

# CS 113 – Computer Science I

## Lecture 12 – Loops & Arrays

Thursday 02/29/24

# Announcements

- HW4 due Sunday
- Midterm next week

# Example: For Loop

initialize

condition

update

```
for (int count = 0; count < 6; count++) {  
}
```

## Exercise 2: **abecedarian**

A word is said to be “abecedarian” if the letters in the word appear in alphabetical order.

Write a method **isAlphaOrder** that takes a word (of any length) and returns a boolean indicating if the letters in the word appear in alphabetical order

**Use a for loop**

# Exercise 3: Isopsephy

Find the numeric value of a word by summing the values of the characters (a = 1, b = 2, ....)

**Use a for loop**

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z

# Nested Loops

# Code Example

```
1. for (int i = 0; i <= 3; i++) {  
2.     for (int j = 0; j <= 3; j++) {  
3.         System.out.print(i + ", " + j + " ");  
4.     }  
5.     System.out.println();  
6. }
```





# Code Example

```
for (int i = 0; i <= 3 ; i++) {  
    for (int j = 0; j <= 3; j++) {  
        System.out.print(i + ", " + j + " ");  
    }  
    System.out.println();  
}
```

i	j	i <= 3	j <= 3
0	0	T	T
0	1	T	T
0	2	T	T
0	3	T	T
0	4	T	F
1	0	T	T
1	1	T	T
1	2	T	T
1	3	T	T
1	4	T	F
2	0	T	T
...	...	...	....

# Exercise: Days in a week

Write a program to print the “even days” in 3 weeks

Week: 1

Day: 2

Day: 4

Day: 6

Week: 2

Day: 2

Day: 4

Day: 6

Week: 3

Day: 2

Day: 4

Day: 6

# Exercise: Print a square

```
$ java Square
```

```
Enter a size: 5
```

```
*****
```

```
*****
```

```
*****
```

```
*****
```

```
*****
```

```
$ java Square
```

```
Enter a size: 2
```

```
**
```

```
**
```

```
$ java Square
```

```
Enter a size: 1
```

```
*
```

# What does this code print?

```
for (int i = 0; i < size; i++) {  
    for (int j = 0; j <= i; j++) {  
        System.out.print("* ");  
    }  
    System.out.println();  
}
```

# Exercise: Spelling

Write a method called `canSpell` that takes two strings (letters and word) and checks whether the set of letters can spell the word.

# Exercise: LoopPattern.java

```
$ java LoopPattern
```

```
Enter a length: 5
```

```
*_*_*
```

```
$ java LoopPattern
```

```
Enter a length: 10
```

```
*_*_*_*_*_*
```

```
$ java LoopPattern
```

```
Enter a length: 0
```

```
$ java LoopPattern
```

```
Enter a length: 1
```

```
*
```

# Agenda

- Nested Loops review
- Arrays of Arrays

# Code Example

```
1. for (int i = 0; i <= 3; i++) {  
2.     for (int j = 0; j <= 3; j++) {  
3.         System.out.print(i + ", " + j + " ");  
4.     }  
5.     System.out.println();  
6. }
```



# Code Example

```
for (int i = 0; i <= 3 ; i++) {  
    for (int j = 0; j <= 3; j++) {  
        System.out.print(i + ", " + j + " ");  
    }  
    System.out.println();  
}
```

i	j	i <= 3	j <= 3

# Code Example

```
for (int i = 0; i <= 3 ; i++) {  
    for (int j = 0; j <= 3; j++) {  
        System.out.print(i + ", " + j + " ");  
    }  
    System.out.println();  
}
```

i	j	i <= 3	j <= 3
0	0	T	T
0	1	T	T
0	2	T	T
0	3	T	T
0	4	T	F
1	0	T	T
1	1	T	T
1	2	T	T
1	3	T	T
1	4	T	F
2	0	T	T
...	...	...	....

# What does this code print?

```
for (int i = 0; i < size; i++) {  
    for (int j = 0; j <= i; j++) {  
        System.out.print("* ");  
    }  
    System.out.println();  
}
```

# Exercise: Nested loops

```
$ java Rectangle  
Enter a width: 2  
Enter a height: 4  
**  
**  
**  
**
```

```
$ java Rectangle  
Enter a width: 2  
Enter a height: 2  
**  
**
```

```
$ java Rectangle  
Enter a width: 7  
Enter a height: 2  
*****  
*****
```

# Agenda

- Nested loops review
- **Arrays of Arrays**

# Arrays

Three ways to initialize an array

1. With an initial value

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

2. With allocated space, but uninitialized

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

3. With an empty array reference

```
int[] numbers = null;
```

# Array Indexing

Access individual elements of an array with indexing

`array[index]`

We use *zero*-based indexing

first element is **0**

last element is **length-1**

Accessing indices out of range results in a **runtime error!**

# Iterating through an array

Write a method called `printArray` that takes in an array of integers and prints out the values in the array:

*`printArray({1,2,3,4})` -> "1 2 3 4"*



# Array Comparison

we can't use "==" to compare arrays

Strings and arrays are **NOT** primitives

They are objects

# Arrays of Arrays

`int[] array1` is an array of ints

`String[] array2` is an array of Strings

What is `int[][] array3`?

An array of integer arrays

What is `String[][] array4`?

An array of String arrays

# 2D array example

What does `int[][] array = new int[4][3]` look like?

# 2D array example

What does `int[][] array = new int[4][3]` look like?

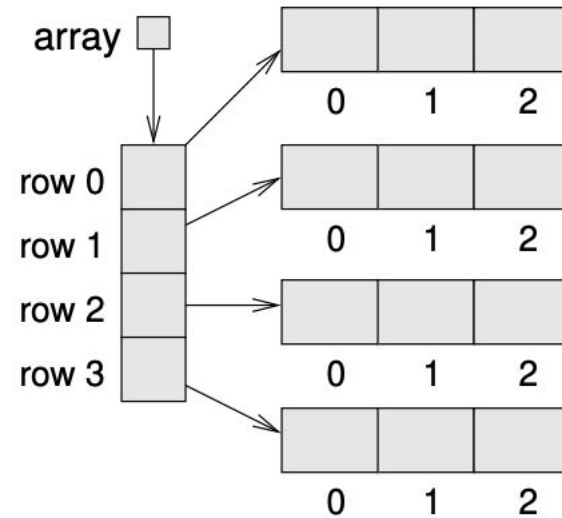


Figure 15.3: Storing rows and columns with a 2D array.

# Declaring and Initializing Arrays

```
int[][] matrix1 = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9}  
};
```

```
int[][] matrix2 = new int[3][4]; //can fill with a loop
```

# 2D Array

Useful for representing a:

- Grid
- Boardgame
- Matrix
- Table
- ...

# Looping Over a 2D array

code

# Array Example

Given a 2-D array, compute the average of all elements

<b>11</b>	<b>12</b>	<b>13</b>	<b>1</b>	<b>6</b>
<b>16</b>	<b>17</b>	<b>18</b>	<b>9</b>	<b>8</b>



# Array Example

write a method **fill** which takes two ints (row and col) and an int[][] and fills that position with the number 100

# Array Example

Given a square 2-D array, compute the sum of the diagonal

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>
<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>
<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
<b>21</b>	<b>22</b>	<b>23</b>	<b>24</b>	<b>25</b>

# Array Example

Given a 2-D array, compute the sum of the perimeter

<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>2</b>	<b>2</b>
<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>3</b>	<b>6</b>
<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>	<b>1</b>	<b>6</b>
<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>	<b>9</b>	<b>8</b>