Lecture 3
Treating Arrays as Containers

If your bookshelf or desk organizers had automatic features…

Recap

Last meeting's Seatwork

Exercise: Filters - Integer

- Create the following methods to filter the elements of the numArr array:

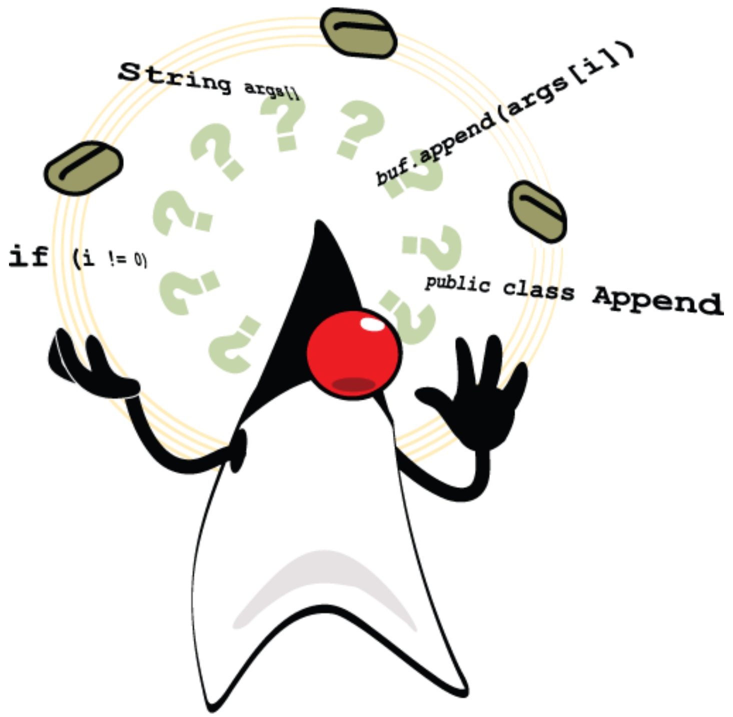
1. showEven – a method that accepts an array numArr. This method must display only the even number elements of numArr and its count (i.e., the number of even elements).
2. showHigher - a method that accepts an array numArr and an int variable val. This method must display the elements of numArr greater than val and its count.
3. showLower - a method that accepts an array numArr and an int variable val. This method must display the elements of numArr lesser than val and its count.
4. showRange - a method that accepts an array numArr. and2 int variables low & high. This method must display the elements of numArr between low & high and its count.
5. showGreatest - a method that accepts a String array wordArr. This method must display the greatest* value element of wordArr.

*The one that comes up **last** alphabetically.

Recap

So, did you do the reading assignment?

Reading Assignment: The Java API



Makes life easier if you know where to look.

But only makes it harder if you don't.

Problem

- Ask the user for 5 ints.
- Display the sum of the numbers.
- **Display the numbers from highest to lowest.**

This time, let’s try putting the whole list in order first.

Summary

Programming Your Own

- + **Flexible** – easy to modify in case of changing requirements
- + **Transparent** – you can see and check how the code works (to update or debug)
- + **Customized** – methods cater specifically for the requirements of the organization
- ✕ **Heavily algorithmic** – need familiarity of any required algorithms
- ✕ **Longer code** – adds more code to your programming projects

Using Prebuilt API Methods

- + **Algorithmically simple** – you don't need to know algorithms
- + **(Usually) Efficient** – algorithms used are usually better than basic ones
- + **Shorter code** – only see method calls and high-level algorithms
- ✕ **API knowledge** – need familiarity of one or more API classes and functionalities
- ✕ **Rigid implementation** – if it's not supported, it can be tricky to get the functionality needed

- Can you find a solution that balances **correctness** and **ease of implementation**?

```
import java.util.Arrays;

public static void printReverse(int[] given){
        for (int i = given.length - 1; i >= 0; i--)
                System.out.println(given[i]);
}

public static void printIntArrayDsc(int[] given){
        Arrays.sort(given);
        printReverse(given);
}
```

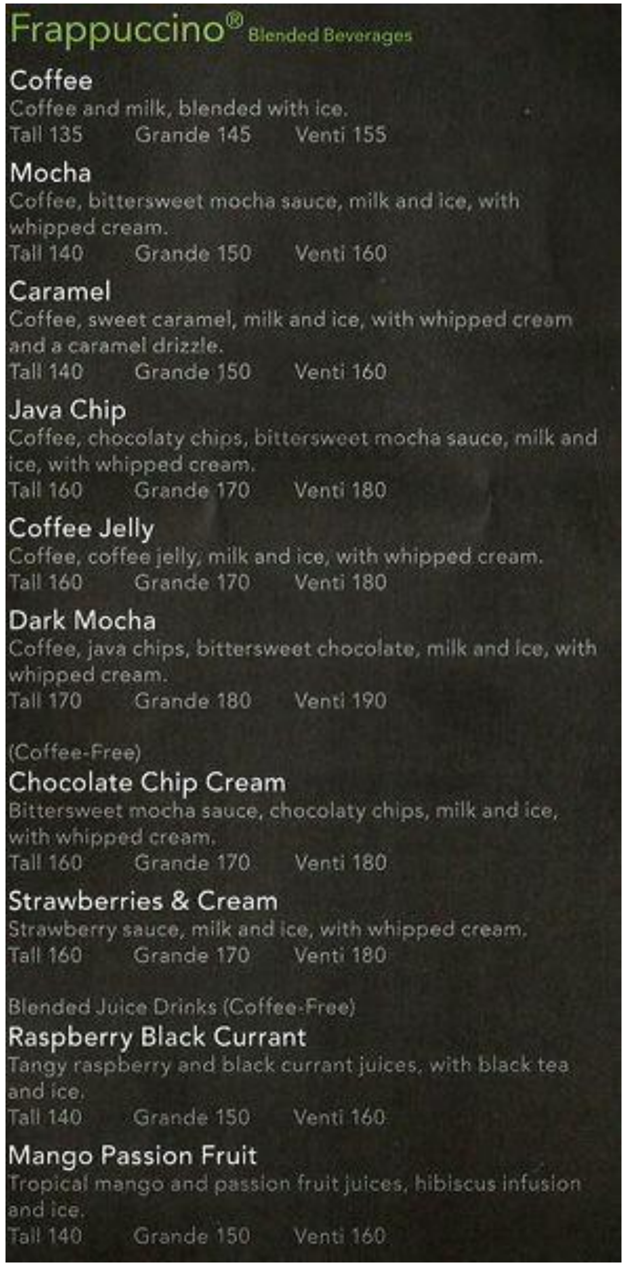
Use the API method that helps solve your problem.

Adjust your algorithms to compensate for what the API is missing.

Put some **meaning** into your data

- Suppose you were creating an electronic menu/price list for a restaurant.
- What data would be part of the menu? How would you store these in your program?
- What basic features would be part of your e-menu? How would you implement these features?

Quick Sample: Starbucks' Frappuccino Menu



Frappuccino® Blended Beverages

Coffee
Coffee and milk, blended with ice.
Tall 135 Grande 145 Venti 155

Mocha
Coffee, bittersweet mocha sauce, milk and ice, with whipped cream.
Tall 140 Grande 150 Venti 160

Caramel
Coffee, sweet caramel, milk and ice, with whipped cream and a caramel drizzle.
Tall 140 Grande 150 Venti 160

Java Chip
Coffee, chocolaty chips, bittersweet mocha sauce, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Coffee Jelly
Coffee, coffee jelly, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Dark Mocha
Coffee, java chips, bittersweet chocolate, milk and ice, with whipped cream.
Tall 170 Grande 180 Venti 190

(Coffee-Free)

Chocolate Chip Cream
Bittersweet mocha sauce, chocolaty chips, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Strawberries & Cream
Strawberry sauce, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Blended Juice Drinks (Coffee-Free)

Raspberry Black Currant
Tangy raspberry and black currant juices, with black tea and ice.
Tall 140 Grande 150 Venti 160

Mango Passion Fruit
Tropical mango and passion fruit juices, hibiscus infusion and ice.
Tall 140 Grande 150 Venti 160

A simple Coffee Frap

Can you identify what data (and type of data) we would need to store for this item?

- Coffee
  - Coffee and milk blended with ice.
    - Tall - 135
    - Grande - 145
    - Venti – 155

Item Name

Description

Size(s)

Price(s)

Note that all of this is only for **one** item.

App Feature: **Print Item**

- To test the basic data requirement, try to implement a feature that would **print out the details** of one menu item. What given data (parameters) would it need to do its job?
  - Name (String)
  - Description (String)
  - Size (3 Strings)
  - Price (3 doubles)

```
public static void printItem(String name, String
description, String size1, String size2, String
size3, double price1, double price2, double price3)
```

There are a lot of issues with this, and it's not just the length of the method.

Do some more **analysis**

```
public static void printItem(String name,
String description, String size1, String
size2, String size3, double price1, double
price2, double price3)
```

- **Issues!**
  - That's a **lot** of parameters!
  - What if an item has less or more than 3 sizes?

Towards a better **representation**

- Each item has
  - Name (String)
  - Description (String)
  - Sizes (bunch of sizes -> String[])
  - Prices (bunch of prices -> double[])

```
public static void printItem(String name,
String description, String[] sizes, double[]
prices)
```

Let’s **compare**

```
public static void printItem(String name,
String description, String size1, String
size2, String size3, double price1, double
price2, double price3)
```

```
public static void printItem(String name,
String description, String[] sizes, double[]
prices)
```

- **Advantages**
  - Fewer parameters (from 8 to just 4)
  - Each item can have a different number of sizes/prices (affects the length of the array)

On the issue of Storage

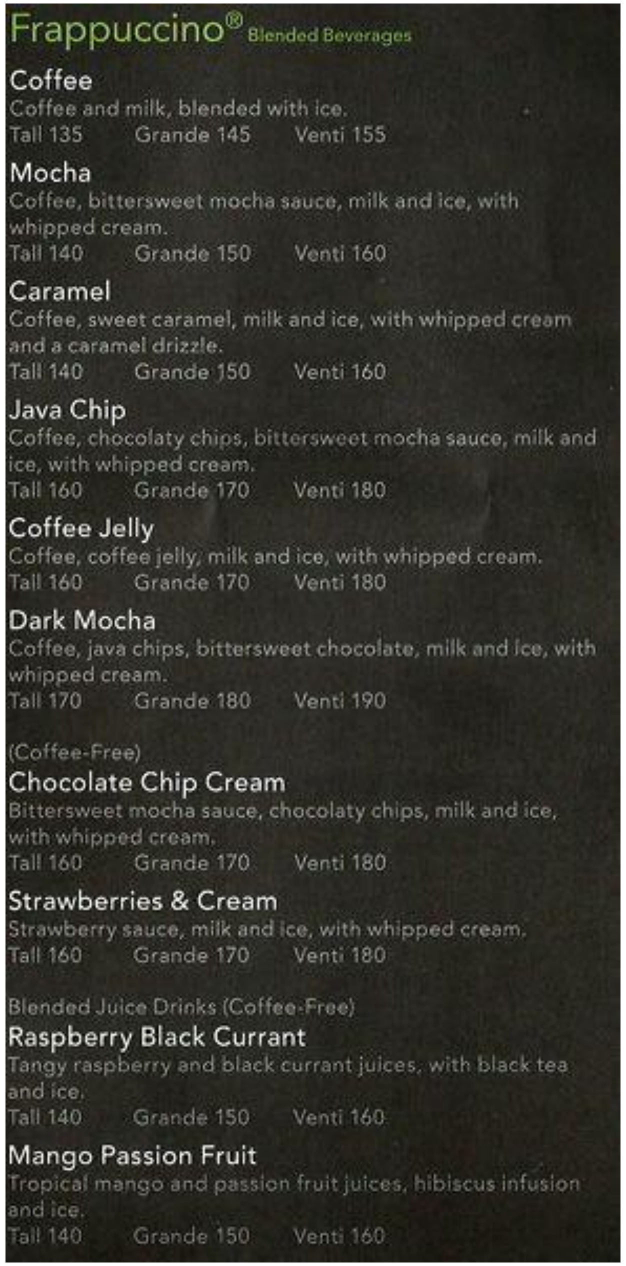
- What about the **entire menu**?
- You probably figured out that we'd need to store the menu items into an array somehow. But recall that each array can only store one type of data.
- A menu item is not *just* a String nor is it *just* an int… So, in order to properly represent our data, we would need to use multiple arrays (one for each data requirement).

In Array Form

To handle **many** items, we would need…

- Frappuccino Menu
  - String[] name;
  - String[] description;
  - String[][] size;
  - double[][] price;

An array can contain any type of element, even other arrays.


Frappuccino® Blended Beverages

Coffee
Coffee and milk, blended with ice.
Tall 135 Grande 145 Venti 155

Mocha
Coffee, bittersweet mocha sauce, milk and ice, with whipped cream.
Tall 140 Grande 150 Venti 160

Caramel
Coffee, sweet caramel, milk and ice, with whipped cream and a caramel drizzle.
Tall 140 Grande 150 Venti 160

Java Chip
Coffee, chocolaty chips, bittersweet mocha sauce, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Coffee Jelly
Coffee, coffee jelly, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Dark Mocha
Coffee, java chips, bittersweet chocolate, milk and ice, with whipped cream.
Tall 170 Grande 180 Venti 190

(Coffee-Free)
Chocolate Chip Cream
Bittersweet mocha sauce, chocolaty chips, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Strawberries & Cream
Strawberry sauce, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Blended Juice Drinks (Coffee-Free)
Raspberry Black Currant
Tangy raspberry and black currant juices, with black tea and ice.
Tall 140 Grande 150 Venti 160

Mango Passion Fruit
Tropical mango and passion fruit juices, hibiscus infusion and ice.
Tall 140 Grande 150 Venti 160

Understanding 2D Arrays

- A regular 1-D array of ints

Index	0	1	2	3	4
Value	43	15	7	101	-7

`arrVar[0]`

- An array of arrays of ints

Col Row	0	1	2	3	4
0	43	15	7	101	-7
1	24	43	4	4	10
2	43	8	54	9	23
3	21	-32	23	88	54

`arrVar[0]`

`arrVar[0][1]`

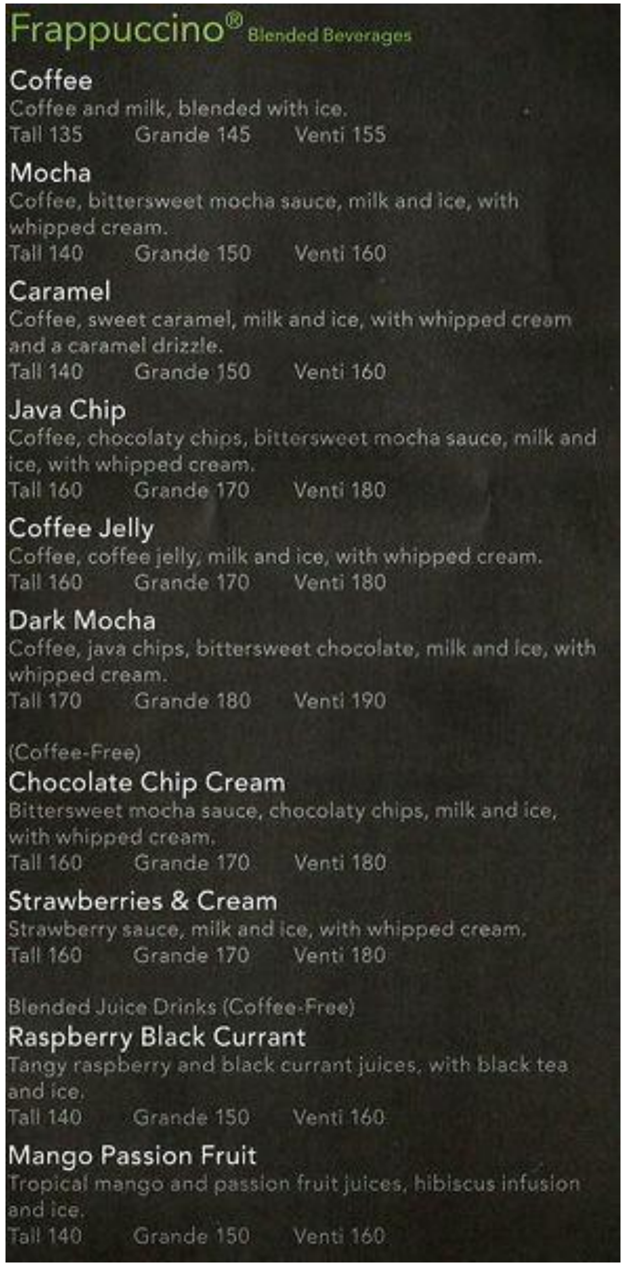
Visualizing the data structures

	Name
0	Coffee
1	Mocha
2	Caramel
…	…

	Description
0	Coffee and milk…
1	Coffee, bittersweet…
2	Coffee, sweet caramel…
…	…

Sizes	0	1	2	…
0	Tall	Grande	Venti	…
1	Tall	Grande	Venti	…
2	Tall	Grande	Venti	…
…	…	…	…	…

Prices	0	1	2	…
0	135	145	155	…
1	140	150	160	…
2	140	150	160	…
…	…	…	…	…



Frappuccino® Blended Beverages

Coffee
Coffee and milk, blended with ice.
Tall 135 Grande 145 Venti 155

Mocha
Coffee, bittersweet mocha sauce, milk and ice, with whipped cream.
Tall 140 Grande 150 Venti 160

Caramel
Coffee, sweet caramel, milk and ice, with whipped cream and a caramel drizzle.
Tall 140 Grande 150 Venti 160

Java Chip
Coffee, chocolaty chips, bittersweet mocha sauce, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Coffee Jelly
Coffee, coffee jelly, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Dark Mocha
Coffee, java chips, bittersweet chocolate, milk and ice, with whipped cream.
Tall 170 Grande 180 Venti 190

(Coffee-Free)
Chocolate Chip Cream
Bittersweet mocha sauce, chocolaty chips, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Strawberries & Cream
Strawberry sauce, milk and ice, with whipped cream.
Tall 160 Grande 170 Venti 180

Blended Juice Drinks (Coffee-Free)
Raspberry Black Currant
Tangy raspberry and black currant juices, with black tea and ice.
Tall 140 Grande 150 Venti 160

Mango Passion Fruit
Tropical mango and passion fruit juices, hibiscus infusion and ice.
Tall 140 Grande 150 Venti 160

Getting a menu item from the arrays

- The **index** becomes the key to accessing our data

```
public static void printItem(String name, String description, String[] sizes, double[] prices)


public static void main(String[] args){
    String[] name;
    String[] desc;
    String[][] size;
    double[][] price;
    …
    printItem(name[1], desc[1], size[1], price[1]);
    // prints out the details for Mocha Frappucino
}
```

Can you create the code to display everything in the menu?

On the issue of **Features**

- Now that you have a list that the computer can process, what sort of features should we **automate**?

Basic **list features**

- Add new item
- Search for an item
- Display an item
- Modify an item
- Delete an item
- Display the list (in different ways)

Design and implement the **methods** that you would need to perform these operations.

Additional **application features**

- What other features would be part of the sample application? This time, focus on different types of transactions.

- How would arrays help you support these features?

Research Assignment: Java Files API

Because we don’t want to manually enter long lists of data.

ResearchMe.txt

- Find the answers to the following questions:

  - How do you **READ** a text file using Java?
  - How do you **WRITE** to a text file using Java?

  - How would you read a list of items from a text file and load them into an array?
  - How would you write the updates to the list back to the text file?