**Kaitlin Hoffmann**

**Office Hours:**

SH 243 MR 11:00 - 12:30 PM via appointment https://calendly.com/hoffmank4/15min

**Email:** hoffmank4@newpaltz.edu

**For TA Office Hours and Email –** Please see syllabus

**NEW! Supplemental Instruction:** Sign up at my.newpaltz.edu

## ARRAYLISTS

# COMPUTER SCIENCE I

## OBJECTIVES

- Math.random()
- Wrapper Classes
- ArrayList

## CREATING RANDOM NUMBERS

- Random numbers can be useful in a lot of situations.
  - Maybe you're creating a game where the enemy hits you and deals a random amount of damage.
  - Maybe you're creating a quiz program that picks a random question that a user has to answer.
- We can use Math.random() from the Java Math class to get random numbers!

## CREATING RANDOM NUMBERS

- Math.random() returns a random double from the interval [0,1) ( inclusive 0, exclusive 1 )
- Example:

```java
double r = Math.random();
for(int i = 0; i < 5; i++) {
    System.out.println(r);
    r = Math.random();
}
```

**OUTPUT:**

```
0.2098007923795917
0.6243700884001604
0.841828017661504
0.10150071260827487
0.008887982570412456
```

## CREATING RANDOM NUMBERS - INTEGERS

- What if we want a random integer instead of a double? Use **casting**. However, since it's [0, 1), we will only get a bunch of zeros!
- Example:

```java
int r = (int)Math.random();
for(int i = 0; i < 5; i++) {
    System.out.println(r);
    r = (int)Math.random();
}
```

**OUTPUT:**

```
0
0
0
0
0
```

**Not very useful!!**

## CREATING RANDOM NUMBERS - INTEGERS

- To get an integer in a specific interval, use the following equation:

$$(int)(Math.random() * (\boldsymbol{max} - \boldsymbol{min})) + \boldsymbol{min}$$

- In other words, you multiply Math.random() by the difference and add the smaller value.
- If the interval is **[a, b]** just convert to **[a, b+1)** first and then follow the same method.

## CREATING RANDOM NUMBERS – INTEGERS

*(int)(Math.random()* ∗ *(**max** – **min**))* + ***min***

- Ex 1: [1, 10)

```java
int r = (int)(Math.random() * (10 - 1)) + 1;
for(int i = 0; i < 5; i++) {
    System.out.println(r);
    r = (int)(Math.random() * (10 - 1)) + 1;
}
```

```
OUTPUT:
  9
  2
  8
  6
  9
```

## CREATING RANDOM NUMBERS – INTEGERS

*(int)(Math.random() ∗ (**max – min**)) + **min***

- Ex 2: [200, 250)

```
int r = (int)(Math.random() * (250 - 200)) + 200;
for(int i = 0; i < 5; i++) {
    System.out.println(r);
    r = (int)(Math.random() * (250 - 200)) + 200;
}
```

```
OUTPUT:
  238
  223
  219
  220
  216
```

## EXAMPLE 1

$$(int)(Math.random() * (\boldsymbol{max} - \boldsymbol{min})) + \boldsymbol{min}$$

- Let's print out 5 random integers in the interval **[500, 700)** using Math.random() and a loop.

## EXAMPLE 1

$$(int)(Math.random() * (\mathbf{max} - \mathbf{min})) + \mathbf{min}$$

- [500, 700)

```
int r = (int)(Math.random() * (700 - 500)) + 500;
for(int i = 0; i < 5; i++) {
    System.out.println(r);
    r = (int)(Math.random() * (700 - 500)) + 500;
}
```

```
OUTPUT:
  516
  501
  657
  529
  554
```

## WRAPPER CLASSES — PRIMITIVE TO OBJECTS

- A **primitive** type value can be automatically converted to an **object** using a wrapper class, and vice versa, depending on the context.
- Converting a primitive value to a wrapper object is called **boxing**. The reverse conversion is called **unboxing**.
- Each primitive type in Java has a **wrapper class**.

## WRAPPER CLASSES — PRIMITIVE TO OBJECTS

- Each primitive type in Java has a **wrapper class**:

| | | |
|---|---|---|
| int | → | Integer |
| double | → | Double |
| char | → | Character |
| boolean | → | Boolean |

etc.

## BENEFITS OF WRAPPER CLASSES

- Why use a wrapper class for primitive types? **So we can use methods!**
- Primitive data types don't have methods we can use on them. <u>HOWEVER</u>, objects do!
  - There may be times where a method will be useful to use on an integer, double, etc. which we will see later on when reading files.
- Another reason – some data structures can only hold objects, NOT primitive types. (We will see this with ArrayLists in a second…)

## OKAY, NOW WHAT?

- Okay, so how do we convert a primitive into a wrapper class? **Easy!** Just declare the wrapper class like you would normally do with any other data type:

```
Integer x = 5;

Double y = 3.25;

Boolean b = true;

Character c = ‘g’;
```

- We will get into some methods later on.

## ARRAY LIST - ANOTHER DATA STRUCTURE AND OBJECT!

- An **ArrayList** object can be used to store a list of **objects**.
- You can create an array to store objects. But, once the array is created, its size is fixed.
- Java provides the ArrayList class, which can be used to store an ***unlimited*** number of objects.
    - You can't store primitive data types in an ArrayList, thus the benefit of wrapper classes!

## ARRAY LIST – HOW TO USE IT

1. To use an ArrayList, we first need to import it just like with Scanner.
   - You can either import it by itself: **import java.util.ArrayList;**
   - OR use **import java.util.*;** which will import all packages under the java utility package (this includes Scanner!).
2. Then create your ArrayList. To create an ArrayList we say:

***ArrayList <TYPE> al = newArrayList <>();***

**Ex:** ArrayList<String> al = **new** ArrayList<>();

## ARRAY LIST – HOW TO USE IT

3. To add values, use the **add** method:

```
ArrayList<String> al = new ArrayList<>();
al.add("hello");
al.add("Goodbye");
al.add("what?");
```

4. To print out the ArrayList, you can use System.out.print:

```
System.out.println(al);
```

```
Output:
[hello, Goodbye, what?]
```

## ARRAY LIST – HOW TO USE IT

5. ArrayLists also start at an index of **0** just like arrays. To get a value from an index, use the **get** method:

```
al => [hello, Goodbye, what?]

        String s = al.get(1);

        System.out.println(s);
```

```
        Output:

        Goodbye
```

## ARRAY LIST - HOW TO USE IT

6. If we want to remove an object, use the **remove** method:

```
al => [hello, Goodbye, what?]

         String s = al.remove(1);

         System.out.println(s);

         System.out.println(al);

         Output:

         Goodbye

        [hello, what?]
```

## ARRAY LIST - HOW TO USE IT

7. If we want to get the amount of objects in an ArrayList, use the **size()** method:

```
al => [hello, Goodbye, what?]

        int len = al.size();

        System.out.println(len);


        Output:

        3
```

## ARRAY LIST – HOW TO USE IT

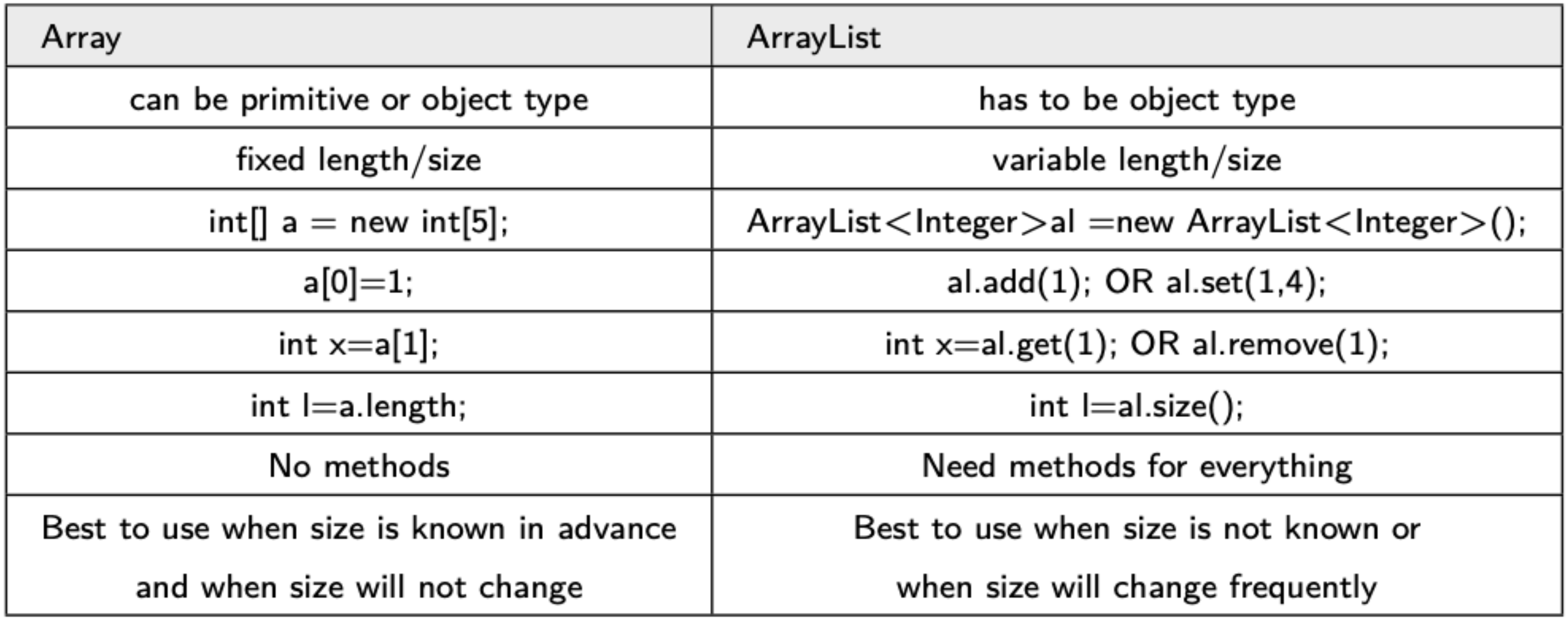
8. If we want to change a value in an ArrayList depending on its index, we use the **set()** method:

```
al => [hello, Goodbye, what?]

       al.set(1, "hi");

       System.out.println(al);


       Output:

       hello, hi, what?]
```

## ARRAY LIST – HOW TO USE IT

- There are many other methods you can use on an ArrayList. To see the list of methods, check out the Java Docs:

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html



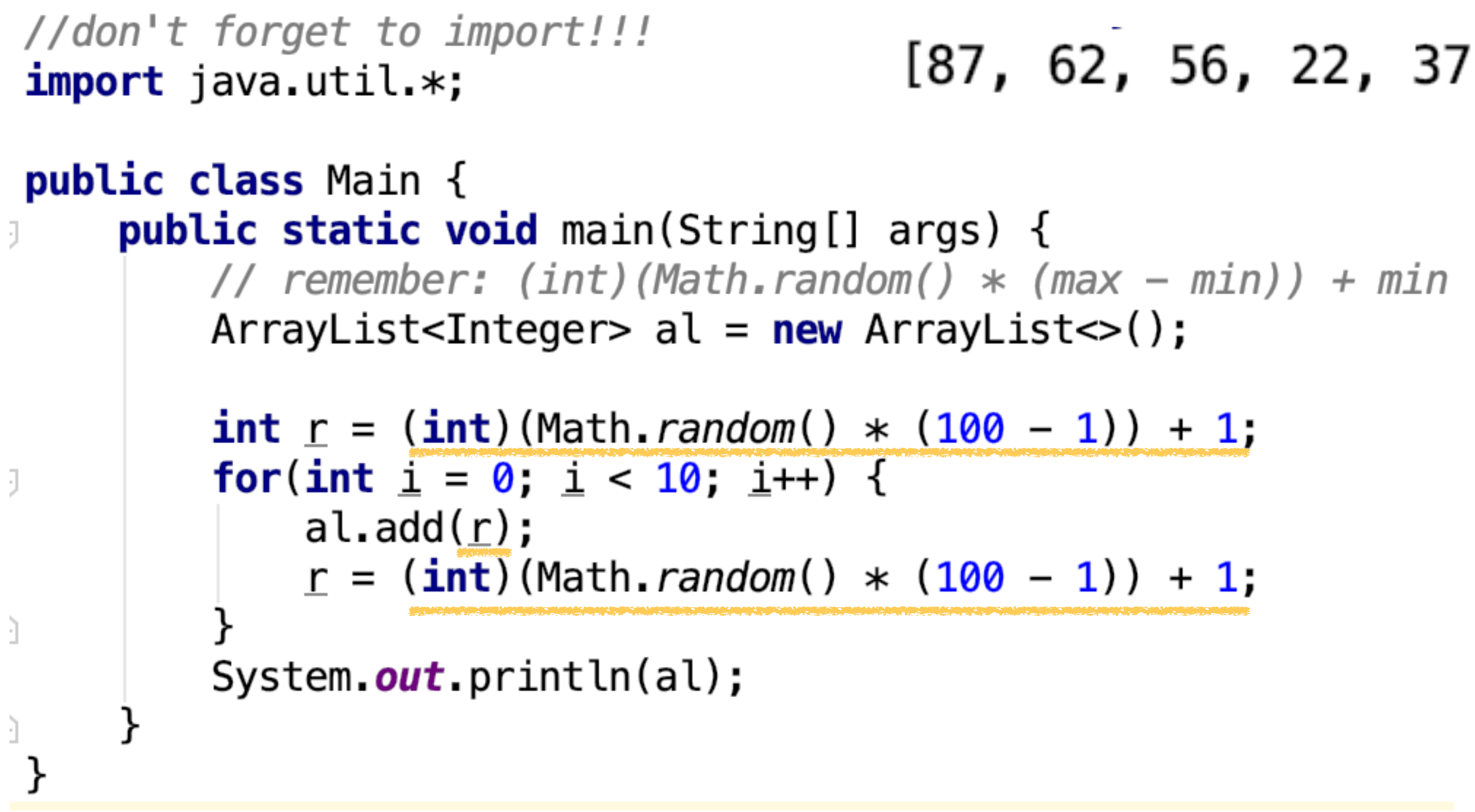
| Array | ArrayList |
|---|---|
| can be primitive or object type | has to be object type |
| fixed length/size | variable length/size |
| int[] a = new int[5]; | ArrayList<Integer>al =new ArrayList<Integer>(); |
| a[0]=1; | al.add(1); OR al.set(1,4); |
| int x=a[1]; | int x=al.get(1); OR al.remove(1); |
| int l=a.length; | int l=al.size(); |
| No methods | Need methods for everything |
| Best to use when size is known in advance and when size will not change | Best to use when size is not known or when size will change frequently |

## ARRAY LIST — EXAMPLE 1

- Let’s fill an ArrayList with 10 random integers from 1 through 100.

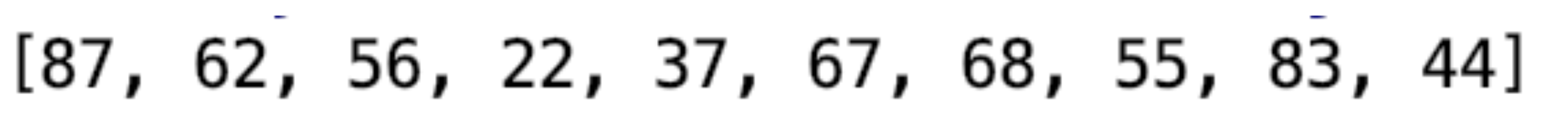
## ARRAY LIST — ONE WAY USING A VARIABLE

- Let's fill an ArrayList with 10 random integers from 1 through 100.



```
//don't forget to import!!!
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // remember: (int)(Math.random() * (max - min)) + min
        ArrayList<Integer> al = new ArrayList<>();

        int r = (int)(Math.random() * (100 - 1)) + 1;
        for(int i = 0; i < 10; i++) {
            al.add(r);
            r = (int)(Math.random() * (100 - 1)) + 1;
        }
        System.out.println(al);
    }
}
```

**Output**

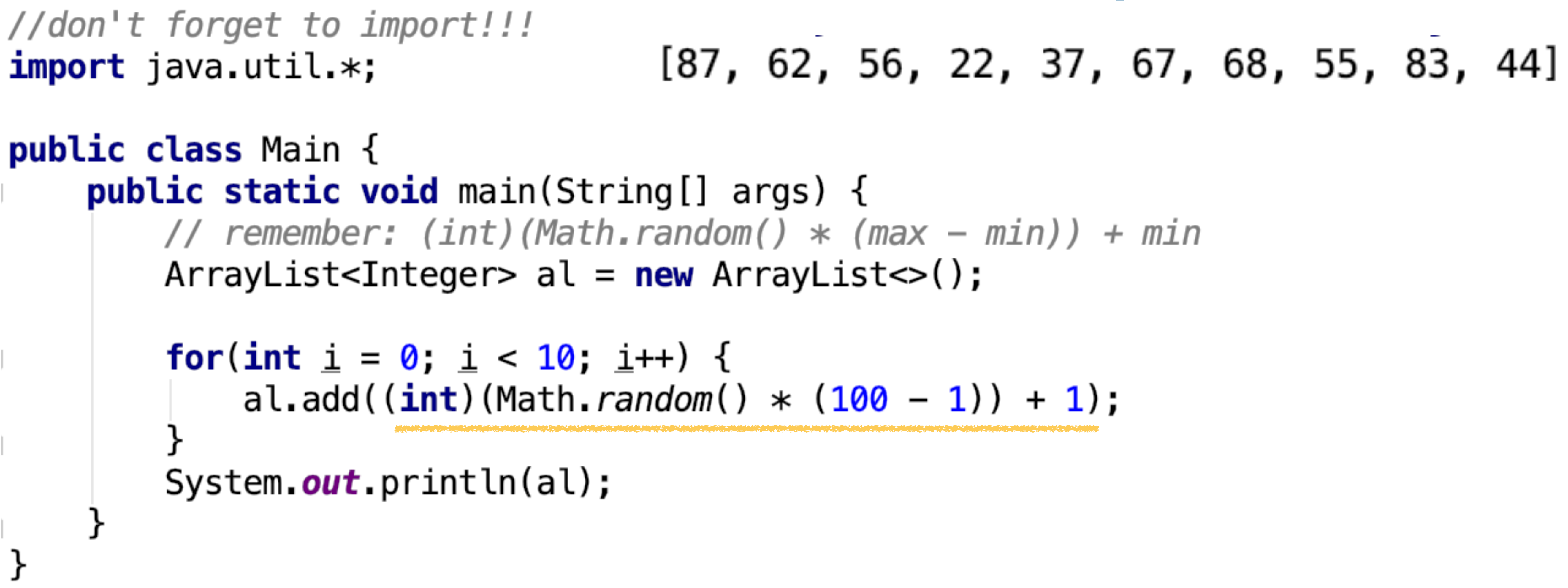


```
[87, 62, 56, 22, 37, 67, 68, 55, 83, 44]
```

## ARRAY LIST — ANOTHER WAY JUST INSERTING DIRECTLY

- Let’s fill an ArrayList with 10 random integers from 1 through 100.

**Output**



```
//don't forget to import!!!
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // remember: (int)(Math.random() * (max - min)) + min
        ArrayList<Integer> al = new ArrayList<>();

        for(int i = 0; i < 10; i++) {
            al.add((int)(Math.random() * (100 - 1)) + 1);
        }
        System.out.println(al);
    }
}
```

```
[87, 62, 56, 22, 37, 67, 68, 55, 83, 44]
```

## ARRAY LIST — AUTOBOXING AND AUTOUNBOXING

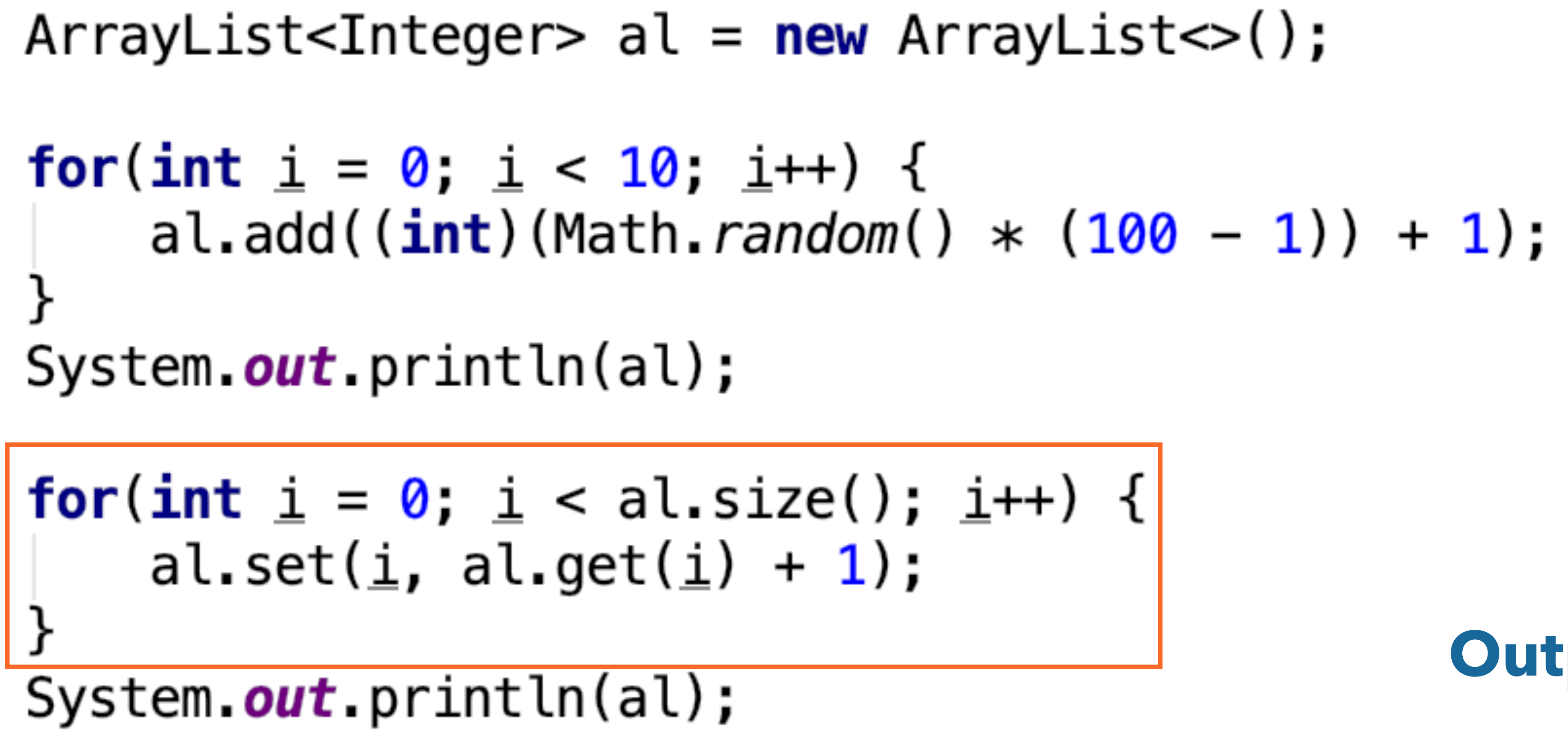
- Wait a second – why were we able to place a primitive int in an ArrayList of Integers (the wrapper class) in the exercise before?
- The compiler will automatically box a primitive value that appears in a context requiring an object, and will unbox an object that appears in a context requiring a primitive value.
  - This is called **autoboxing** and **autounboxing**.

## ARRAY LIST — EXAMPLE 2

- Let's take that same ArrayList and add 1 to every value.
- We will have to use the get method and set method

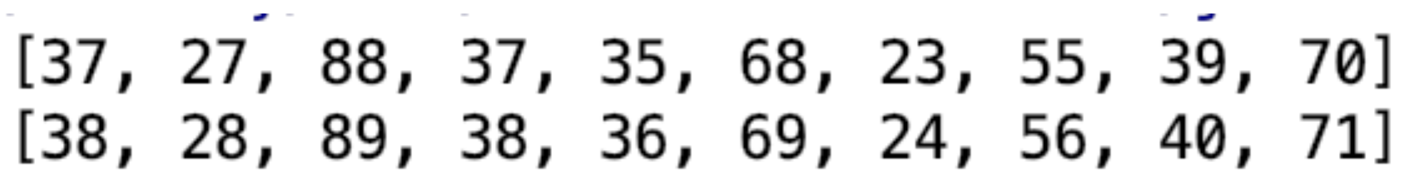
## ARRAY LIST — EXAMPLE 2

- Let's take that same ArrayList and add 1 to every value.
- We will have to use the get method and set method



```java
ArrayList<Integer> al = new ArrayList<>();

for(int i = 0; i < 10; i++) {
    al.add((int)(Math.random() * (100 - 1)) + 1);
}
System.out.println(al);

for(int i = 0; i < al.size(); i++) {
    al.set(i, al.get(i) + 1);
}
System.out.println(al);
```

**Output**



```
[37, 27, 88, 37, 35, 68, 23, 55, 39, 70]
[38, 28, 89, 38, 36, 69, 24, 56, 40, 71]
```

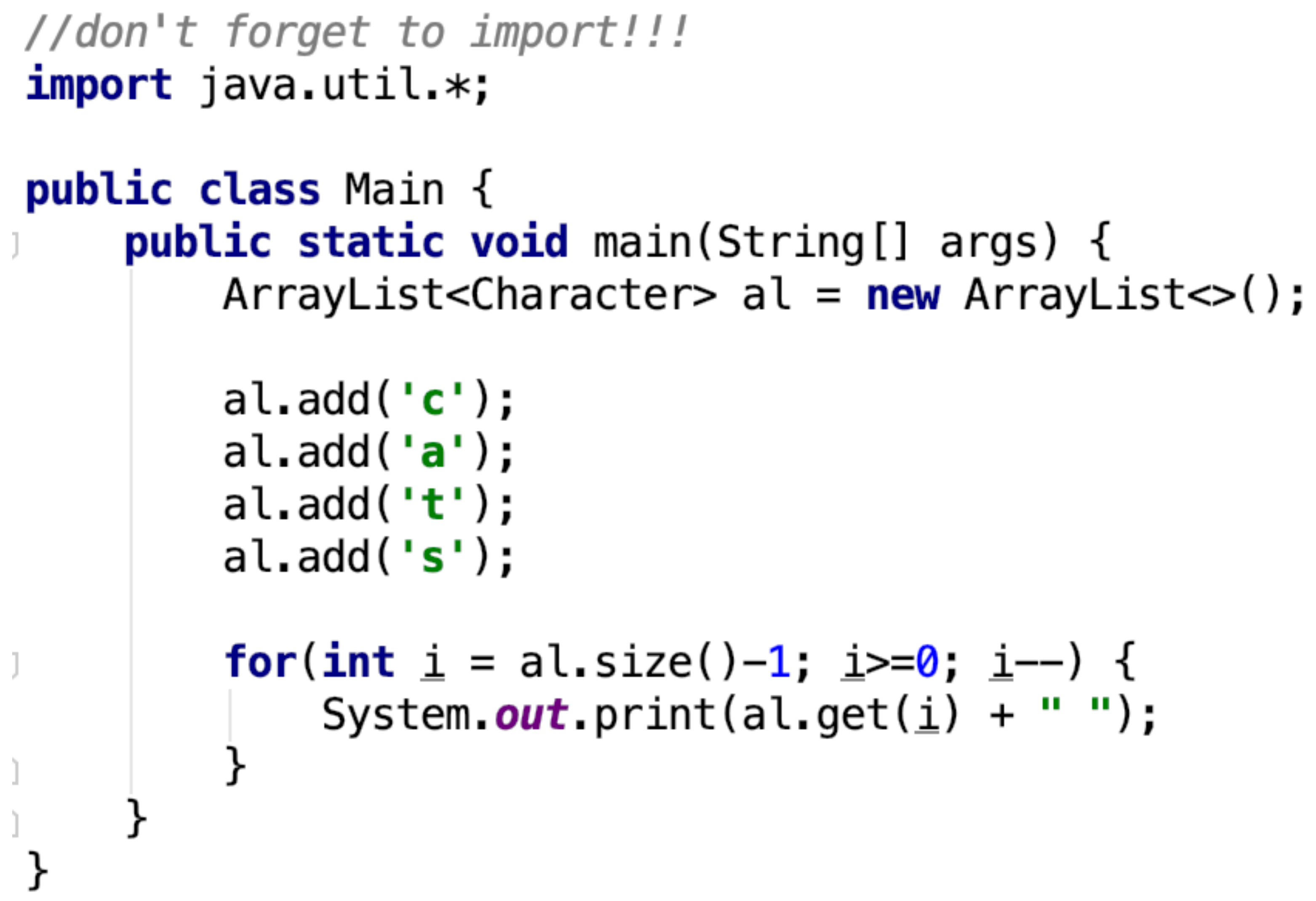
## ARRAY LIST — EXERCISE

- Create an ArrayList of **Characters**
- Add the following characters to it using the **add** method:

  **'c'**, **'a'**, **'t'**, **'s'**

- Print the ArrayList contents **backwards** using a for loop
  - HINT: You need to use the get() method instead array[] like you do with arrays.

## ARRAY LIST — EXERCISE



```
//don't forget to import!!!
import java.util.*;

public class Main {
    public static void main(String[] args) {
        ArrayList<Character> al = new ArrayList<>();

        al.add('c');
        al.add('a');
        al.add('t');
        al.add('s');

        for(int i = al.size()-1; i>=0; i--) {
            System.out.print(al.get(i) + " ");
        }
    }
}
```

**Output**

```
s t a c
```