CSE1030 – Introduction to Computer Science II

Lecture #16

Arrays II

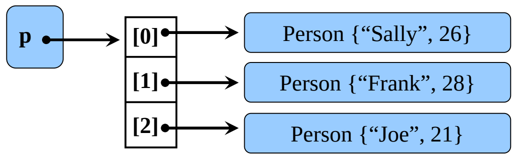
CSE1030 – Lecture #16

- Review: Arrays
- Regular 2D Arrays
- Irregular 2D Arrays
- We're Done!

Review: An Array is…

- A Name, and a **Table of Arrows (Pointers)**, to Blocks of Memory:

```
Person[] p = new Person[] {
   new Person("Sally", 26),
   new Person("Frank", 28),
   new Person("Joe", 21),
};
```



```
class example1
{
   public static void main(String[] args)
   {
      Person[] p = {
         new Person("Sally", 26),
         new Person("Frank", 28),
         new Person("Joe", 21),
      };

      System.out.println("Here's #2:");
      System.out.println( p[1] );

      System.out.println("Here's All of Them:");
      for(int i = 0; i < p.length; i++)
         System.out.println("  " + p[i]);

      System.out.println("Cause an Error! #4:");
      System.out.println( p[3] );
   }
}
```

Array Example Output

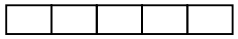
```
>java example1
Here's #2:
Frank(28)
Here's All of Them:
  Sally(26)
  Frank(28)
  Joe(21)
Cause an Error! #4:
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 3
        at example1.main(example1.java:20)
```

CSE1030 – Lecture #16

- Review: Arrays
- Regular 2D Arrays
- Irregular 2D Arrays
- We're Done!

The Big Idea so far…

- When data "looks like" this:



(and you can't use, or don't need the complexity of, a Collection)

- Use an array:

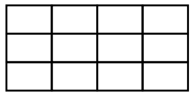
```
Object[] array = new Object[5];
```

array =

array[0]
array[1]
array[2]
array[3]
array[4]

New Idea… What about Tables?

- What do we do when the data "looks like" this?



- Use a 2-Dimensional array:

```
Object[3][4] array = new Object[3][4];
```

array =

array[0][0]	array[0][1]	array[0][2]	array[0][3]
array[1][0]	array[1][1]	array[1][2]	array[1][3]
array[2][0]	array[2][1]	array[2][2]	array[2][3]

2D Array Notation (1/4)

- Declare Arrays:

```
int[][] someNumbers;
String[][] words;
```

- Constructing Empty Arrays:

```
someNumbers = new int[10][5];
String[] words = new String[3][2];
```

2D Array Notation (2/4)

- Initialising with Hardcoded Values:

```
int[][] someNumbers = {
   { 2, 3, 5, 7, 11, },
   { 13, 17, 19, 23, 31, },
};

String[][] words = {
   { "Hello", "Good Bye" },
   { "Bonjour", "Au revoir" }
};
```

Array Notation (3/4)

- Using Arrays:

```
int n = someNumbers[i][j];

somenumbers[0][4] = 11;

String greeting = words[1][0];
```

- Array Size

# rows:	# columns:
`someNumbers.length`	`someNumbers[0].length`
`words.length`	`words[0].length`

2D Array Notation (4/4)

- Accessing a single Row:

```
int[][] someNumbers = {
   { 2, 3, 5, 7, 11, },
   { 13, 17, 19, 23, 31, },
};

int[] oneRow = someNumbers[1];

for(int i = 0; i < oneRow.length; i++)
   System.out.print(" " + oneRow[i]);
```

- Output:

```
 13 17 19 23 31
```

Example Program: ChessBoard.java

- The game Chess is played on a board that is 8 squares x 8 squares

- So a 2D 8x8 array is an appropriate way to store the board information…

ChessBoard.java Output:

```
+--------+--------+--------+--------+--------+--------+--------+--------+
|  Rook  | Knight | Bishop | Queen  |  King  | Bishop | Knight |  Rook  |
+--------+--------+--------+--------+--------+--------+--------+--------+
|  Pawn  |  Pawn  |  Pawn  |  Pawn  |  Pawn  |  Pawn  |  Pawn  |  Pawn  |
+--------+--------+--------+--------+--------+--------+--------+--------+
|        |        |        |        |        |        |        |        |
+--------+--------+--------+--------+--------+--------+--------+--------+
|        |        |        |        |        |        |        |        |
+--------+--------+--------+--------+--------+--------+--------+--------+
|        |        |        |        |        |        |        |        |
+--------+--------+--------+--------+--------+--------+--------+--------+
|        |        |        |        |        |        |        |        |
+--------+--------+--------+--------+--------+--------+--------+--------+
|  Pawn  |  Pawn  |  Pawn  |  Pawn  |  Pawn  |  Pawn  |  Pawn  |  Pawn  |
+--------+--------+--------+--------+--------+--------+--------+--------+
|  Rook  | Knight | Bishop | Queen  |  King  | Bishop | Knight |  Rook  |
+--------+--------+--------+--------+--------+--------+--------+--------+
```

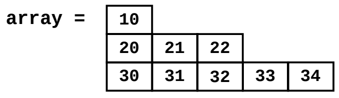
CSE1030 – Lecture #16

- Review: Arrays
- Regular 2D Arrays
- Irregular 2D Arrays
- We're Done!

Irregular 2D Arrays

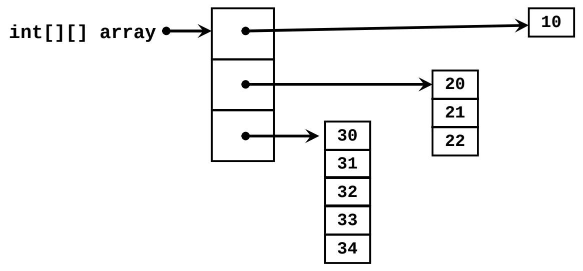
- Have a number of Rows
- But the number of columns differ in some of the rows

```
int[][] array = {
   { 10, },
   { 20, 21, 22, },
   { 30, 31, 32, 33, 34, },
};
```



How are Irregular 2D Arrays Possible?

- A 2D Array is really an "Array of Arrays":



example2.java

```
class example2
{
   public static void main(String[] args)
   {
      int[][] array = {
         { 10, },
         { 20, 21, 22, },
         { 30, 31, 32, 33, 34, },
      };

      System.out.println("Rows:");
      System.out.println("  array.length = " + array.length);
```

```
         System.out.println();
         System.out.println("Columns:");
         for(int i = 0; i < array.length; i++)
            System.out.println(
                  "  Column " + i
                + " has length: array[" + i + "].length = "
                + array[i].length);

         System.out.println();
         System.out.println("Entire array:");
         for(int i = 0; i < array.length; i++)
         {
            for(int j = 0; j < array[i].length; j++)
               System.out.print(" " + array[i][j]);
            System.out.println();
         }

         System.out.println();
         System.out.println("Just the 2nd row array:");
         int[] secondRow = array[1];
         for(int j = 0; j < secondRow.length; j++)
            System.out.print(" " + secondRow[j]);
         System.out.println();
      }
   }
```

example2 Output

```
>java example2
Rows:
  array.length = 3

Columns:
  Column 0 has length: array[0].length = 1
  Column 1 has length: array[1].length = 3
  Column 2 has length: array[2].length = 5

Entire array:
 10
 20 21 22
 30 31 32 33 34

Just the 2nd row array:
 20 21 22
```

Example Problem

- We want a database to allow us to catalogue the Moons of the planets in our Solar System

Earth	-	Moon
Mars	-	Phobos
	-	Deimos
Jupiter	-	Io
	-	Europa
	-	Ganymede
	-	etc.
Saturn	-	Titan
	-	Rhea
	-	etc.
Uranus	-	Cordelia
	-	Ophelia
	-	etc.
etc.		

Inefficient Array Implementation

- We could build an array like this:

```
String[][] = PlanetMoonDataBase = {
   { "Earth", "Moon" },
   { "Mars", "Phobos" },
   { "Mars", "Deimos" },
   { "Jupiter", "Io" },
   { "Jupiter", "Europa" },
   { "Jupiter", "Ganymede" },
   { "Jupiter", "Callisto" },
   { "Jupiter", "Amalthea" },
   { "Jupiter", "Himalia" },
   { "Jupiter", "Elara" },
   { "Jupiter", "Pasiphae" },
   { "Jupiter", "Sinope" },

   etc.
};
```

Redundancy Wastes Space!

A More Efficient Solution…

```
static String[][] Moons = {
   // Mercury = 0;
   {},

   // Venus   = 1;
   {},

   // Earth   = 2;
   {"Moon"},

   // Mars    = 3;
   {"Phobos", "Deimos",},

   // Jupiter = 4;
   {
      "Io", "Europa", "Ganymede", "Callisto", "Amalthea",
      "Himalia", "Elara", "Pasiphae", "Sinope", "Lysithea",
      "Carme", "Ananke", "Leda", "Metis", "Adrastea",
      "Thebe", "Callirrhoe", "Themisto", "Kalyke", "Iocaste",
   etc...
```

Moons.java

- Example Program

Advanced Usage of Arrays…

- You can have higher-dimensional arrays:

```
int[][][] array = {
       ...
};
```

- You can have arrays of Objects:

```
Object[] array = {
   new Moons(),
   new ChessBoard(),
   new Integer(10),
   new String[] { "Hi", "Bye" },
};
```

CSE1030 – Lecture #16

- Review: Arrays
- Regular 2D Arrays
- Irregular 2D Arrays
- We're Done!

Next topic…

Linked Lists