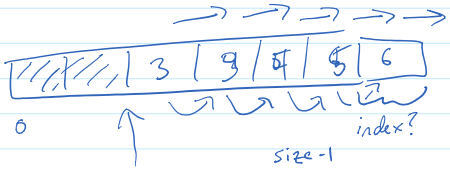
After inserting (3,2,1) [3,2,1]

Methods on Lists:

List interface
- Insert (int index, E elem) → void
- Append (E elem) → void
- remove (int index) → E
- get (int index) → E
- size() → int

```
public interface List<E> {
    /** ... */
    public void insert(int index, E elem);
    /** ... */
    public void append(E elem);
    /** ... */
    public E remove(int index);
    /** ... */
    public E get(int index);
    /** ... */
    public int size();
}

public class ArrayList<E> implements List<E> {

    Object[] theArray; // E[] can work
    int size;

    public ArrayList() {
        size = 0;
        theArray = new Object[2];
    }
    @SuppressWarnings("unchecked")
    private E convertElement(int index) {
        return (E) theArray[index];
    }

    public void insert(int index, E elem) {
        if (size >= theArray.length) {
            expandArray(); // "Wishful thinking"
        }
        for (int i=size; i > index; i--) {
```

Exception code



```
                expandArray();  // "Wishful thinking"
            }
            for (int i = size; i > index; i--) {
                theArray[i] = theArray[i-1];
            }
            theArray[index] = elem;
            size++;
        }

        public void append(E elem) {
            insert(size, elem);
        }

        private void expandArray() {
            Object[] newArray = new Object[2 * theArray.length];
            for (int i = 0; i < size; i++) {
                newArray[i] = theArray[i];
            }
            theArray = newArray; // Object References!
        }
```