CS 145

Chapter 10 – Array Lists Introduction.

Day 2

COLLECTIONS

Warm up

- Currently I have the following:

```
String [] names = new String[25];
String [] IDs   = new String[25];
```

- What is a better way to do it so that I can make the class bigger or smaller as necessary? And what would the code look like?

Warm up Answers Option1

- What is a better way to do it so that I can make the class bigger or smaller as necessary? And what would the code look like?

```
ArrayList<String> names = new ArrayList<String>();
ArrayList<String> IDs = new ArrayList<String>();
```

Warm up Answers Option1

- What would be even better?

```
public Class Student
{
  private String name, ID;
}
```

- Then

```
ArrayList<Student> names = new ArrayList<Student>();
```

`ArrayList` of primitives?

- The type you specify when creating an `ArrayList` must be an object type; it cannot be a primitive type.

```
// illegal -- int cannot be a type parameter
ArrayList<int> list = new ArrayList<int>();
```

- But we can still use `ArrayList` with primitive types by using special classes called *wrapper* classes in their place.

```
// creates a list of ints
ArrayList<Integer> list = new
ArrayList<Integer>();
```

Wrapper classes

Primitive Type	Wrapper Type
int	Integer
double	Double
char	Character
boolean	Boolean

- A wrapper is an object whose sole purpose is to hold a primitive value.

- Once you construct the list, use it with primitives as normal:

```
ArrayList<Double> grades = new ArrayList<Double>();
grades.add(3.2);
grades.add(2.7);
...
double myGrade = grades.get(0);
```

Wrapper Weirdness.

```
Integer a = 100;  Integer b = 100;
Integer x = 2333; Integer y = 2333;

System.out.println(a);
System.out.println(b);
System.out.println(a == b);
System.out.println(x);
System.out.println(y);
System.out.println(x == y);
```

ArrayList as parameter

```
public static void name(ArrayList<Type> name) {
```

- Example:

```
// Removes all plural words from the given list.
public static void removePlural(ArrayList<String>
  list) {
    for (int i = 0; i < list.size(); i++) {
        String str = list.get(i);
        if (str.endsWith("s")) {
            list.remove(i);
            i--;
        }
    }
}
```

- You can also return a list:

```
public static ArrayList<Type> methodName(params)
```

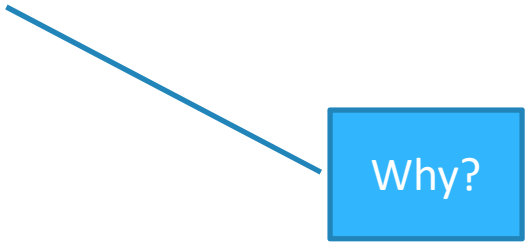
Exercise, revisited

- Write a program that reads a file and displays the words of that file as a list.
  - First display all words.
  - Then display only the words with the letter “E” in them.
  - Then display them in reverse order.
  - Then display them with all “E” words removed.

Exercise solution (partial)

```
ArrayList<String> allWords = new ArrayList<String>();
Scanner input = new Scanner(new File("words.txt"));
while (input.hasNext()) {
    String word = input.next();
    allWords.add(word);
}
System.out.println(allWords);

// remove all “E” words
for (int i = 0; i < allWords.size(); i++) {
    String word = allWords.get(i);
    if (word.toUpperCase().contains("E")) {
        allWords.remove(i);
        i--;
    }
}
```

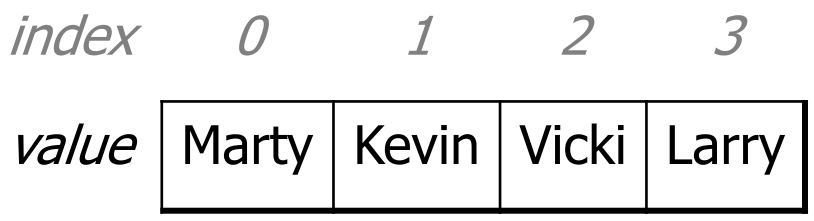


Out-of-bounds

Legal indexes are between **0** and the **list's size() - 1**.

- Reading or writing any index outside this range will cause an `IndexOutOfBoundsException`.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Marty");   names.add("Kevin");
names.add("Vicki");   names.add("Larry");
System.out.println(names.get(0));        // okay
System.out.println(names.get(3));        // okay
System.out.println(names.get(-1));       // exception
names.add(9, "Aimee");                   // exception
```



ArrayList "mystery"

```
ArrayList<Integer> list = new ArrayList<Integer>();
for (int i = 1; i <= 10; i++) {
    list.add(10 * i);   // [10, 20, 30, 40, ..., 100]
}
```

- What is the output of the following code?

```
for (int i = 0; i < list.size(); i++) {
    list.remove(i);
}
System.out.println(list);
```

- Answer:

```
[20, 40, 60, 80, 100]
```

ArrayList "mystery" 2

```
ArrayList<Integer> list = new ArrayList<Integer>();
for (int i = 1; i <= 5; i++) {
    list.add(2 * i);   // [2, 4, 6, 8, 10]
}
```

- What is the output of the following code?

```
int size = list.size();
for (int i = 0; i < size; i++) {
    list.add(i, 42);   // add 42 at index i
}
System.out.println(list);
```

- Answer:

```
[42, 42, 42, 42, 42, 2, 4, 6, 8, 10]
```

Exercise

- Write a method `addStars` that accepts an array list of strings as a parameter and places a `*` after each element.
  - Example: if an array list named `list` initially stores:
    `[the, quick, brown, fox]`
  - Then the call of `addStars(list);` makes it store:
    `[the, *, quick, *, brown, *, fox, *]`
- Write a method `removeStars` that accepts an array list of strings, assuming that every other element is a `*`, and removes the stars (undoing what was done by `addStars` above).

```
public static void addStars(ArrayList<String> list) {
    for (int i=1; i<=list.size(); i+=2) {
        list.add(i, "*");
    }
 }

public static void removeStars(ArrayList<String> list) {
    for (int i=1; i<=list.size(); i++) {
        list.remove(i);
    }
 }
```