Recap: Array Declaration and Types

- Just like ordinary variable, must
  - declare array before we use it
  - give array a type
  - Since cansSold contains integers, make integer array:
    ```java
    int[] cansSold = new int[10];
    ```
- Looks like variable declaration, except:
  - empty brackets on the left tell Java that cansSold is an array...
  - the number in the brackets on the right tells Java that array should have room for 10 elements when it's created

Arrays With Non-Primitive Types

```java
cansSold[0] = 185;
cansSold[1] = 92;
cansSold[2] = 370;
cansSold[3] = 570;
cansSold[5] = 209;
cansSold[6] = 128;
cansSold[7] = 84;
cansSold[8] = 151;
cansSold[9] = 329;
cansSold[10] = 563;
```

Arrays of Objects

```java
location[0] = "Chan Centre";
location[1] = "Law School";
location[2] = "Main Library";
```

Arrays and Object Design

- Think about a cleaner way to do all this...
  ```java
  CashIn[] cashIn = new CashIn[10];
  ```
- Multiple array approach not very object-oriented
  - can create arrays of objects
  - can create objects of our own design...
  ```java
  MultiObject[] multiObject = new MultiObject[10];
  ```

Recap: Array Declaration and Types

- News: Assignment 2 out
  - due Fri Mar 26
  - start now, it's challenging!

Arrays With Non-Primitive Types

```java
// do useful stuff here
System.out.println("Element 4 is " + cansSold[4]);
```
Arrays and Object Design

Cokematic object design - contains
- number of cans remaining: integer
- location: String,
- number of cans sold: integer
- cash collected: double

CokeEmpire
- Populate array with Cokematic objects
  myMachines.addCokematic(0, 100, "Chan Centre", 185, 201.25);

Cokematic
- Cokematic object design - contains
  - number of cans remaining: integer
  - location: String,
  - number of cans sold: integer
  - cash collected: double

CokeEmpire
- Populate array with Cokematic objects
  myMachines.addCokematic(1, 150, "Law School", 92, 100.50);

CokeEmpire
- Populate array with Cokematic objects
  myMachines.addCokematic(2, 200, "Main Library", 370, 612.75);

CokeEmpire
- What does this return?
  myMachines.getCokematic(index).getCansSold();

CokeEmpire
- Contains array of Cokematic objects
  public class CokeEmpire
  {
    private Cokematic[] collection;     // what does this do?
    public CokeEmpire()
    { collection = new Cokematic[10]; // what does this do?
      collection[0] = new Cokematic(100, "Chan Centre", 185, 201.25);
    }
    public void buyCoke(Cokematic c)
    { c.cansSold = c.numberOfCans; c.cashIn = c.cash; System.out.println("Have a Coke"); }
    public void reloadMachine()
    { System.out.println("Reloading machine"); }
    System.out.println(collection[0].toString());
    System.out.println("Adding machine");
  }

CokeEmpire
- In driver, executing
  CokeEmpire myMachines = new CokeEmpire();
  myMachines.addCokematic(0, 100, "Chan Centre", 185, 201.25);

CokeEmpire
- In driver, executing
  CokeEmpire myMachines = new CokeEmpire();
  myMachines.addCokematic(1, 150, "Law School", 92, 100.50);

CokeEmpire
- In driver, executing
  CokeEmpire myMachines = new CokeEmpire();
  myMachines.addCokematic(2, 200, "Main Library", 370, 612.75);
Arrays of Arrays

In any given array, all data must be of same type
All arrays in array of arrays must be of same type
So easier to use a two-dimensional array!

Two-Dimensional Arrays

In Java, 2D array implemented internally as array of arrays
but externally syntax of 2D array may seem easier to use

Arrays of Arrays

In any given array, all data must be of same type
All arrays in array of arrays must be of same type
So easier to use a two-dimensional array!

Two-Dimensional Arrays

In any given array, all data must be of same type
All arrays in array of arrays must be of same type
So easier to use a two-dimensional array!

Two-Dimensional Arrays

In Java, 2D array implemented internally as array of arrays
but externally syntax of 2D array may seem easier to use