2-D Arrays

CSE 142, Summer 2003

Computer Programming 1

http://www.cs.washington.edu/education/courses/142/03su/

Review – Arrays

- Simple, ordered collections
- Elements of a particular array all have the same type
- Size fixed when array created

```
Rectangle[ ] rects = new Rectangle[count];
```

- Indexed access to elements

```
rects[3] = new Rectangle( );
rects[3].moveBy(10, 20);
```

2-D Arrays

- Suppose we want to represent a list of things, each of which consists of a list of other things
- For example
  - » several patterns, each of which includes several shapes
  - » several groups of computers
  - » several groups of books
- One way to do this is with a *2-dimensional* array
  - » The first dimension is the group
  - » The second dimension is the elements of the group

Using 2-D arrays

- Type pattern
  - » **<*elem type*>[ ][ ]**
  - » for example: **`Shape[][]`**
- New expression pattern
  - » **new <*elem type*>[<*dim 1 size*>][<*dim 2 size*>]**
  - » for example: **`Shape[][] pat = new Shape[10][5];`**
  - » for example: **`Shape[][] pat = new Shape[10][];`**
- Access expression / assignment pattern
  - » **<*array*>[<*dim 1 index*>][<*dim 2 index*>]**
  - » for example: **`Shape x = pat[1][0];`**
  - » "from pattern **`pat[1]`** select the first Shape **`pat[1][0]`"**

Picture

2-D Array = Array of Arrays

- A 2-D array is just an array of arrays
- There are various ways to access the elements
  - » access an element directly

```
Shape x = raggedPatterns[i][j];
```

  - » get the row array, then access elements of the row individually

```
Shape[] row = raggedPatterns[i];
Shape x = row[j];
```

Summary

- 2-D arrays
  - » In Java, just an array of arrays
  - » Syntax is extension of 1-D array case
    - *type*[ ][ ] *name* = new *type*[*nRows*][*nCols*]
    - *name*[*r*][*c*]
- n-D arrays
  - » you can have any number of dimensions
- arrays of objects are very handy, and usually more useful than a 2-D array