

Scoot Over

ArrayList methods



The waiting list

PAUSE AND TRY

A fifth party walks in. Then Raj leaves. What do you have to do to this array?

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

waiting

Pia	Raj	Tam	Uma
0	1	2	3

Talk with a partner. Where does the fifth name go? What is left behind when Raj leaves?

By the end of class you can...

- 1 Import ArrayList and create an empty list of a chosen element type.
- 2 Use size, add, and get to build a list and read from it.
- 3 Insert with add(index, obj) and replace with set, and tell them apart.
- 4 Remove elements and predict how the later indexes shift.
- 5 Explain that a list holds references to objects.

A large, white, rounded square containing a large, bold, teal number '1'.

1

PART 1

A list that can grow

Making an empty list

EMPTYLIST.JAVA (IMPORT AND MAIN)

```
import java.util.ArrayList;

ArrayList<String> tasks =
    new ArrayList<String>();
System.out.println(tasks.size());
System.out.println(tasks);
```

OUTPUT

```
0
[]
```

An ArrayList is **mutable in size**: it can grow and shrink.

`import java.util.ArrayList;`
ArrayList is in java.util, so import it

`ArrayList<String>`

the type: a list whose elements are Strings

`new ArrayList<String>()`

the constructor: always builds an empty list

The element type must be a class



No primitives in the angle brackets. A list of numbers is an **ArrayList<Integer>** or an **ArrayList<Double>**.

DOES NOT COMPILE

```
ArrayList<double> prices =  
    new ArrayList<double>();
```

unexpected type:
required: reference
found: double

COMPILES

```
ArrayList<Double> prices =  
    new ArrayList<Double>();  
prices.add(2.5);
```

prints [2.5]

CAUGHT BY THE COMPILER

```
ages.add("ten");
```

(ages is an ArrayList<Integer>)

incompatible types:
String cannot be converted
to Integer

A large, white, rounded square containing the number '2' in a bold, teal, sans-serif font.

2

PART 2

add, size, and get

Append, count, and look

GROCERY.JAVA (MAIN)

```
ArrayList<String> list =  
    new ArrayList<String>();  
list.add("milk");  
list.add("eggs");  
list.add("bread");  
System.out.println(list);  
System.out.println(list.size());  
System.out.println(list.get(0));  
System.out.println(list.get(list.size() - 1));  
System.out.println(list.add("jam"));  
System.out.println(list);
```

OUTPUT

```
[milk, eggs, bread]  
3  
milk  
bread  
true  
[milk, eggs, bread, jam]
```

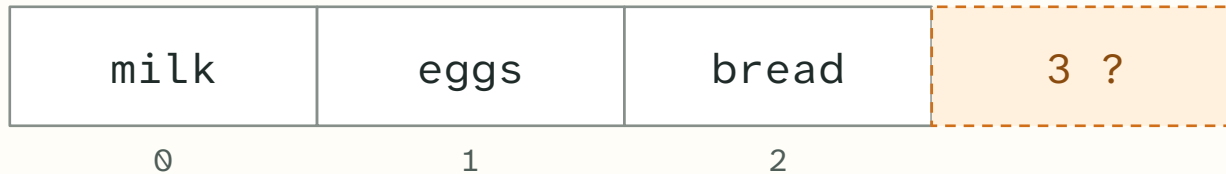
Method	Returns
add(obj)	true
size()	number of elements
get(index)	element at index

size() is not a valid index



Valid indexes run from **0** to **size() - 1**. Anything else throws an **IndexOutOfBoundsException**.

list



```
System.out.println(list.get(list.size()));
```

CRASH

```
Exception in thread "main" java.lang.IndexOutOfBoundsException:  
Index 3 out of bounds for length 3
```

Which hat?

PAUSE AND TRY

Write the exact output.

```
ArrayList<String> hats =  
    new ArrayList<String>();  
hats.add("cap");  
hats.add("beanie");  
hats.add("fedora");  
System.out.println(hats.get(1) + " " + hats.size());  
System.out.println(hats.get(hats.size() - 2));
```

A large, white, rounded square containing the number '3' in a bold, teal, sans-serif font.

3

PART 3

Insert or replace?

add(index, obj) inserts and scoots

INSERT.JAVA (MAIN)

```
ArrayList<String> colors =  
    new ArrayList<String>();  
colors.add("red");  
colors.add("blue");  
colors.add("green");  
System.out.println(colors);  
colors.add(1, "gold");  
System.out.println(colors);
```

OUTPUT

```
[red, blue, green]  
[red, gold, blue, green]
```

Before

red	blue	green
0	1	2

After add(1, "gold")

red	gold	blue	green
0	1	2	3

Everything from the index on moves **right** (index + 1). Size goes **up by 1**. Valid index: 0 to size().

KEY IDEA

set(index, obj) replaces

SETDEMO.JAVA (MAIN)

```
ArrayList<String> colors =  
    new ArrayList<String>();  
colors.add("red");  
colors.add("gold");  
colors.add("blue");  
colors.add("green");  
String old = colors.set(2, "pink");  
System.out.println(old);  
System.out.println(colors);
```

OUTPUT

```
blue  
[red, gold, pink, green]
```

Before

red	gold	blue	green
0	1	2	3

After set(2, "pink")

red	gold	pink	green
0	1	2	3

	add(i, x)	set(i, x)
Size	+ 1	same
Returns	nothing (void)	the old element
Valid i	0 to size()	0 to size() - 1

Track the list

PAUSE AND TRY

Draw the list after each line. What prints?

```
1  ArrayList<String> c =  
2      new ArrayList<String>();  
3  c.add("x");  
4  c.add("y");  
5  c.add(0, "w");  
6  c.set(1, "v");  
7  c.add(c.size(), "z");  
8  System.out.println(c);  
9  System.out.println(c.size());
```

Hint: ask "does this make room, or overwrite?" for every call with an index.

A large, bold, teal number '4' is centered within a white rounded square that has a slight 3D effect and a shadow.

PART 4

Remove, and what a list holds

remove(index) closes the gap

REMOVEDEMO.JAVA (MAIN)

```
ArrayList<String> pets =  
    new ArrayList<String>();  
pets.add("kit");  
pets.add("pup");  
pets.add("cub");  
pets.add("joey");  
String gone = pets.remove(1);  
System.out.println(gone);  
System.out.println(pets);  
System.out.println(pets.get(1));
```

Before

kit	pup	cub	joey
0	1	2	3

After remove(1)

kit	cub	joey
0	1	2

Later elements move **left** (index - 1). Size goes **down by 1**.

OUTPUT

```
pup  
[kit, cub, joey]  
cub
```

remove takes an index, even for numbers



The Quick Reference method is **remove(int index)**. The argument is always a position.

REMOVEINT.JAVA (MAIN)

```
ArrayList<Integer> nums =  
    new ArrayList<Integer>();  
nums.add(7);  
nums.add(2);  
nums.add(9);  
nums.remove(2);  
System.out.println(nums);
```

Before

7	2	9
0	1	2

After remove(2)

7	2
0	1

The 2 is still there. Index 2 is gone.

OUTPUT

[7, 2]

A list holds references

CLASS LAMP

```
class Lamp {  
    private boolean on;  
    public void toggle() {  
        on = !on;  
    }  
    public boolean isOn() {  
        return on;  
    }  
}
```

OUTPUT

```
true  
true  
false
```

IN MAIN

```
ArrayList<Lamp> room =  
    new ArrayList<Lamp>();  
Lamp desk = new Lamp();  
room.add(desk);  
desk.toggle();  
System.out.println(room.get(0).isOn());  
desk = new Lamp();  
System.out.println(room.get(0).isOn());  
System.out.println(desk.isOn());
```

add stores a copy of the **reference**. A change through desk shows through room.get(0); pointing desk at a new Lamp does not change the list.

Remove, then replace

PAUSE AND TRY

Draw the list after each line. What prints?

```
1  ArrayList<String> q =  
2      new ArrayList<String>();  
3  q.add("kit");  
4  q.add("pup");  
5  q.add("cub");  
6  q.add("joey");  
7  q.remove(2);  
8  q.set(0, q.remove(1));  
9  System.out.println(q);  
10 System.out.println(q.size());
```

Hint: in the set line, the argument `q.remove(1)` runs before set does.

Words to know

ArrayList

A list of object references that can grow and shrink. In java.util.

Type parameter

The element type in angle brackets: `ArrayList<String>`.

size()

Returns the number of elements. Arrays use length instead.

Insert

`add(index, obj)`: later elements move right; size + 1.

Replace

`set(index, obj)`: nothing moves; returns the old element.

IndexOutOfBoundsException

Thrown when an index is outside the list's valid range.

Who moved?



A list holds 5 names. Explain to your partner what happens to the name at index 3 after remove(1).

SENTENCE STARTER

After remove(1), the name that was at index 3 is now at index _____,
because _____.

Next: the Scoot Over puzzle worksheet.