CS 302: Introduction to Programming in Java

Lecture 11

Yinggang Huang

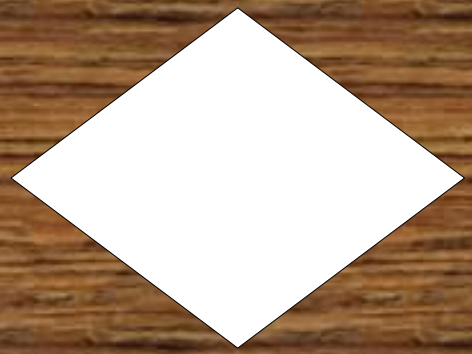
Review

- How do we call a method?
- What are method inputs called?
- How many values can be returned from a method?
- Write a method header called doSomething that returns a boolean and has 2 parameters: an int and a String

Why do we use Methods?

- Increase modularity
- Increase readability / maintainability
- Reduce redundancy
- Ex. P1:
  - Multiple Validation loops -> a single validation method
  - Multiple modes -> multiple methods (better style, easier debugging, etc.)

Calling Methods



ClassName.methodName(arguments)

ex. Math.pow(2, 4);

objectName.method(arguments)

ex. in.nextLine() or rand.nextInt(10)

- For now, we will only create static methods

- If calling static methods that are defined within the same class that they are being called from, the ClassName. identifier can be dropped from the method call (just call the method using its name)

Arguments vs Parameters

- Arguments (book calls them "parameter values") get passed to the method
- Parameters (book calls them "parameter variables") are defined in the method header
- Arguments must match the parameter definitions in type, order, and number
  - Do not need to have the same name
- Argument values get COPIED into the parameter variables
  - Changing the parameter does **NOT change the original argument**

Args vs Parameters Example

```
int a = 4, b = 5;

int area = rectArea(a, b);

...

public static int rectArea(int width, int height)
{
...
}
```

Blue = arguments
Red = parameters

Arguments must match parameters in **number**, **order**, and **type**

- a's value is copied to width
- b's value is copied to height

Variable Scope

- Just like what we talked about in ifs and loops
  - A variable declared within braces is ONLY valid within those braces
  - That means you can't use variables defined in a method outside of that method!!!
- Can use the same variable name in different scopes

Ex.

```
public static double rectArea(int length, int width)

public static double cubeVolumn(int lenght, int width, int depth)
```

Return Statement

- **Immediatly exits the method**
- Can return
  - Literal – return 4;
  - Variable – return x;
  - Result of an expression – return (x && y || (3+z < 5));
  - Result of another method call – return doSomething(x);
- Return type must match the type in the method header
- If the method returns nothing, it is of type void

Void methods

- Ex. public static void main(String[] args)
- Used when the method doesn't return anything
- Often used for displaying things
- Can still use the return statement to exit the method immediatly
  - In this case the statement is simply: return;

```
printStars(3, 4);
...
public static void printStars(int
    width, int height)
{
   for (int i = 0; i < width; i++)
      for (int j = 0; j < width; j ++)
         print("*");
      print("\n");
}
```

Return vs. Break

```
for (int i = 0; i < 10; i++)
{
   if (i == 5) break;
}
```

Vs

```
for (int i = 0; i < 10; i++)
{
   if (i == 5) return;
}
```

- Break simply breaks out of the current loop
  - What would happen in a nested loop?
- Return immediatly exits the method and returns the return value (if any)
  - What would happen in a nested loop?

Return within Conditionals

```
public static String getDay(int day)
{
   if (day == 1) return "Sunday";
   if (day == 2) return "Monday";
   if (day == 3) return "Tuesday";
   ...
}
```

- Note we don't need else ifs because the return statement exits the method immediatly!
- If we do branch every possible traversal must have a return statement!

Practice 1 (take home)

- Remember we had a practice (refer to HopeAndChange.java on the course website under "In-Class Example Code" tab)
- Let's do the same thing but now we use a static method to calculate change given coin value (25 for quarter, 10 for dime, 5 for nickel, 1 for penny), coin name ("quarter", "dime", "nickel", "penny"), change (centsLeft defined previously). So the parameter values for this method should be coinValue, coinName, centsLeft with appropriate data types, respectively.
- The method should be able to print out number of coins (quarters, dimes, nickels or pence) AND return the remainder (centsLeft).
- In the main method, call the method for the number of quarters, dimes, nickels, pennies.

Copying Arrays

int[] testArray1 = {1, 2, 3};

- What value does testArray1 hold? (What type of variables are arrays?)
- What happens if I do:
  - int[] testArray 2 = testArray1;
- If we want to actually copy an array:
  - Arrays.copyOf(arrayToCopy, n);
    - n = how many elements to copy over – if n is < arrayToCopy.length will only copy first n; if n is > arrayToCopy.length will copy over the entire array and give you (n – arrayToCopy.length) extra indices

Using Arrays with Methods

- What happens when arguments are passed to a method?
- What does the array variable hold?

- When arrays are passed to methods the address of the array is the value copied over
- When array elements are passed over, things work as usual
- Be VERY careful with this

Array – Method examples

```
int[] myArray = {1, 2, 3};
printArray(myArray);


public static void printArray(int[] array)
{
   for (int i = 0; i < array.length; i++)
   {
      System.out.println(array[i]);
   }
}
```

Array – Method examples

```
int[] myArray = {1, 2, 3};
multiplyArray(myArray);


public static void multiplyArray(int[] array)
{
   for (int i = 0; i < array.length; i++)
   {
      array[i] = array[i] * 2;
   }
}
```

Array – Method examples

```
int[] myArray = {1, 2, 3};
int x = square(myArray[0]);

public static int square(int a)
{
  return a*a;
}
```

Practice 2 (find a substring)

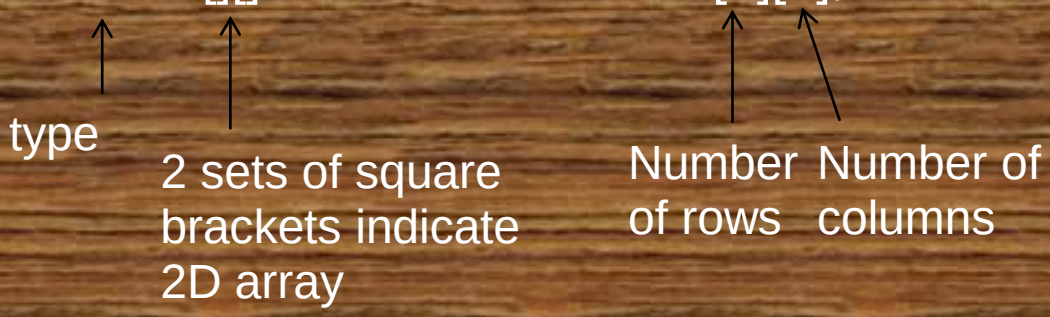
- Write a static method ***indexOf(String in, String sought, int fromIndex)***
- Returns the index within this string (in) of the first occurrence of the specified substring (sought), starting at the specified index (fromIndex). It returns -1 if not found. For example:
- If in = “ababcababc”, sought = “abc”, fromIndex = 0, then the method returns 2
- If in = “ababcababc”, sought = “abc”, fromIndex = 3, then the method returns 7
- If in = “ababcababc”, sought = “acb”, fromIndex = 0, then the method returns -1 (no match in String in)
- Click on http://docs.oracle.com/javase/6/docs/api/java/lang/String.html#indexOf(java.lang.String, int) for more info

2-D Arrays (Matrix)

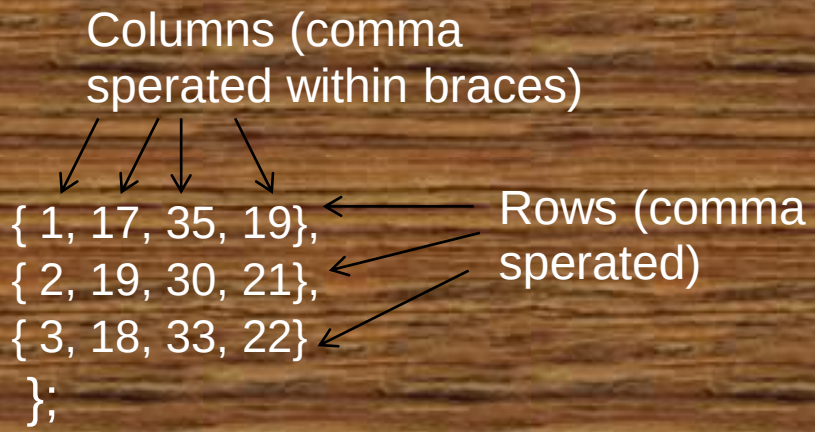
- It is often useful to have more than one row of data
- Solution: multi-dimensional arrays
- 2-D Array is basically a table
  - Has rows and columns
  - Remember – No changing length of either rows or columns once they have been initialized!

Declaration and Initialization

- 2 ways
  - If we don't know the data yet:
    - double[][] data = new double[3][4];



  - If we know the data:
    - double[][] data = {



Accessing Elements

- double[][] data = { { 1, 17, 35, 19},
- { 2, 19, 30, 21},
- { 3, 18, 33, 22} };
- Each element now has a row and column value thus specify each to get the element you want
- double val = data[0][0]; //val = 1

- val = data[2][3]; // val = 22
- REMEBER BOTH ROWS AND COLUMNS ARE 0-INDEXED!!!

Row and Column Length

- Row length = same as with 1D array
  - int rowLength = data.length;
- Column length = must specify column to get the length of whatever column you specify
  - int column0Length = data[0].length;
- For this class all columns will have the same length

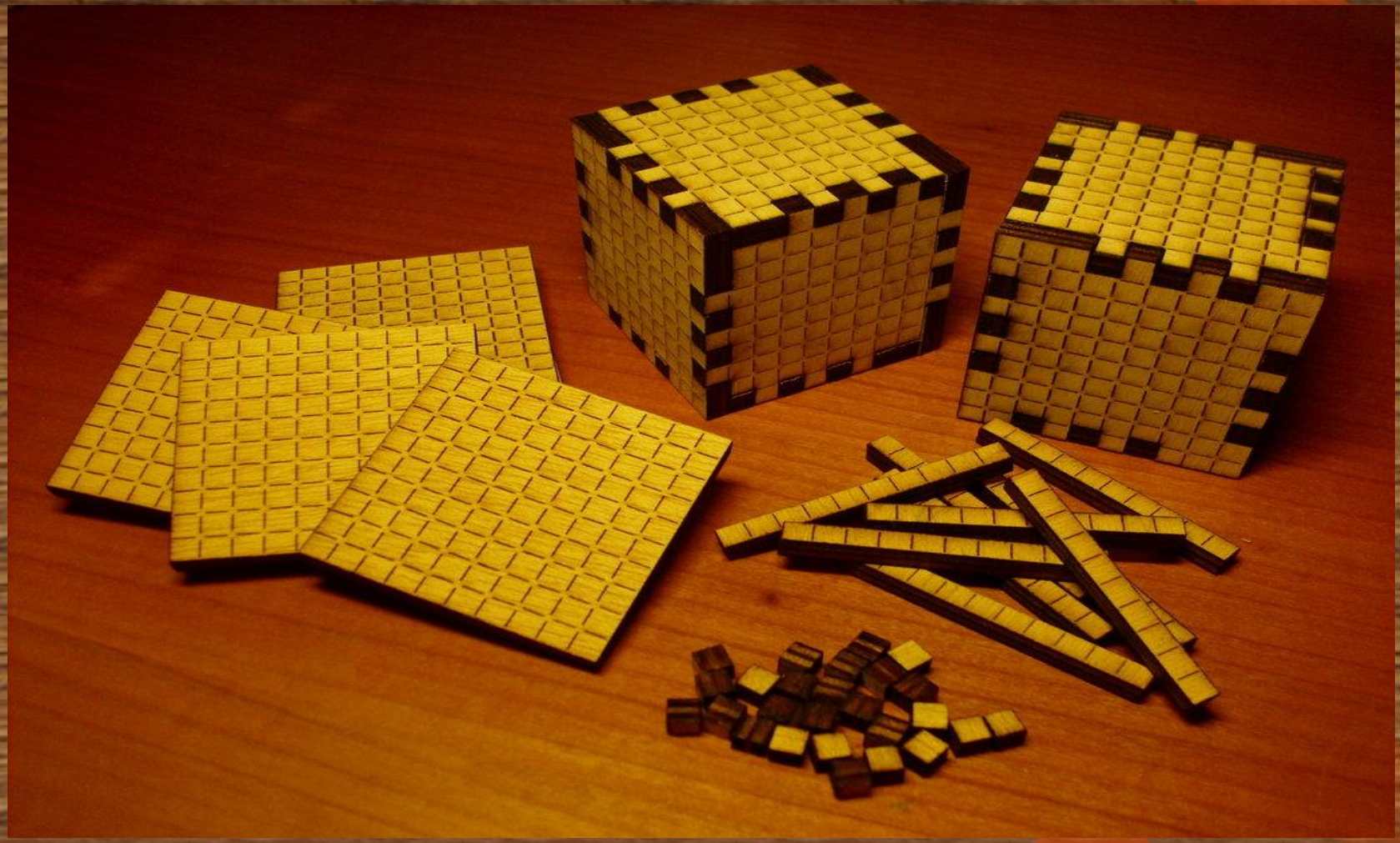
Working with 2D Arrays

- Often use nested for loops
- Ex. print values:

```
for (int row = 0; row < data.length; row++)
{
   for (int col = 0; col < data[0].length; col++)
   {
      System.out.print(data[row][col] + "\t");
   }
   System.out.println();
}
```

Recap

- Arrays are constructs that store multiple values of the same type
- They are used to simplify code and to simplify the manipulation of lots of data



Practice 3 (Maybe take home)

- Write a static method to print out all elements in a 2D int array
- public static void print2DArray(int[][] array)
- Test if the method works fine by calling it in main method, remember to declare and initialize a 2D array to be passed into the method