Arrays

Arrays are like an ornament storage box.

1D

2D

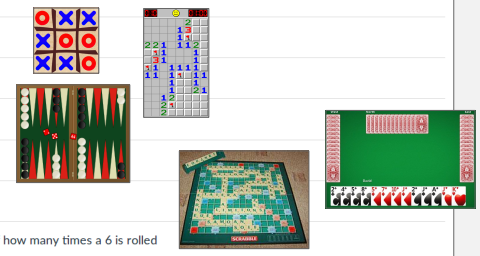
3D

- the number of dimensions = the number of indexes required to identify one compartment

key properties: multiple things of the same kind AND a grid or two indexes to refer to a particular value

In which of the following situations would a 2D array be appropriate? Choose all that apply.

- ★ the game tic-tac-toe
- ★ the game minesweeper
- ☐ the game backgammon
- ☐ the card game Hearts
- ★ the game Scrabble
- ☐ rolling dice: keeping track of how many times a 6 is rolled
- ☐ rolling dice: keeping track of how many times each value 1-6 is rolled (i.e. keeping track of how many 1s, how many 2s, etc)
- ★ rolling pairs of dice: keeping track of how many times each pair of values is rolled (i.e. keeping track of how often you roll two 1s, a 1 and a 2, a 1 and a 3, etc)



Consider the following array declaration:

```
int[][] numbers = new int[5][3];
```

Which of the following expressions refer to legal spots in the array? (i.e. they are not out of bounds) Choose all that apply.

- ☐ numbers[5][3]
- ☐ numbers[3][5]
- ★ numbers[4][1]
- ☐ numbers[1][4]
- ★ numbers[0][2]
- ☐ numbers[-1][2]
- ★ numbers[2][0]
- ☐ numbers[2][-1]

What value is printed when the following code is executed?

```
int[][] numbers = new int[5][3];
for ( int row = 0 ; row < 5 ; row++ ) {
   for ( int col = 0 ; col < 3 ; col++ ) {
     numbers[row][col] = row*col+row;
   }
}
System.out.println(numbers[2][1]);
```

- for a symmetric array, is it better to store it in a triangular shape or just have duplicate values?

  ☐ rolling pairs of dice: keeping track of how many times each pair of values is rolled (i.e. keeping track of how often you roll two 1s, a 1 and a 2, a 1 and a 3, etc)

  - if which die each value appeared on isn't important, `count[1][2]` and `count[2][1]` could both store the number of times one 1 and one 2 have been rolled

  – while it is possible to create a 2D array where the rows do not all have the same number of columns, this is not usually done
  – instead, decide which slot to use (e.g. `count[i][j]` where $i \leq j$) and leave the other empty

- can you turn a 1D array into a 2D array, similar to how a 1D array is resized?

```
int[] a = new int[5];
int[] b = new int[a.length*2];
…
a = b;
```

```
int[] a = new int[5];
int[][] b = new int[5][5];
…
a = b;
```

`int[]` and `int[][]` are not compatible types, so the assignment statement is not legal