

# Lesson 3-2:

# Classes and Methods

Computer Science 46A: Introduction to Programming

San José State University

# Announcements

- Start Homework #1 (Doc says Homework 3)
  - Where is the writeup?
  - I will provide previous homework solution to bootstrap you
- Lab #2 tomorrow
  - Where does indexing begin for Java
- Zybooks
  - Will drop one lowest grade

# Learning Objectives

By the end of this lesson, you should be able to:

1. Use static final variables to eliminate “magic numbers”
2. Use the "this" reference for code clarity
3. Use local variables inside methods

# Magic Numbers

# A return to the Circle Example

- In the Circle class from Lecture 3-1, we defined the variable PI in each of our methods
- This variable has the following characteristics:
  - It is used in more than one method
  - It is buried in the code
  - We might want to update the value with a more exact value
- When defined in this way, the value 3.1 is a "magic number"

# “Final Static” Variables

- Final static variables are constants that never change while using the class
  - Static = same for every instance of the class (i.e. when you make objects)
  - Final = value of the variable can never change when you use the class
- Declared similar to instance variables:  
outside the constructors and methods
- Syntax:  

```
private final static [variable type] [variable name] = [value];
```

See revised code in Example: [Circle](#)

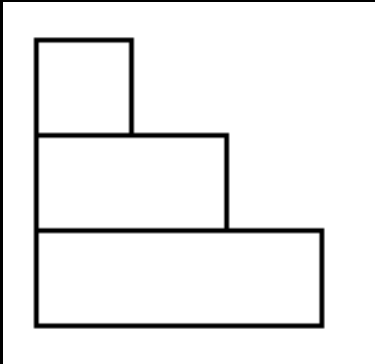
# Style Alert: Magic Numbers

- Classes should no longer have magic numbers! 😊
- In other words, all constants should be defined in the header of your code
- Convention for constants is the SCREAMING\_SNAKE\_CASE
  - All letters capitalized, separated by spaces
- Example:
  - `private final static int A_GOOD_NAME = 3;`

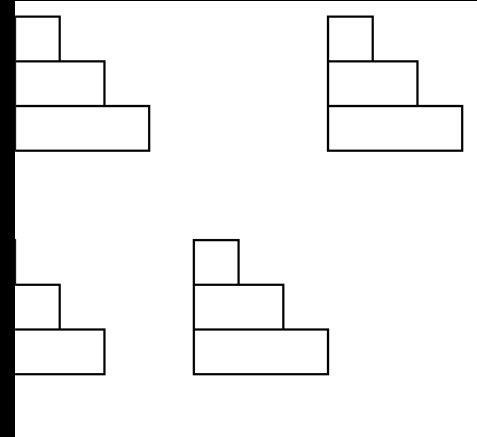
# Overview of Participation Exercise 3-2a:

## Stair

1) Write a class with a method `draw()` to draw a 3-rectangle stair set



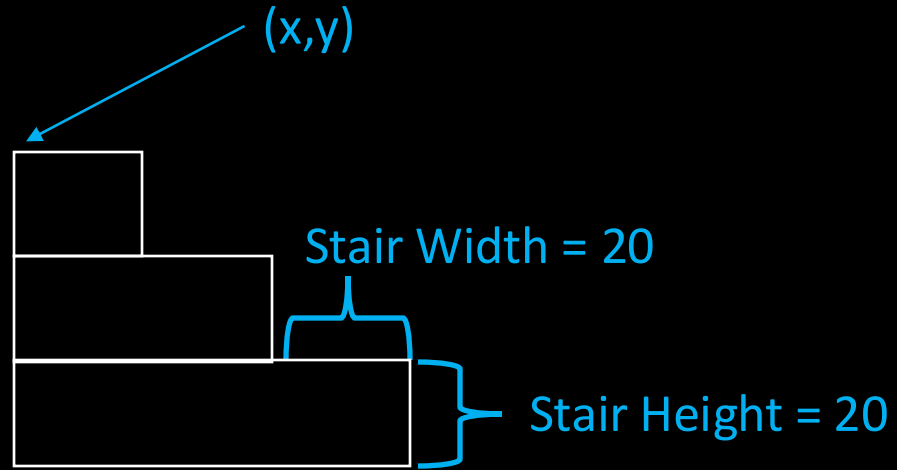
2) Use the `StairTester` class to draw 4 different rectangles



Results of `StairTester.java`

Codecheck Link: [HERE](#) and on Canvas

# Exercise 3-2a: Mapping out the Dimensions



Recall: a Rectangle object is declared and assigned with the following syntax:  
`Rectangle [variable name] = new Rectangle(x, y, width, height);`

# Instructions for Participation Exercise 3-2a:

## Stair

- 1) View the `StairWithMagicNumbers` class to see how the main method works
- 2) Fill in steps 1-4 as you did for previous exercises (constructors and methods)
- 3) Fill in the draw method by generalizing the `StairWithMagicNumbers` class
  - 1) Copy/paste the code in the draw() method
  - 2) Edit to remove magic numbers

Note: This problem will be very similar to Homework 3 Problem C

# The “this” keyword

See the [Year.java](#) in Examples for a demo

# Poll Everywhere

- [PollEv.com/narayanbalasubramanian644](https://PollEv.com/narayanbalasubramanian644),
- [https://PollEv.com/multiple\\_choice\\_polls/wbixCExSdM9Xzm8GUEW95/respond](https://PollEv.com/multiple_choice_polls/wbixCExSdM9Xzm8GUEW95/respond)

# Poll Everywhere

What is the output of the `YearTester` class?

A) 2016

B) 2018


C) None of the above

# The `this` reference

- To avoid confusion, Java has a special keyword “`this`” which can be used to explicitly access instance variables in the current object
- Variables referenced in a method are, by default, those declared within the method
  - If a variable is not declared in a method, then it is search for in the instance variables
- All instance variables are implicitly those of the `this` object unless a variable of the same name is defined inside a method

# Using the printYear() method for the instance variable:

```
public void printYear(int year)
{
    System.out.print(this.year);
}
```

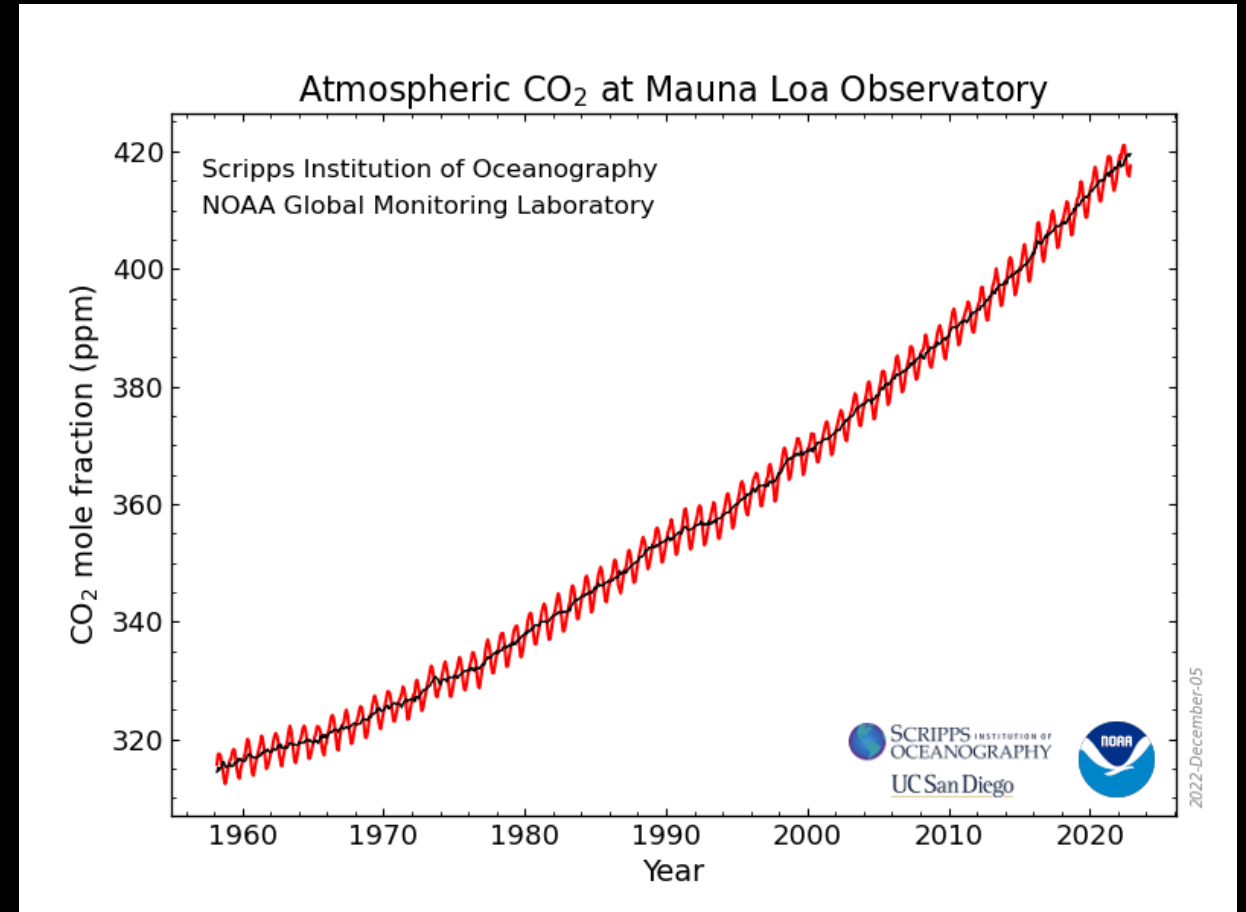


With the object this,  
this.year refers to value  
stored in the instance  
variable year

# Local Variables

# Exercise 3-2b: An example in Climate Science

- Increasing carbon dioxide levels in our atmosphere is one of the main drivers of climate change
- Carbon dioxide levels have been measured each year since 1958
- Let's write a class to determine the CO<sub>2</sub> levels in a given year



# Overview of Participation Exercise 3-2b:

## CarbonDioxide

Write a class called `CarbonDioxide` which has:

A constructor:

- `CarbonDioxide()`

Five methods

- `getLevel()`
- `getYear()`
- `setLevel()`
- `setYear()`
- `levelMessage()`

```
1980
Expected: 1980
2022
Expected: 2022
346.0
Expected: 346.0
409.0
Expected: 409.0
The carbon dioxide level in 1980 is 346.0 ppm.
Expected: The carbon dioxide level in 1980 is 346.0 ppm.
The carbon dioxide level in 2022 is 409.0 ppm.
Expected: The carbon dioxide level in 2022 is 409.0 ppm.
1990
Expected: 1990
The carbon dioxide level in 1990 is 361.0 ppm.
Expected: The carbon dioxide level in 1990 is 361.0 ppm.
```

Results of `CarbonDioxideTester.java`

Codecheck Link: [HERE](#) and on Canvas

# The `setLevel()` method

The formula to calculate the CO2 concentration is:

$$\text{level} = 1.5 * (\text{year} - 1958) + 313$$

But there's two issues with this statement:

1. It has magic numbers
2. It is not written in a descriptive way

We can solve the second issue by implementing local variables

# Local Variable Characteristics

- Local variables are variables which are only used *inside* a method
- They are special in that they are removed from memory after the method is complete
- No other method can access local variables
- Local variables can be very helpful in improving the readability of your code

# Rewrite the `setLevel()` method

Rewrite this formula:

$$\text{level} = 1.5 * (\text{year} - 1958) + 313$$

By defining three local variables:

- `numberOfYears` – the number of years since 1958
- `totalIncrease` – the total increase in the CO<sub>2</sub> level over `numberOfYears`
- `newLevel` – the level of CO<sub>2</sub> in 1958 plus the `totalIncrease`

# The `setLevel()` method revised

Previous formula:

```
level = 1.5*(year - 1958) + 313;
```

New formulation:

```
numberOfYears = year - 1958;
```

```
totalIncrease = 1.5*numberOfYears;
```

```
newLevel = totalIncrease + 313;
```

# Final note for Participation Exercise 3-2b:

## CarbonDioxide

The `CarbonDioxide` class should have 3 constants:

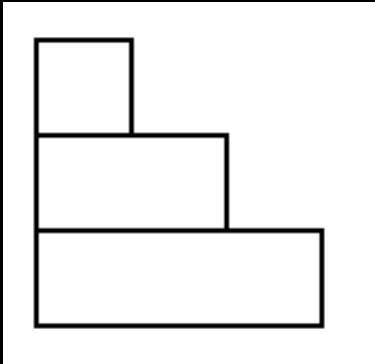
- `FIRST_YEAR = 1958;`
- `CO2_LEVEL_1958 = 313;`
- `CO2_INCREASE_PER_YEAR = 1.5;`

Write the `setLevel()` method with your new formula, replacing any constants with the above constants to avoid “magic numbers”

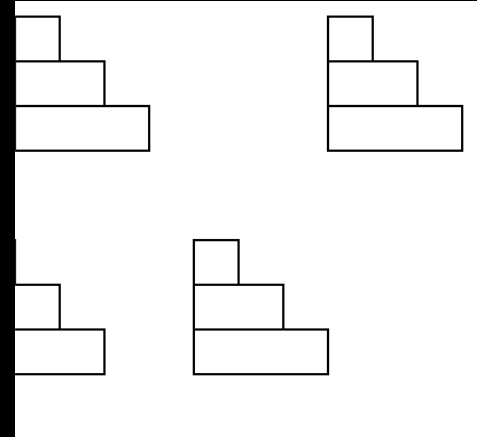
# Participation Exercises

# Participation Exercise 3-2a: **Stair**

1) Write a class with a method `draw()` to draw a 3-rectangle stair set



2) Use the `StairTester` class to draw 4 different rectangles



Results of **StairTester.java**

Codecheck Link: [HERE](#) and on Canvas

# Participation Exercise 3-2b: **CarbonDioxide**

Write a class called **CarbonDioxide** which has:

A constructor:

- **CarbonDioxide()**

Five methods

- **getLevel()**
- **getYear()**
- **setLevel()**
- **setYear()**
- **levelMessage()**

```
1980
Expected: 1980
2022
Expected: 2022
346.0
Expected: 346.0
409.0
Expected: 409.0
The carbon dioxide level in 1980 is 346.0 ppm.
Expected: The carbon dioxide level in 1980 is 346.0 ppm.
The carbon dioxide level in 2022 is 409.0 ppm.
Expected: The carbon dioxide level in 2022 is 409.0 ppm.
1990
Expected: 1990
The carbon dioxide level in 1990 is 361.0 ppm.
Expected: The carbon dioxide level in 1990 is 361.0 ppm.
```

Results of **CarbonDioxideTester.java**

Codecheck Link: [HERE](#) and on Canvas