CSE 142

2-D Arrays

Introduction

- **Review:**
  - **Simple, one-dimensional arrays**
- **Today:**
  - **Two-dimensional arrays**
- **Reading**
  - **Dugan notes: ch. 20**

Review – Arrays

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

```
Rectangle[ ] rects = new Rectangle[42+17];
```

- **Indexed access to elements.**

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


2-D Arrays

- **Suppose we want to represent a picture.**
  (Disclaimer: simple-minded representation for lecture purposes.)
- **Want a rectangular, *2-dimensional* matrix of colored Rectangles, each of size 1x1.**
- **We can create an array with 2 dimensions to hold the picture.**
  - **Type pattern:** <*elem type*>[ ][ ]
  - **New expr pattern:** new <*elem type*>[<*dim 1 size*>][<*dim 2 size*>]
  - **Access expr/assignment pattern:** <*array*>[<*dim 1 index*>][<*dim 2 index*>]

```
Rectangle[ ][ ] picture = new Rectangle [40] [60];
picture[0][0] = new Rectangle(0,0,1,1, Color.blue, true);
```


2-D Array = Array of Arrays

- **A 2-D array is really just an array of arrays.**
  (In languages like FORTRAN and C/C++, this isn't true.)
- **It's possible to manipulate each row array separately.**
  - **(Draw the picture!)**

```
Rectangle[ ][ ] picture = new Rectangle[40][60];
picture[0][0] = new Rectangle(0,0,1,1, Color.blue, true);
...
Rectangle[ ] firstRow = picture[0];
firstRow[0] = new Rectangle(0,0,1,1, Color.red, true);
```

- **What do the following evaluate to?**

```
picture.length
firstRow.length
picture[0][0].length
```


Collections of Collections

- **Arrays of arrays are just a special case of allowing collections to hold any kind of object, including another collection.**
- **If more convenient, we could have used ArrayLists whose elements were ArrayLists to represent the picture.**

2-D Array Traversal

- **Typical traversal is to go through the rows and, for each row, go through the columns. Called "row-major order".**

```
public void initialize(Rectangle[ ][ ] picture, Color initialColor) {
    for (int row = 0; row < picture.length; row++) {
        for (int col = 0; col < picture[row].length; col++) {
            picture[row][col] = new Rectangle(col,row,1,1,initialColor,true);
        }
    }
}
```

- **Notice how the upper bounds of the two loops are computed.**

Exercise: Shift Picture to Left

```
// Copy colors one cell to the left, setting last column to white
public void shiftLeft(Rectangle[ ][ ] picture) {

    for (int row = 0; row <                    ; row++) {

        for (int col = 0; col <                    ; col++) {

            Rectangle thisPixel =

            Rectangle pixelToRight =

            thisPixel.setColor( pixelToRight.getColor( ) );
        }

    }
}
```


Exercise: Shift Picture Down

```
// Copy colors one cell downwards, setting first row to white
public void shiftDown(Rectangle[ ][ ] picture) {









}
```

- **Hint: row-major order might not be the right approach.**