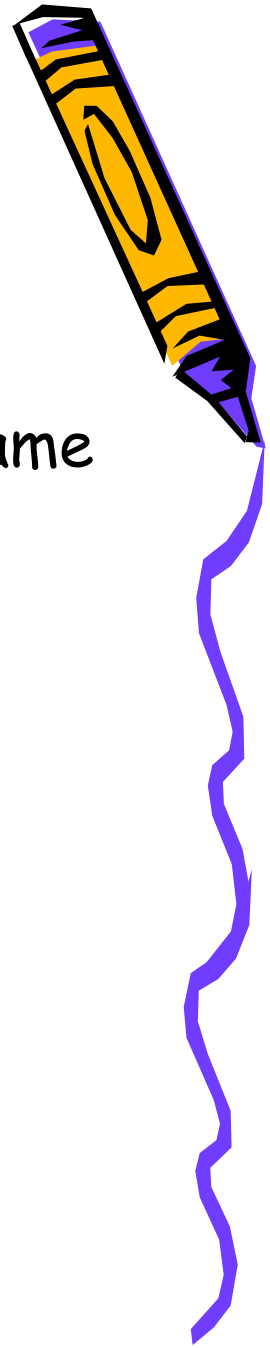
1D Arrays (syntax)



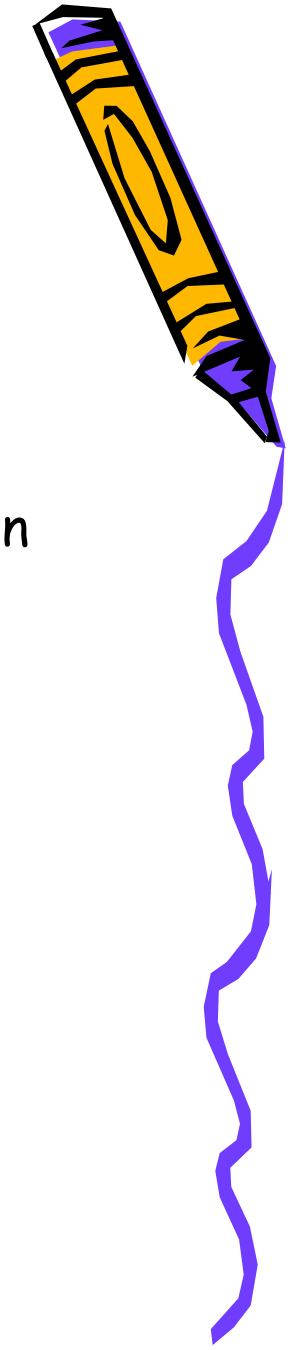
- What if we need to store many values of the same type?
  - Declare and use an array data structure
- 1D Array Syntax
  - Define variable

```
dataType[][] arrayVarName;     //preferred
dataType arrayVarName[][];
```

  - Allocate memory

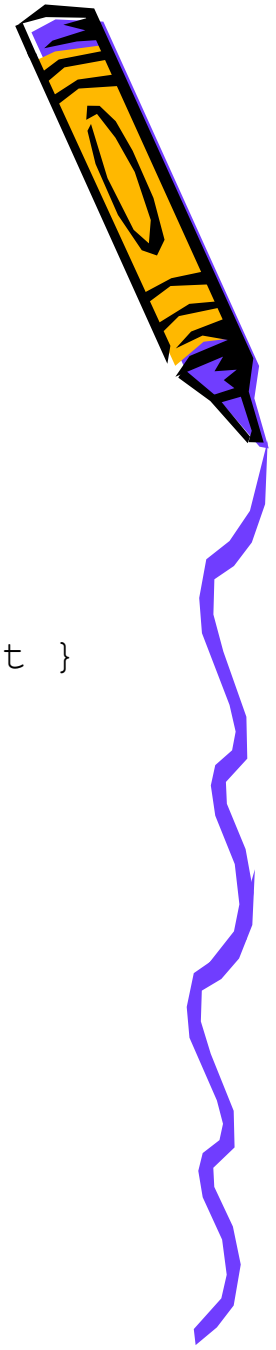
```
arrayVarName = new dataType[nbrRows][nbrCols];
```

1D Arrays (semantics)



- 1D Array Semantics
  - Define a variable that will refer to a memory location where the 1D array data begins
  - Index represents relative position in array
    - Ranges from `0` to `arrayVarName.length-1`

1D Arrays (use)



- 1D Array
  - For loop

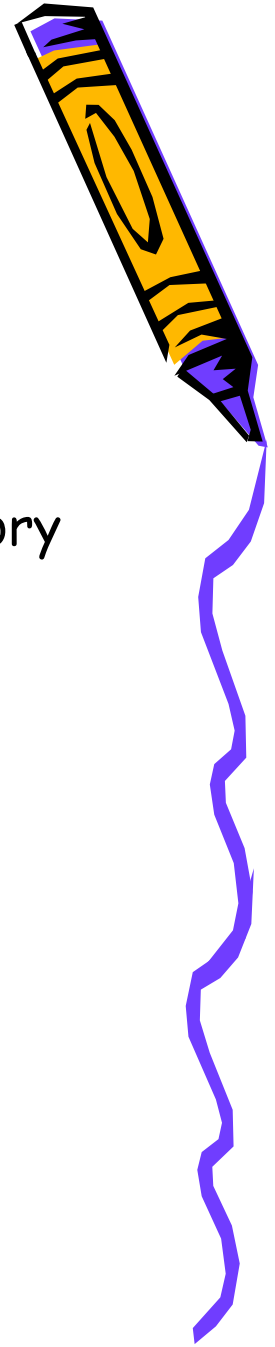
```
for (int idx=0; idx < arrayVarName.length; idx++)
{ //use idx to do something with each array element }
```

  - Foreach loop

```
for (arrayType element : arrayVarName)
{ //do something with each array element }
```

  - While loop
    - What would this look like?

1D Arrays (internals, API)



- 1D Array
  - Array variable is reference to array location in memory
    - Passing arrays to a method
    - Returning an array from a method
    - Copying arrays
  - Java API
    - Arrays class
      - Lots of static methods to manipulate 1D arrays, including
        - » copyOf, deepEquals, equals
    - System.arraycopy method

2D Arrays (syntax, semantics)

- Syntax
  - Define variable

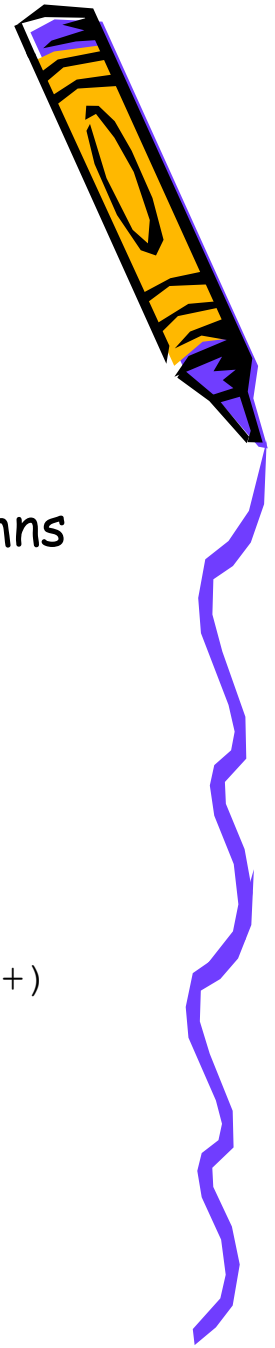
```
dataType[][] arrayVarName;     //preferred
dataType arrayVarName[][];
```

  - Allocate memory

```
arrayVarName = new dataType[nbrRows][nbrCols];
```

- Semantics
  - Define a variable that will refer to a memory location where the 2D array data begins
  - First index is row number
    - Ranges from `0` to `nbrRows-1`
  - Second index is column number
    - Ranges from `0` to `nbrCols-1`

2D Arrays (example)



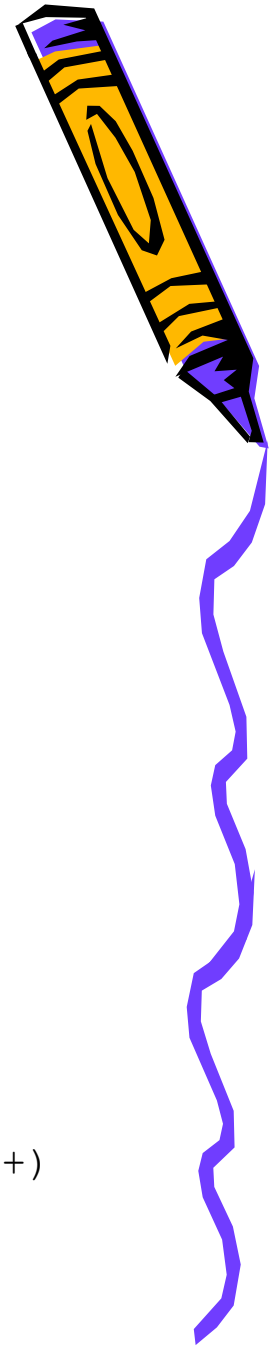
- Example
  - Define and initialize an array with 3 rows and 8 columns

```
int[][] numbers = { {1,3,5,7,9,11,13,15},
                    {2,4,6,8,10,12,14,16},
                    {2,3,5,7,11,13,17,19} };
```

  - Display contents of this array

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

2D Arrays (example #2)



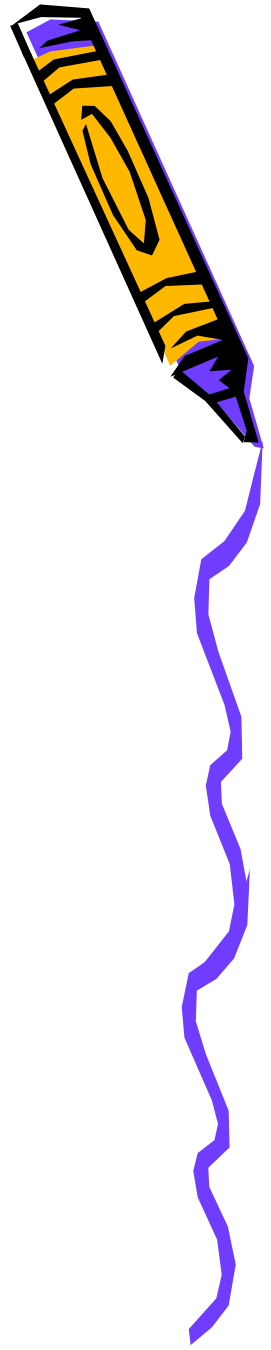
- Another example
  - Define and initialize a ragged array
    - Each row can have a different number of columns

```
int[][] numbers = { {1,2},
                    {1,2,3,4},
                    {1,2,3},
                    {1,2,3,4,5},
                    {1} };
```

  - Display contents of this ragged array

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

2D Arrays (example #3)



- Yet another example
  - Define an array that can contain any type of object

```
final int MAX = 4;
Object[][] objs;
```

  - Allocate memory

```
objs = new Object[MAX][MAX];
```

  - Store objects in rows 0 and 1 of this array

```
int row = 0;
for (int col=0; col < MAX; col++)
   objs[row][col] = new Integer(col * 10);
row++;
for (int col=0; col < MAX; col++)
   objs[row][col] = new Double(col * 0.5);
//etc.
```