Data Structures

Day 3: Digging in to Java

David Cooper

Entry Survey (partial responses)

- 11 people filled out the survey
- I'll re-open the survey. What's the best time to close it?
- Let's look so far…

+

Questions about me

- How did you begin coding?
- How and when did you become interested in Computer science?
- What kind of work have you done in CS?
- What are your personal research interests?
- Which part of your research may I join later for independent study?
- Who was your greatest mentor and why?
- What weird side hobbies do you have, computer-related or not?
- What is your teaching style?

+
Questions about the class

- Do higher level computer science courses mainly focus on programming too?
- When are your office hours?
- What is the most efficient way to reach you and ask questions outside class?
- What are your expectations of us in the course?

Your Experience

- Past programming
  - Processing (9/11)
  - JavaScript (4/11)
  - Java (2/11)
  - Python (3/11)
  - other(1/11)
- Computers
  - web, email, word processing (32%)
  - use when I have to (7%)
  - spreadsheets and powerpoint (36%)
  - other math tools (20%)
- Class year
  - Second (50%)
  - Third or more (50%)
- Operating System
  - Mac (45%)
  - Windows (50%)
- Major
  - Computer Science
  - Linguistics (undeclared)
  - Math
  - None/Undecided
  - Physics
  - Political Science
  - Russian

+ What you hope to learn

- Do I enjoy CS enough to minor in it?
- the logic behind code
- Everything that the course has to offer.
- how to process data with code.
- Java/Master a programming language
- data structures
- more computing ability.
- enough to be comfortable and to enjoy computer science
- Java syntax
- more basics.

+

Concerns

- ability to grasp some of the abstract concepts
- Workload.
- I didn't take discrete math.
- I am worried about what I forgot.
- time management
- staying motivated
- transitioning from Python
- difficulty.
- Ability to comprehend the code.
- None.
- Learning the correct syntax
- Real world applications.
- enough resources beyond the lecture.

Prerequisites

- CMSC 110 or 105
- Motivation
- Ability and willingness to read.
- Willingness to try…

…then fail…

…then try again.

Expectations

- What can you expect from me?
  - challenging assignments
  - guidance
  - knowledge
  - willingness to help
  - willingness to take your feedback seriously

- What do I expect from you?
  - participation, questions, and answers
  - confusion.
  - preparedness
  - work completed and submitted on time
  - feedback

My Course Goals for you

- Preparedness for continued study in Computer Science
- Practical knowledge of programming Java.
- Understanding of standard Data Structures
- Understanding of how standard Data Structures fit into the Java API

What will be covered

- Java fundamentals
- Lists & Java Collections Framework
- ArrayLists, Linked Lists
- Algorithm Efficiency
- Stacks & Queues
- Recursion
- Trees
- Sets & Hash Tables
- Sorting Algorithms
- Graphs

+ Review

```
/**
 * Bare Bones Application
 */
public class JavaApplication {
    /**
     * this program prints the arguments
     * entered at the command line.
     * @param args - the arguments typed
     *              on the command line
     */
    public static void main(String[] args) {
        System.out.println("Arguments entered:");
        for(int i = 0; i < args.length; ++i) {
            System.out.println("\t" + args[i]);
        }
    }
}
```

+ Primitive Data Types

Display 1.2 **Primitive Types**

TYPE NAME	KIND OF VALUE	MEMORY USED	SIZE RANGE
`boolean`	`true` or `false`	1 byte	not applicable
`char`	single character (Unicode)	2 bytes	all Unicode characters
`byte`	integer	1 byte	−128 to 127
`short`	integer	2 bytes	−32768 to 32767
`int`	integer	4 bytes	−2147483648 to 2147483647
`long`	integer	8 bytes	−9223372036854775808 to 9223372036854775807
`float`	floating-point number	4 bytes	$-3.40282347 \times 10^{+38}$ to $-1.40239846 \times 10^{-45}$
`double`	floating-point number	8 bytes	$\pm 1.76769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$

Literals and Assignment

```
boolean result = true;
char capitalC = 'C';

byte b = 100;
short s = 10000;
int i = 100000;

double d1 = 123.4;
float f1 = 123.4f;
```

```
// The number 26, in decimal
int decVal = 26;

// The number 26, in hexadecimal
int hexVal = 0x1a;

// The number 26, in binary
int binVal = 0b11010;
```

Character and String Literals

- \b (backspace),
- \t (tab),
- \n (line feed),
- \f (form feed),
- \r (carriage return),
- \" (double quote),
- \' (single quote),
- \\ (backslash).
- null: used as a value for any reference type (not for primitive types)

Type Casting and Constants

- int x = 5;
- float y = 4.7;
- x = (int) y;
- y * x; // gets 23.5
- (int) y + x; // gets 9

- static final int MIN = 0;
- static final char END = 'e';

Storage model

- Simple types
  - basic data types
  - Always have a value
- Reference types
  - Are always Objects
  - can be null
  - must be instantiated
  - Wrappers exist for basic data types (Integer, Float, etc.)

Reading for today

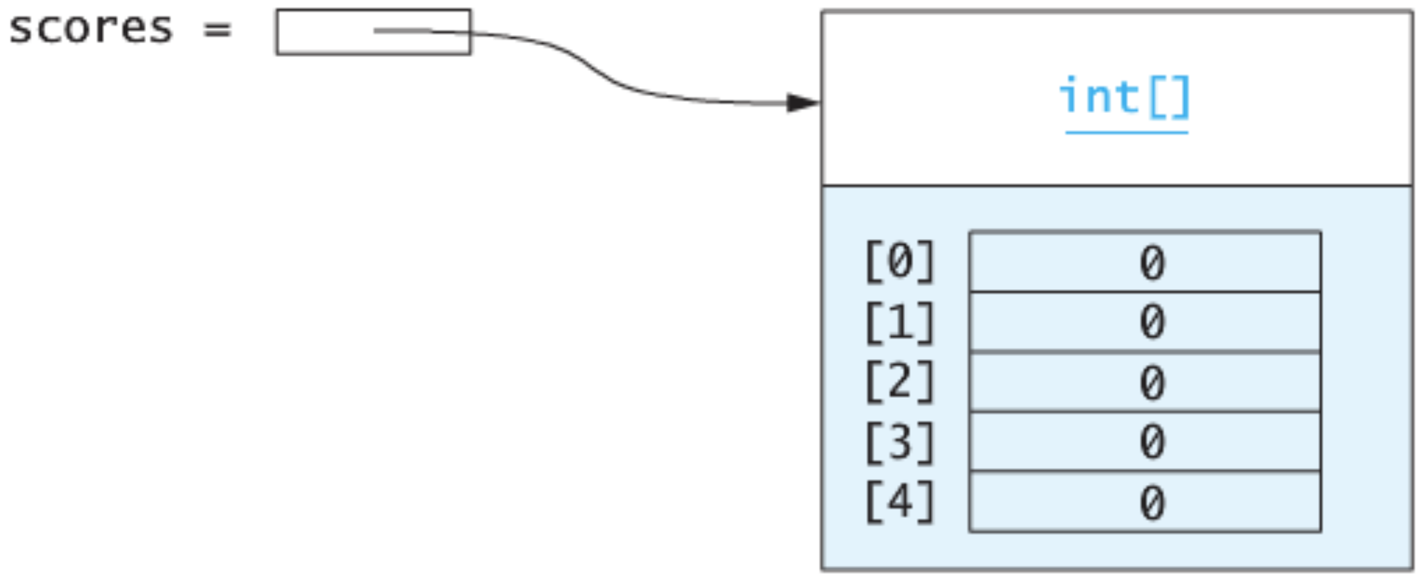
- A.8 Arrays (questions?)
- A.9 I/O using JOptionPane (questions?)
- A.10 I/O Using Streams and the Scanner Class
- A.11 Catching Exceptions

+
Arrays

- fixed size
- multiple things of the same type
- passed by reference by default
- Library methods for copying values
  - "Grow" an array using Arrays.copyOf:
    - int[] scores = {1,2,3,4};
    - int[] tempScores = Arrays.copyOf(scores, 2 * scores.length);
    - scores = tempScores;
  - Copy values using System.arrayCopy:
    - System.arraycopy(source, sourcePos, destination, destPos, numElements);

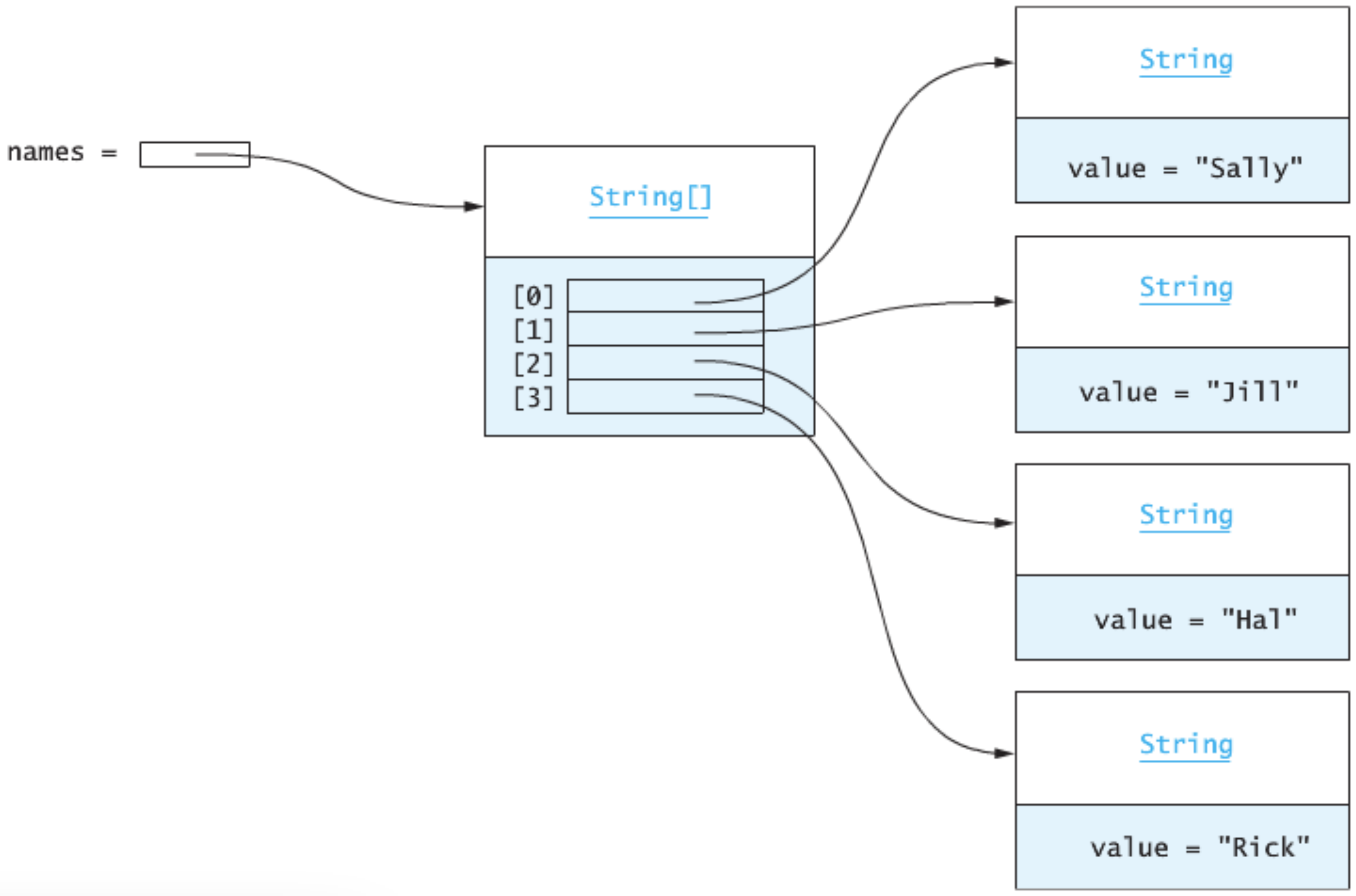
+ Arrays

```
int[] scores = new int[5];  // An array
```



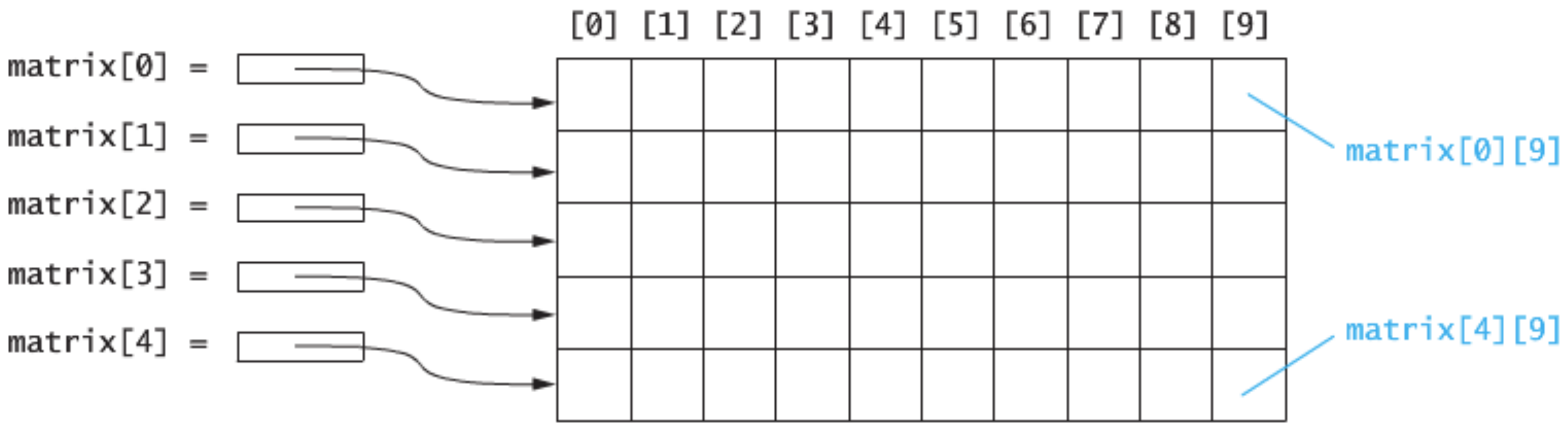
+ Array of Strings

```
String[] names = {"Sally", "Jill", "Hal", "Rick"};
```



2-D Array

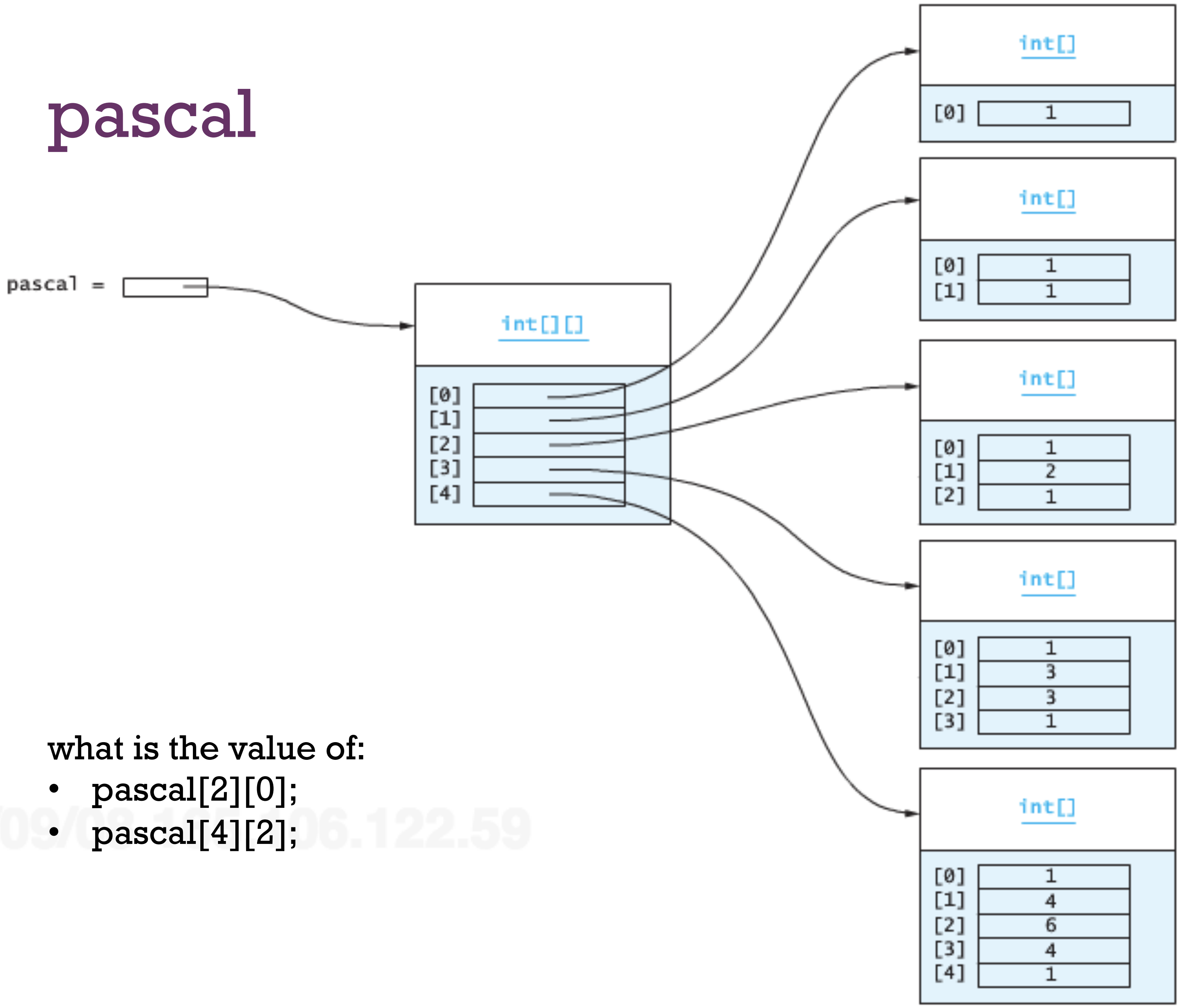
- double[][] matrix = new double[5][10];



Ragged Array

- int[][] pascal = new int[5][]; // make a ragged array with 5 rows
- pascal[0] = new int[1]; // make the first row have 1 column;
- pascal[1] = new int[2]; // make the second row have 2 cols;
- Or, in a loop
  - for (int i = 0; i < pascal.length; ++i) {
    - pascal[i] = new int[i+1];
  - }

pascal

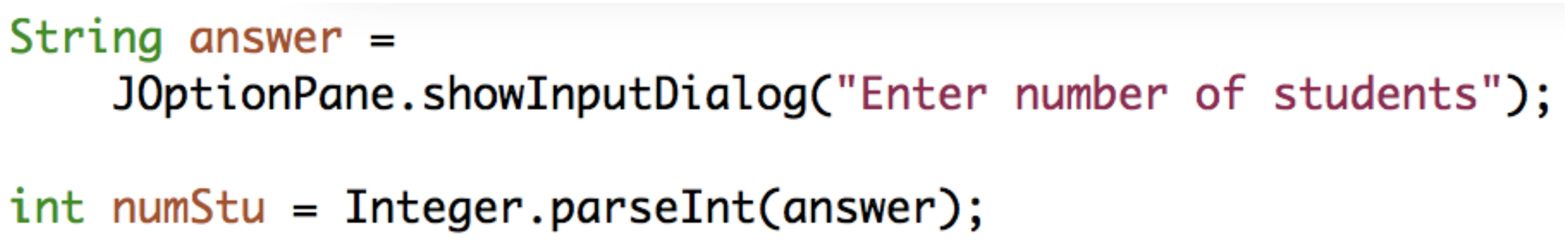


what is the value of:

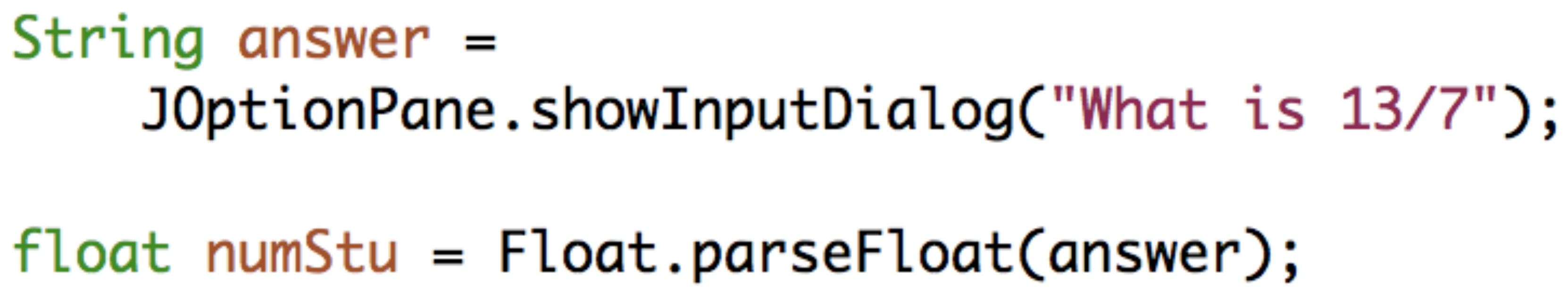
- pascal[2][0];
- pascal[4][2];

Arrays of Objects

I/0: JOptionPane

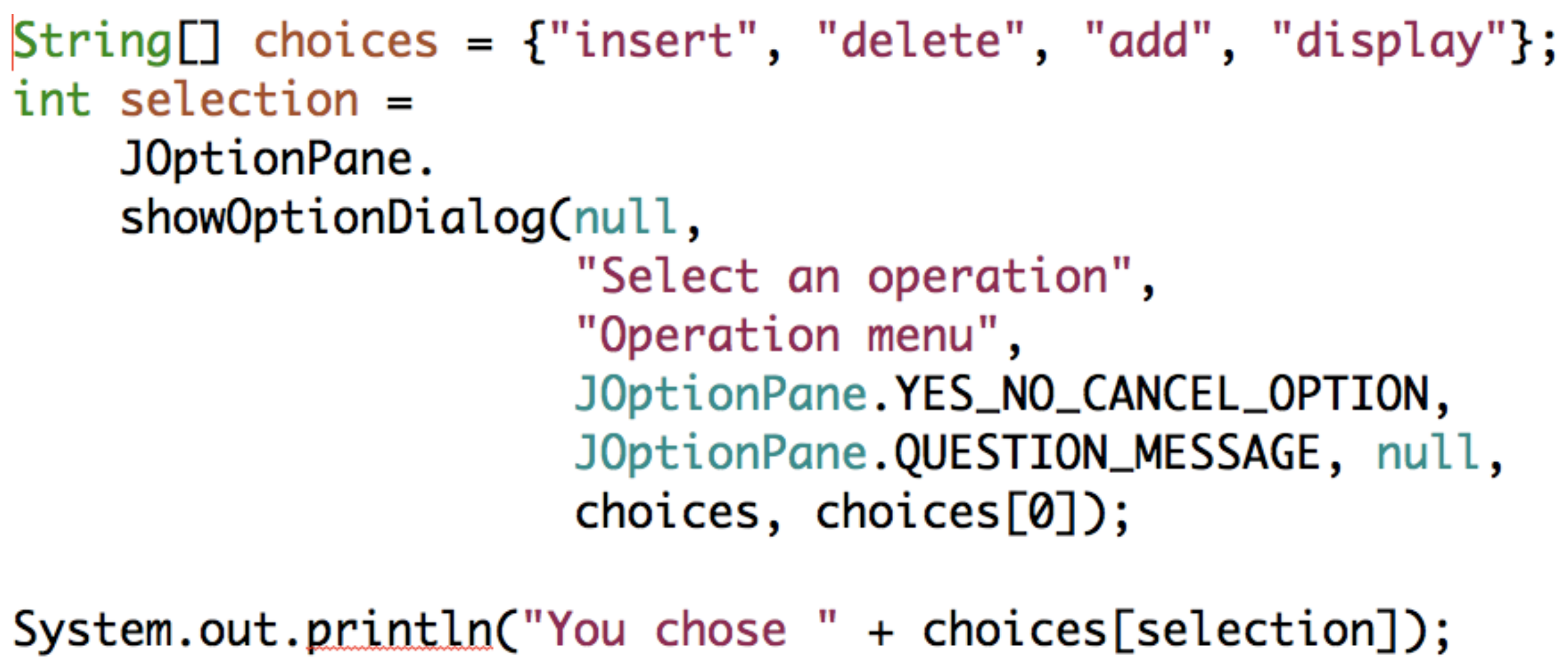


```
String answer =
    JOptionPane.showInputDialog("Enter number of students");

int numStu = Integer.parseInt(answer);
```



```
String answer =
    JOptionPane.showInputDialog("What is 13/7");

float numStu = Float.parseFloat(answer);
```

JOptionPane choices



```java
String[] choices = {"insert", "delete", "add", "display"};
int selection =
    JOptionPane.
    showOptionDialog(null,
                     "Select an operation",
                     "Operation menu",
                     JOptionPane.YES_NO_CANCEL_OPTION,
                     JOptionPane.QUESTION_MESSAGE, null,
                     choices, choices[0]);

System.out.println("You chose " + choices[selection]);
```

+ I/O Streams

- InputStreams:
  - System.in
- OutputStreams:
  - System.out
  - System.err

- Managable Input:
  - Scanner
  - StreamTokenizer(advanced)
- Managable Output
  - PrintWriter

- Objects related to streams:
  - String
  - File
  - Reader (input)
  - Writer (output)

+ Scanner Examples

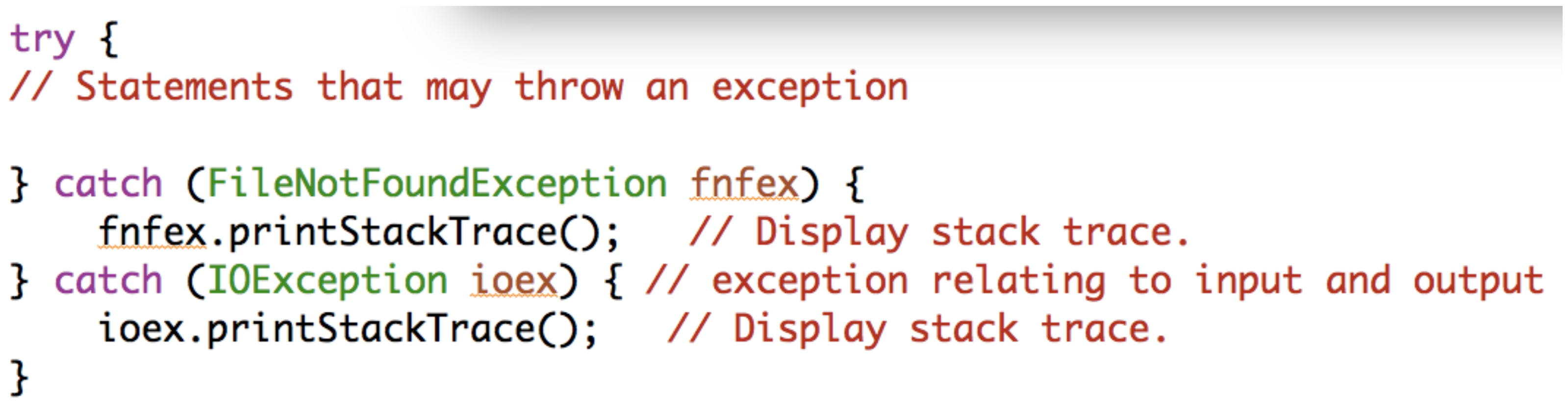
```
Scanner sysIn = new Scanner(System.in);
Scanner fileIn = new Scanner(new File("zips.txt"));
Scanner stringIn = new Scanner("here is some text.");

int     x =     sysIn.nextInt();
float   y =    fileIn.nextFloat();
String z = stringIn.next();
```

+ PrintWriter Example

```
PrintWriter fileOut =
    new PrintWriter(new FileWriter("testFile.txt"));

fileOut.println("Hello, File");
```

+

Catching Exceptions



```
try {
// Statements that may throw an exception

} catch (FileNotFoundException fnfex) {
    fnfex.printStackTrace();    // Display stack trace.
} catch (IOException ioex) { // exception relating to input and output
    ioex.printStackTrace();    // Display stack trace.
}
```