

# WES-CS GROUP MEETING #11

## Exercise 1: Two Dimensional Arrays

First, draw a one-dimensional array containing the names of the people in your group. Then find out the answers to the following questions for each person:

0. In what month were you born?
1. What color are your eyes?
2. What is your favorite ice-cream flavor?

Fill in a two-dimensional array with one row for each person and three columns (one for each question), where row  $k$  contains the answers for person  $k$ .

Below is an example for a group of four people.

	[0]	[1]	[2]	[3]
names:	"Susan"	"David"	"Ann"	"Amy"

	<i>column 0</i>	<i>column 0</i>	<i>column 0</i>	
info:	"January"	"blue"	"vanilla"	<i>row 0</i>
	"December"	"blue"	"coffee"	<i>row 1</i>
	"November"	"blue"	"vanilla"	<i>row 2</i>
	"November"	"blue"	"coffee"	<i>row 3</i>

### Part (a)

Assume that you have an array called `names` that holds the names of the people in some group, and an array called `info` that holds the answers to some set of questions for that group (note that you shouldn't assume a particular number of people or a particular number of questions). For each of the following English descriptions, write the corresponding code.

- the expression that tells you the number of entries in the `names` array
- the expression that tells you the index of the last entry in the `names` array
- the expression that tells you the number of rows in the `info` array
- the expression that tells you the number of columns in row `k` of the `info` array
- code to print all values in row `k` of the `info` array (one per line)
- code to print all values in column `k` of the `info` array (one per line)
- code to print the value in the lower left corner of the `info` array
- code to print the value in the upper right corner of the `info` array
- code to print the value in the lower right corner of the `info` array

### Part (b)

On the next page is an incomplete `GroupInfo` class that has two array data members, `names` and `info`. Assume that those arrays have been initialized to hold the names of the people in your group and their answers to the three questions. Complete the `printPairs` methods by filling in the blanks so that when `print` is called it prints all pairs of people with the same answer to each of the three questions (the method should work for an array with any number of rows, not just the number for your group).

For example, for the two arrays given on the previous page, the output should be:

Same birth month: (Ann Amy)

Same eye color: (Susan David) (Susan Ann) (Susan Amy) (David Ann) (David Amy) (Ann Amy)

Same flavor: (Susan Ann) (David Amy)

Once you've filled in the blanks, trace the resulting code using the arrays you made for your group to be sure that it works correctly.

```

class GroupInfo {

    public String[] names;
    public String[][] info;

    public void print() {
        System.out.print("Same birth month: ");
        printPairs(0);
        System.out.println();
        System.out.print("Same eye color: ");
        printPairs(1);
        System.out.println();
        System.out.print("Same flavor: ");
        printPairs(2);
        System.out.println();
    }

    private void printPairs(int col) {
        // col is the column with the answers to the question of interest
        // print all pairs of names with matching answers in that column
        for (int j= ___ ; j< ___ ; j++) {
            String oneVal = info[ ___ ][ ___ ];
            for (int k = ___ ; k < ___ ; k++) {
                if (oneVal.equals(info[ ___ ][ ___ ])) {
                    System.out.print("(" + names[ ___ ] + " " + names[ ___ ] + ")");
                }
            }
        }
    }
}

```

## Exercise 2: Practice with Inheritance

The code shown below defines three classes: Drink, CocaCola, and Milk. Coca-Cola and Milk are each subclasses of Drink.

```
class Drink {
    private double myPrice;
    public Drink(double price) { myPrice = price; }
    public double getPrice() { return myPrice; }
    public String getName() { return "Drink"; }
    public void printLabel() {
        System.out.println( getName() );
        System.out.println("Price: " + getPrice());
    }
}

class CocaCola extends Drink {
    private static final double PRICE = 1.25;
    public CocaCola() { super(PRICE); }
    public String getName() { return "Coca-Cola Classic"; }
}

class Milk extends Drink {
    private static final double PRICE = .75;
    private int percentFat; // either 1 or 2
    public Milk( int fat ) {
        super(PRICE);
        percentFat = fat;
    }
    public String getName() { return "Milk"; }
    public int getFat() { return percentFat; }
}
```

Your job is to complete the `Test` class below by writing the `getDrink` method and adding the code in the `main` method that prints the fat content if the drink is milk.

```
import java.util.*;

class Test {

    // WRITE THE getDrink METHOD HERE

    public static void main(String[] args) {
        Drink drink = getDrink();
        drink.printLabel();
        // ADD CODE HERE TO PRINT THE FAT CONTENT IF drink IS MILK
    }
}
```

### Part (a)

The `getDrink` method should create and return a `Drink`, randomly choosing between `CocaCola` and `Milk`. If it chooses `Milk`, it should randomly choose between 2% and 1% for the fat content. (Note that the `getDrink` method should be a *static* method of the `Test` class.) Before writing the code, think about the following questions:

- What are the possible types that will actually be returned by the `getDrink` method?
- What should the return type of the `getDrink` method be?
- When `printLabel` is called (just after the call to `getDrink`), it causes a call to `getName`. There are three versions of that method, defined in the `Drink`, `Coca-Cola`, and `Milk` classes. What determines which version is actually called?

### Part (b)

What code could we add at the end of the `main` method that would figure out whether `drink` is a `Milk` object, and if so, would print its fat content?

### Exercise 3: Tic Tac Toe

For this exercise you will program a tic-tac-toe game.

To play tic-tac-toe, two players take turns placing their symbol (an X or an O) on an empty space in a 3-by-3 grid. A player wins if they get three in a row vertically, horizontally, or diagonally.

Below is an example to show how your program should work; your program would actually print the grids one under the other, but to save space they're shown here one next to the other.

```
  1 | 2 | 3      X | 2 | 3      X | 2 | 3
---+---+---    ---+---+---    ---+---+---
  4 | 5 | 6      4 | 5 | 6      4 | O | 6
---+---+---    ---+---+---    ---+---+---
  7 | 8 | 9      7 | 8 | 9      7 | 8 | 9

Player 1: 1      Player 2: 5      Player 1: 2
```

---

```
  X | X | 3      X | X | 3      X | X | X
---+---+---    ---+---+---    ---+---+---
  4 | O | 6      4 | O | 6      4 | O | 6
---+---+---    ---+---+---    ---+---+---
  7 | 8 | 9      O | 8 | 9      O | 8 | 9

Player 2: 7      Player 1: 3      You win!!!
```

You should define a `TicTacToe` class, with a main method that works as follows:

- (1) Create an instance of the `TicTacToe` class, with an initially empty grid (implemented using a two-dimensional array).
- (2) Display the grid on the screen.
- (3) Repeat until a player wins, or the grid is full:
  - Ask players alternately to place their symbols.
  - Read the number of the grid square entered by the player.
  - Check that the player has entered a legal move. If it is legal, put the player's symbol in the selected grid square; otherwise, print an error message and ask the player to enter a new choice.
- (4) If a player has won, print "You win!!"; if the grid is full, print "Game over!".

Below is a list of methods to implement in your `TicTacToe` class, and some questions to think about before you actually start writing the code.

- (1) A constructor that initializes the grid to be empty. (What should the types of the values stored in the grid be? How should they be initialized to represent an empty tic-tac-toe board? To choose the best values, think about what will make the other operations -- printing the grid, checking to see if a player has won -- easiest.)
- (2) A `moveOK` method that returns true if the player's move is legal. (What does it mean for a move to be legal? What should the method's parameters be? The player will enter the number of a grid square; how will you find the corresponding index into the grid array?)
- (3) A `doMove` method that puts the player's symbol in the square of the grid selected by the player. (What should the method's parameters be, and how should it work?)
- (4) A `gameWon` method that returns true if one of the players has won (has three symbols in a row horizontally, vertically, or diagonally).
- (5) A `gridFull` method that returns true if the grid is full.
- (6) A `printGrid` method that prints the current grid.

### Answers to the array description/code matching question

the number of entries in the names array	<code>names.length</code>
the index of the last entry in the names array	<code>names.length - 1</code>
the number of rows in the info array	<code>info.length</code>
the number of columns in row k of the info array	<code>info[k].length</code>
print all values in row k of the info array (one per line)	<pre>for (int j=0; j&lt;info[k].length; j++) {     System.out.println(info[k][j]); }</pre>
print all values in column k of the info array (one per line)	<pre>for (int j=0; j&lt;info.length; j++) {     System.out.println(info[j][k]); }</pre>
print the value in the lower left corner of the info array	<code>System.out.println(info[info.length-1][0]);</code>
print the value in the upper right corner of the info array	<code>System.out.println(info[0][info[0].length-1]);</code>
print the value in the lower right corner of the info array	<code>System.out.println(info[info.length-1][info[info.length-1].length-1]);</code>

## Answer to the GroupInfo code question

```
private void printPairs(int col) {
    // col is the column of interest
    // print all pairs of names with matching answers in that column
    // (j is the current row, and k iterates through the rest of the rows)
    for (int j=0; j<info.length; j++) {
        String oneVal = info[j][col];
        for (int k = j+1; k < info.length; k++) {
            if (oneVal.equals(info[k][col])) {
                System.out.print("(" + names[j] + ", " + names[k] + ")");
            }
        }
    }
}
```