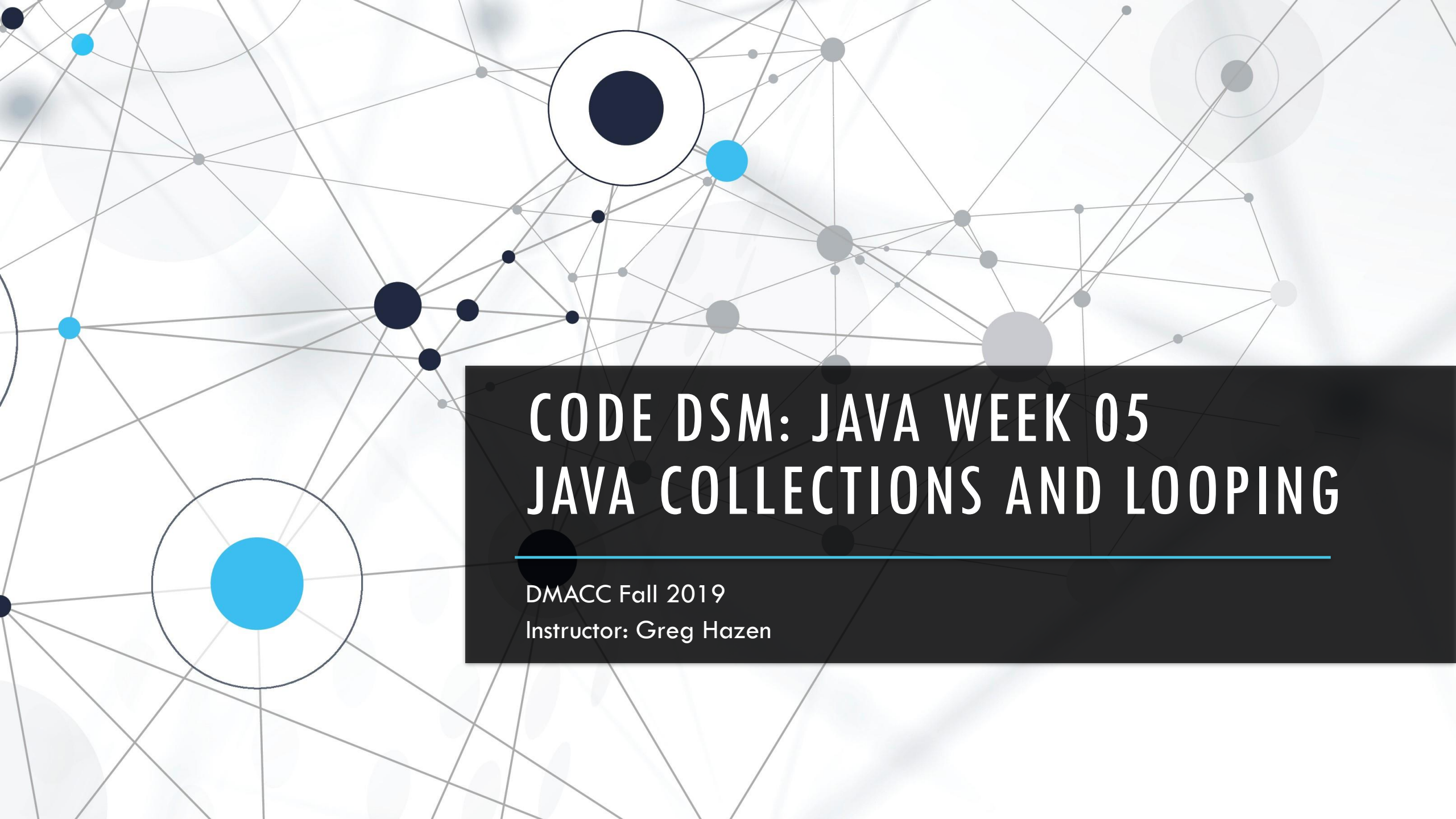


CODE DSM: JAVA WEEK 05
JAVA COLLECTIONS AND LOOPING

DMACC Fall 2019

Instructor: Greg Hazen

AGENDA REVIEW

REVIEW WEEK 4 HOMEWORK

AGENDA WEEK 5

JAVA ARRAYS

PROBLEM WITH ARRAYS

JAVA COLLECTIONS

REVIEW WEEK 4 ASSIGNMENT

Switch to IntelliJ!

WHY USE COLLECTIONS?

Hypothetical Situation

Store 10 numbers. After storing them, I'll ask questions like, "what's the greatest number?"

Solution

Store in 10 variables

WHY USE COLLECTIONS?

Hypothetical Situation

Store 100000 numbers. After storing them, I'll ask questions like, "what's the greatest number?"

Solution

???

ARRAYS

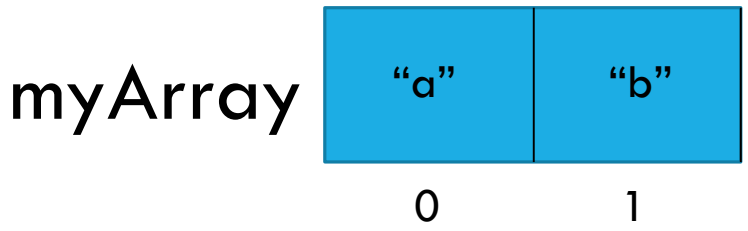
A series or list of variables in computer memory

All variables share the same name, and must be the same data type

Access using a zero-based index

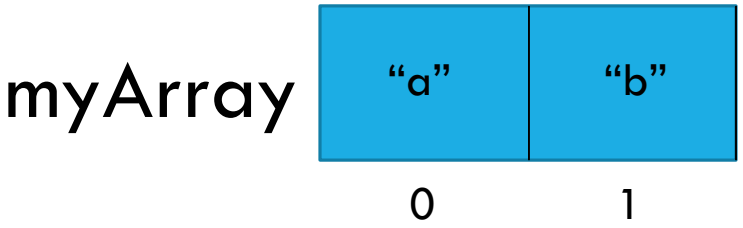
ARRAYS

```
String[] myArray = new String[2];
myArray[0] = “a”;
myArray[1] = “b”;
```



ARRAYS

String[] myArray = { “a”, “b” };



ARRAYS

Let's see more in IntelliJ!

Note you need the new keyword even with primitive arrays

Individual elements can be used just like any other variable

Array length

IndexOutOfBoundsException if your index is negative or greater than length-1

LOOPING ARRAYS

Let's see more in IntelliJ!

Looping by index

- Remember it's zero-based and the last index is length-1

Foreach, also known as enhanced loop

EXERCISE: ARRAYS

In a main method

- Create an array that stores five foods (e.g. “Burger”, “Fries”…)
- Print each food on a new line

EXERCISE: ARRAYS

Using your main method from earlier

- Keep your array that stores five foods
- Create another array that stores five corresponding prices (e.g. 2.50, 1.75…)
- Add a method that accepts (1) the foods array, (2) prices array and (3) a String with the name of a food. The method will print the price of the food.

Example, given

Foods: “burger”, “fries”, “pop”, “shake”, “salad”

Prices: 2.00, 0.75, 1.00, 2.50, 3.00

Choice: “pop”

Output: $1.00

MULTI-DIMENSIONAL ARRAYS

boolean[][] multiDimensional = new boolean[3][3];

Same rules apply as single-dimensional arrays

Not a lot of good use-cases in Java because there's better ways to organize code

LIMITATIONS TO ARRAYS

No way to add or remove from an array without creating a new one

Could create a helper method

- Prone to error
- Already exists

COLLECTIONS

In Java 6 the Collections API was introduced

Contains helper methods for working with collections

- add() and remove()
- size() and isEmpty()

Biggest benefit is dynamically adding/removing data!

COLLECTIONS

Collection Interface is abstract

Many different implementations available

- List: allows duplicates
- Set: doesn't allow duplicates
- And others…

WHEN SHOULD I USE ARRAYS?

When there's a fixed number of elements

- Months of the Year

If concerned about performance

- Dynamically adding can be expensive
- Whiteboard example of adding value to sorted list

WHEN SHOULD I NOT USE ARRAYS?

When there's a dynamic number of elements

When you have a LOT of data

- Finding contiguous memory to fit a really big array might be impossible causing OutOfMemoryError
- Whiteboard examples of contiguous memory, linked list memory and hybrid approach ArrayList

ONLY OBJECTS

Arrays allowed primitives or Objects

Collections only allow Objects

- Given a primitive it'll auto-box it to an Object
  - boolean to Boolean
  - byte to Byte
  - char to Character
  - float to Float
  - int to Integer
  - long to Long
  - short to Short
  - double to Double

GENERICS

Specify the Object type the Collection contains

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

Java 7+

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

GENERICS

Allows Java and developers to know what Objects to expect

```
Collection<String> names = new ArrayList<>();
names.add("Greg"); // Only accepts a String
```

LIST

Very similar to arrays, but no predefined size

Zero-based indices again

Several implementations

- ArrayList (most commonly used)
- LinkedList
- Stack

LIST

Unlike arrays we need the *new* keyword

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

Let's explore the helper methods in IntelliJ!

LOOPING AND ITERATING LISTS

Just like array we can use

- for loop by index
- for each loop (enhanced loop)

Collection also has iterator

What happens when you remove while looping?

Let's see in IntelliJ!

EXERCISE: REFACTOR ARRAY TO ARRAYLIST

Using your food ordering app from earlier, refactor it to using ArrayList instead.

Example, given

Foods: “burger”, “fries”, “pop”, “shake”, “salad”

Prices: 2.00, 0.75, 1.00, 2.50, 3.00

Choice: “pop”

Output: $1.00

SET

Doesn't allow duplicate values

Can loop and iterate over elements just like List

Can't access elements by index

Two implementations; only different in how they store and search their elements

- HashSet
- TreeSet

SET

```
Set<String> uniqueNames = new HashSet<>();
```

Let's explore the helper methods in IntelliJ!

MAP

Keep track of elements using a key

- "burger" is key to 2.00

Doesn't implement Collection Interface

Keys are stored in a Set, so they're unique

Two implementations; only different in how they store and search their elements

- HashMap
- TreeMap

MAP

Takes two generic types

- Key and Value can be different types

```
Map<String, Double> menu = new HashMap<>();
```

Let's explore the helper methods in IntelliJ!

EXERCISE: REFACTOR ARRAYLIST TO HASHMAP

Using your food ordering app from earlier, refactor it to using HashMap instead.

Example, given

Foods: “burger”, “fries”, “pop”, “shake”, “salad”

Prices: 2.00, 0.75, 1.00, 2.50, 3.00

Choice: “pop”

Output: $1.00

LOOP PERFORMANCE

```
for(String name : names) {
    for(String name : names) {
        // do something
    }
}
```

Big-O Notation of n^2

LOOP PERFORMANCE

```
for(String name : names) {
   // Store a variable with result from first loop
}
for(String name : names) {
   // do something using the result from the first loop
}
```

Big-O Notation of 2n

NO QUIZ THIS WEEK

Reminder that attendance is graded

NEW ASSIGNMENT SUBMISSION PROCESS

Now we'll be cloning the assignment and creating a private repository

Let's try it together with the week 5 assignment!

ASSIGNMENT

Review assignment for any clarifications

SEE YOU NOVEMBER 8TH!

Don’t forget to ask question early in the week