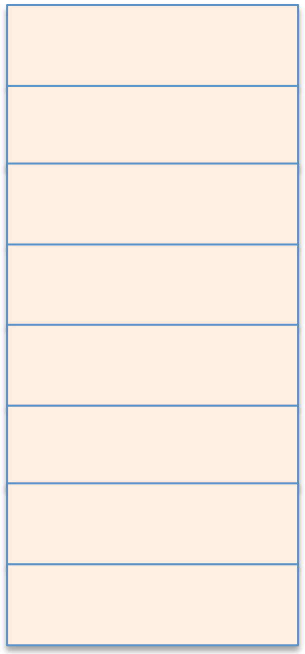
ArrayList and Generics
Why and How?

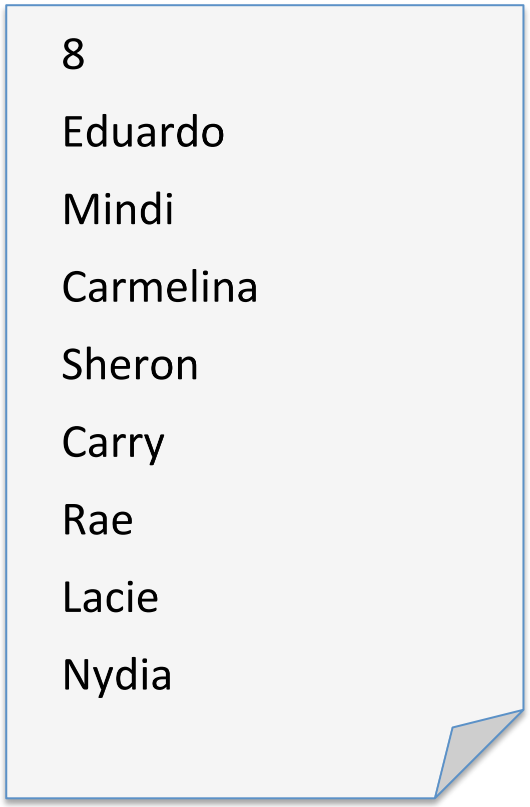
Barış Aktemur

Case Study

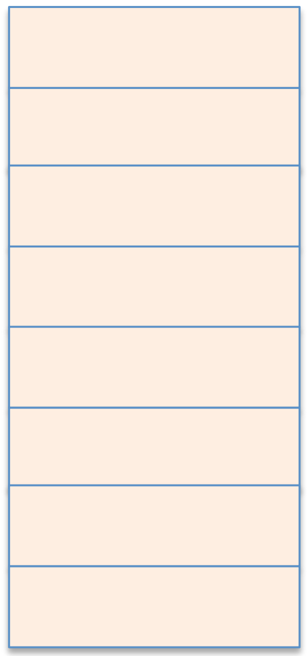
- Let’s start our discussion with a problem
- Read an unsorted list of names from a file and then print the names in alphabetical order

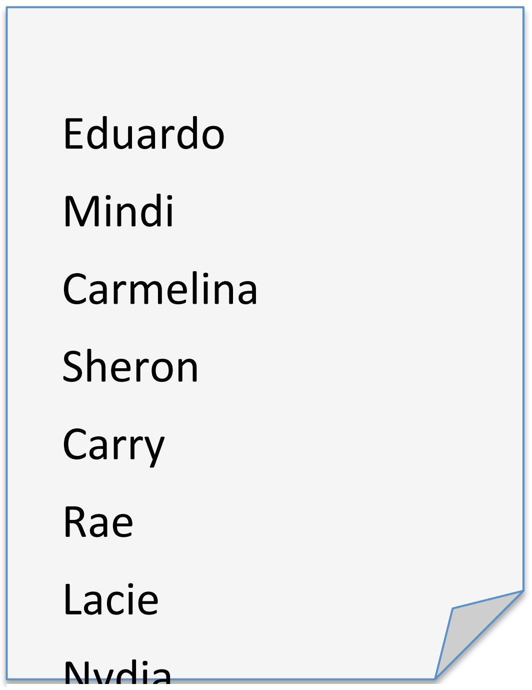
```
String[] names = new String[8];
```




8
Eduardo
Mindi
Carmelina
Sheron
Carry
Rae
Lacie
Nydia

```
String[] names = new String[4];
```





Eduardo

Mindi

Carmelina

Sheron

Carry

Rae

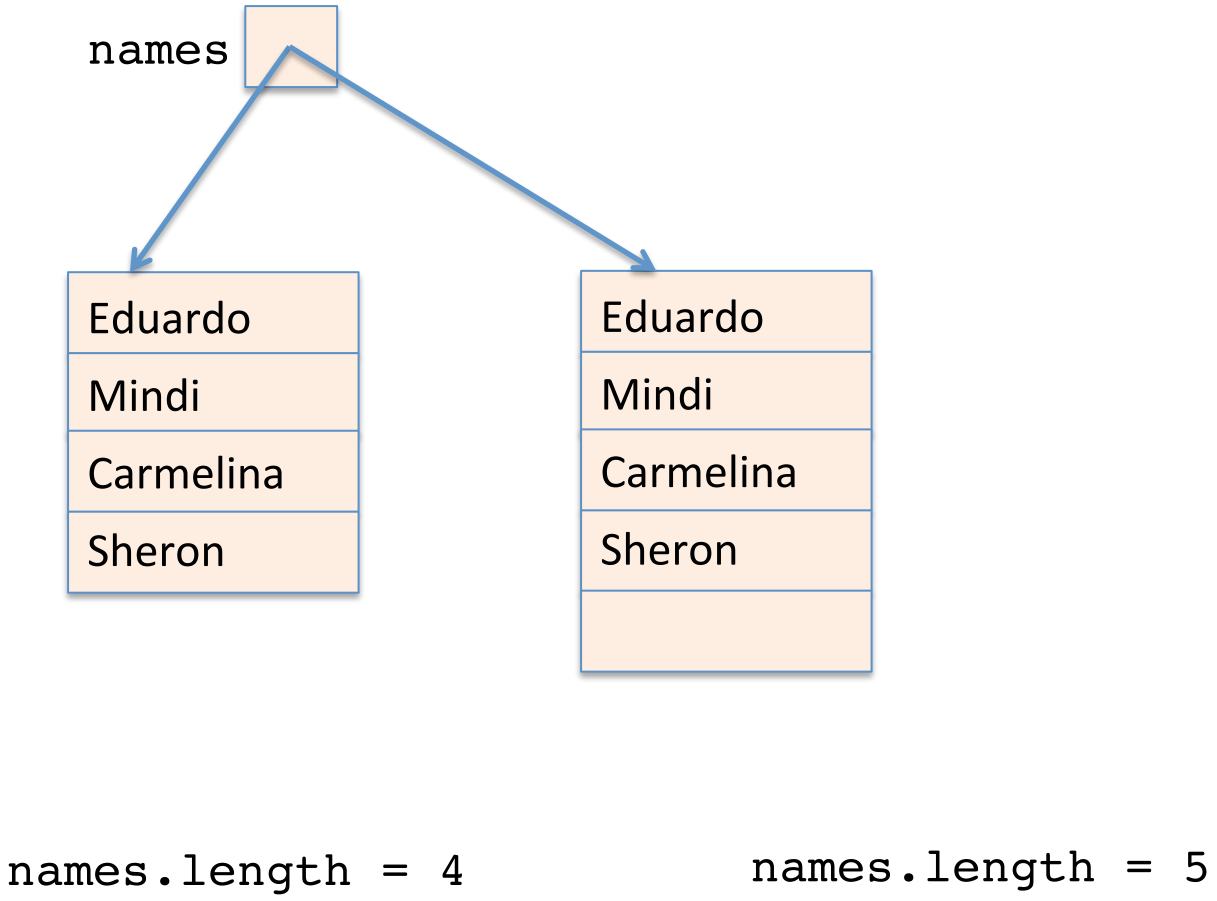
Lacie

Nydia

Array Expansion

- Arrays have fixed sizes. How do you extend them?
  1. Create a new array that has a bigger size.
  2. Copy all the existing items to the new array
  3. Move the array pointer to point to the new one

```
String[] names = new String[4];
...
```



This is an expensive operation!!!

How is it done in code

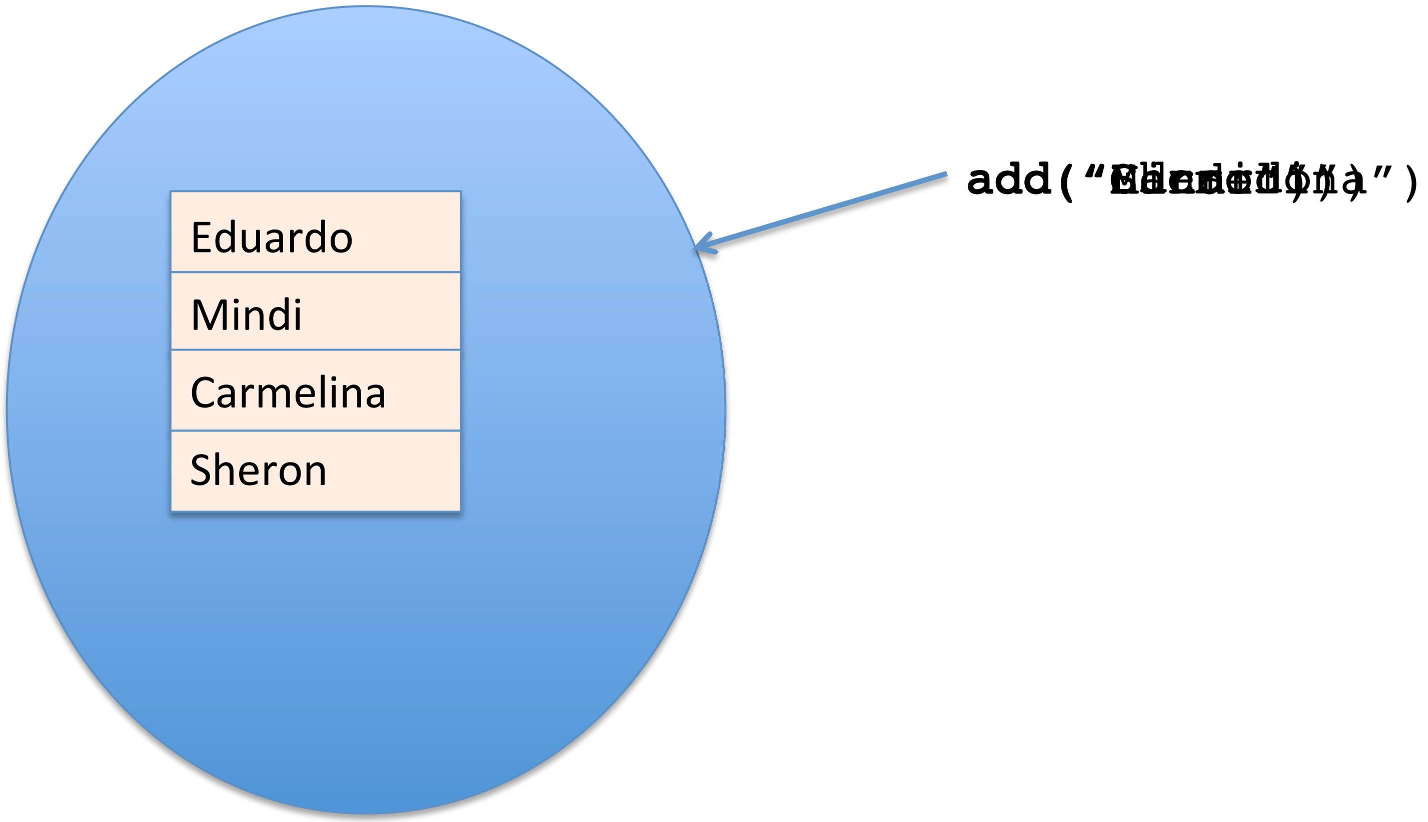
```
String[] names = new String[4];

// Expand names by one
String[] newNames = new String[names.length + 1];

// Copy each item one by one
for (int i = 0; i < names.length; i++) {
    newNames[i] = names[i];
}

// Make the array pointer point to the new array
names = newNames;
```

A better way…



```java
public class FlexibleStringArray {
    private String[] items;
    private int numberOfItems;

    public FlexibleStringArray(int initialSize) {
        items = new String[initialSize];
        numberOfItems = 0; // initially the container is empty
    }

    public void add(String newItem) {
        if (numberOfItems == items.length) {
            // no space left. Expand the array
            String[] newItems = new String[items.length + 1];
            for (int i = 0; i < items.length; i++) {
                newItems[i] = items[i];
            }
            items = newItems;
        }
        items[numberOfItems] = newItem;
        numberOfItems++;
    }

    public int getSize() {
        return numberOfItems;
    }
}
```

Using FlexibleStringArray

```
FlexibleStringArray names = new FlexibleStringArray(4);
```

instead of

```
String[] names = new String[4];
```

An optimization

```
if (numberOfItems == items.length) {
    // no space left. Expand the array
    String[] newItems = new String[items.length + 1];
    for (int i = 0; i < items.length; i++) {
        newItems[i] = items[i];
    }
    items = newItems;
}
```

After the array becomes full, each new item will force copy of the entire array.

```
if (numberOfItems == items.length) {
    // no space left. Expand the array
    String[] newItems = new String[items.length * 2];
    for (int i = 0; i < items.length; i++) {
        newItems[i] = items[i];
    }
    items = newItems;
}
```

Once you're at it, allocate more space with the anticipation that more elements will arrive.

Only Strings?

- What if you wanted to store Integer objects instead of Strings?
- How about Student objects? BankAccount? MatrixElement?

```java
public class FlexibleStringArray {
    private String[] items;
    private int numberOfItems;

    public FlexibleStringArray(int initialSize) {
        items = new String[initialSize];
        numberOfItems = 0; // initially the container is empty
    }

    public void add(String newItem) {
        if (numberOfItems == items.length) {
            // no space left. Expand the array
            String[] newItems = new String[items.length + 1];
            for (int i = 0; i < items.length; i++) {
                newItems[i] = items[i];
            }
            items = newItems;
        }
        items[numberOfItems] = newItem;
        numberOfItems++;
    }

    public int getSize() {
        return numberOfItems;
    }
}
```

```java
public class FlexibleStudentArray {
    private Student[] items;
    private int numberOfItems;

    public FlexibleStudentArray(int initialSize) {
        items = new Student[initialSize];
        numberOfItems = 0; // initially the container is empty
    }

    public void add(Student newItem) {
        if (numberOfItems == items.length) {
            // no space left. Expand the array
            String[] newItems = new Student[items.length + 1];
            for (int i = 0; i < items.length; i++) {
                newItems[i] = items[i];
            }
            items = newItems;
        }
        items[numberOfItems] = newItem;
        numberOfItems++;
    }

    public int getSize() {
        return numberOfItems;
    }
}
```

Generics

- FlexibleStringArray and FlexibleStudentArray are very similar.
- They also don’t make any particular use of the String or Student classes.
- We would have to duplicate code for each class that we want to write a FlexibleArray for.
- Use generics!
  - Parameterize a class on a type.

```
public class FlexibleArray<T> {
    private T[] items;
    private int numberOfItems;

    public FlexibleArray(int initialSize) {
        items = new T[initialSize];
        numberOfItems = 0; // initially the container is empty
    }

    public void add(T newItem) {
        if (numberOfItems == items.length) {
            // no space left. Expand the array
            T[] newItems = new T[items.length + 1];
            for (int i = 0; i < items.length; i++) {
                newItems[i] = items[i];
            }
            items = newItems;
        }
        items[numberOfItems] = newItem;
        numberOfItems++;
    }

    public int getSize() {
        return numberOfItems;
    }
}
```

Argh!!! Java compiler yells!

```
public class FlexibleArray<T> {
    private T[] items;
    private int numberOfItems;

    public FlexibleArray(int initialSize) {
        items = (T[])(new Object[initialSize]);
        numberOfItems = 0; // initially the container is empty
    }

    public void add(T newItem) {
        if (numberOfItems == items.length) {
            // no space left. Expand the array
            T[] newItems = (T[])(new Object[items.length + 1]);
            for (int i = 0; i < items.length; i++) {
                newItems[i] = items[i];
            }
            items = newItems;
        }
        items[numberOfItems] = newItem;
        numberOfItems++;
    }

    public int getSize() {
        return numberOfItems;
    }
}
```

Using FlexibleArray

```
FlexibleArray<String> names = new FlexibleArray<String>(4);
```

instead of

```
String[] names = new String[4];
```

```
FlexibleArray<Student> kids = new FlexibleArray<Student>(40);
```

instead of

```
Student[] kids = new Student[40];
```

ArrayList

- **java.util.ArrayList** does essentially what we did using FlexibleArray.