COMPSCI 121: BASIC METHODS & CLASSES

SPRING 20

BYTE FROM COMPUTING HISTORY

Grace Hopper developed the first computer language, which eventually became known as COBOL.

GOALS FOR TODAY'S CLASS

- **Demo in JGRASP**
  - How classes work together
- **Lecture**
  - More about Classes, Objects, Methods, & Variables

IMPORTANT TO NOTE

Exam 1 covers content from chapters 1 to 3 (ignore the optional sections).

You'll get practice worksheets and a sample exam.

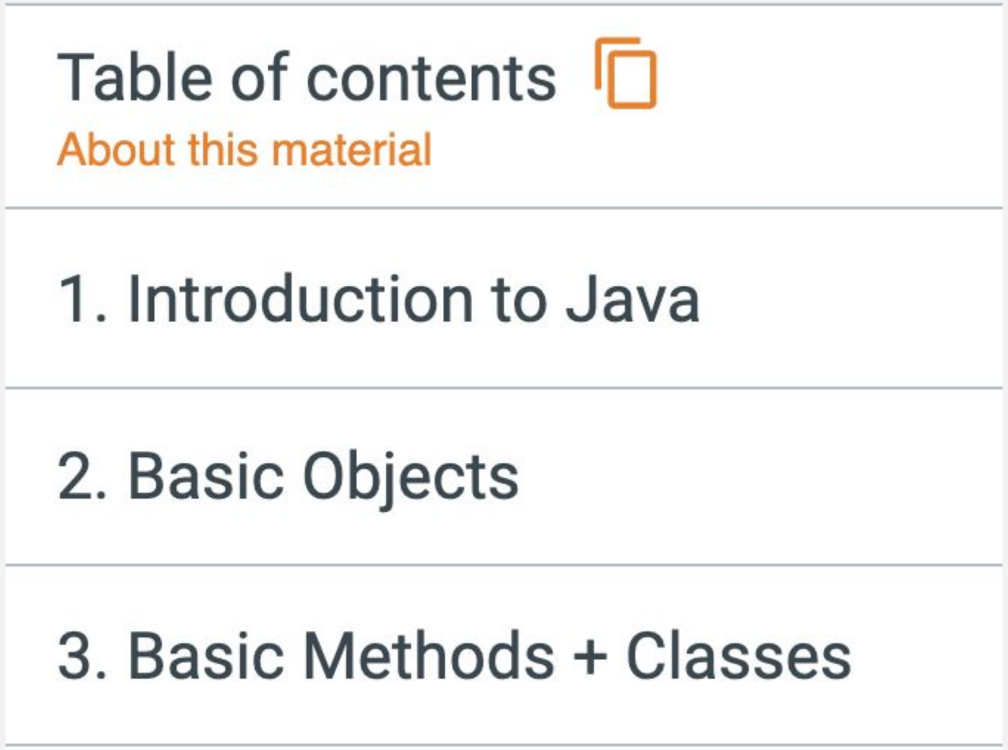
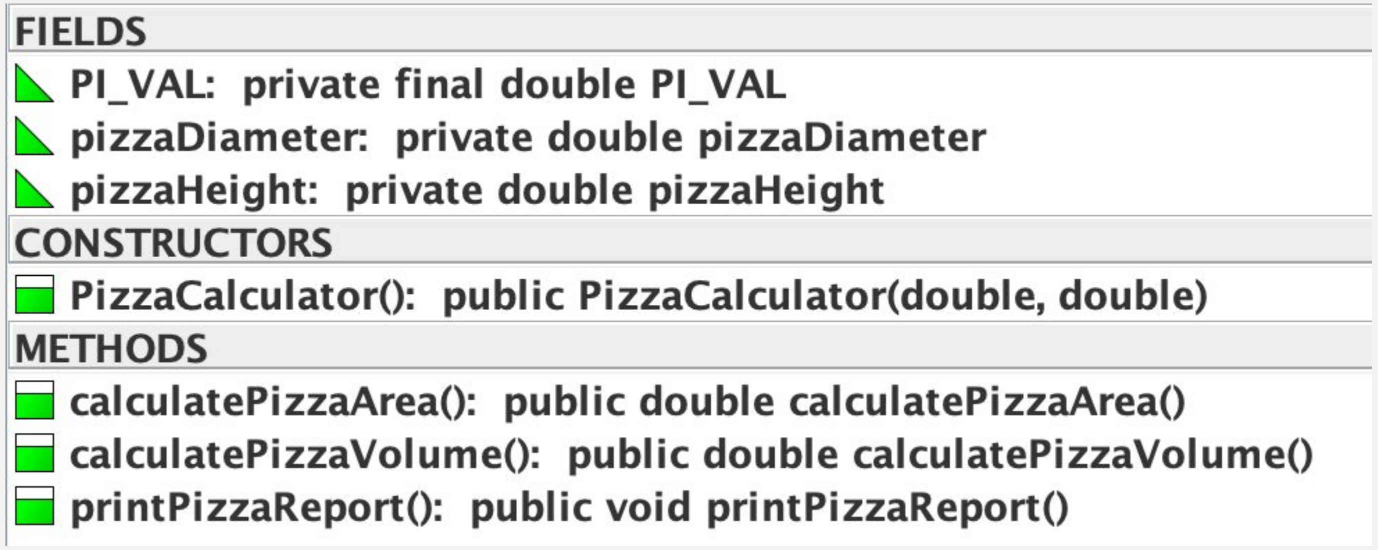


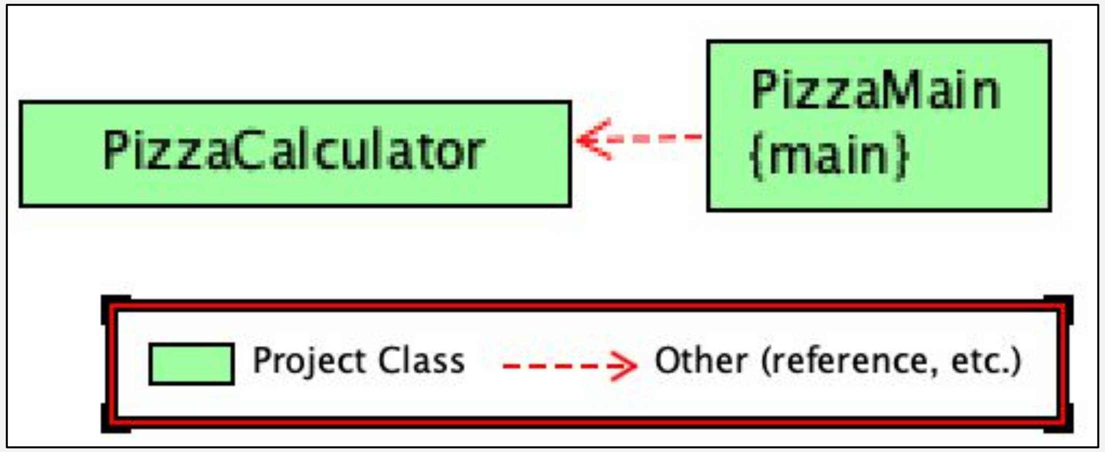
Table of contents

About this material

PIZZA PROJECT - DESIGN - T-P-S



FIELDS
PI_VAL: private final double PI_VAL
pizzaDiameter: private double pizzaDiameter
pizzaHeight: private double pizzaHeight
CONSTRUCTORS
PizzaCalculator(): public PizzaCalculator(double, double)
METHODS
calculatePizzaArea(): public double calculatePizzaArea()
calculatePizzaVolume(): public double calculatePizzaVolume()
printPizzaReport(): public void printPizzaReport()

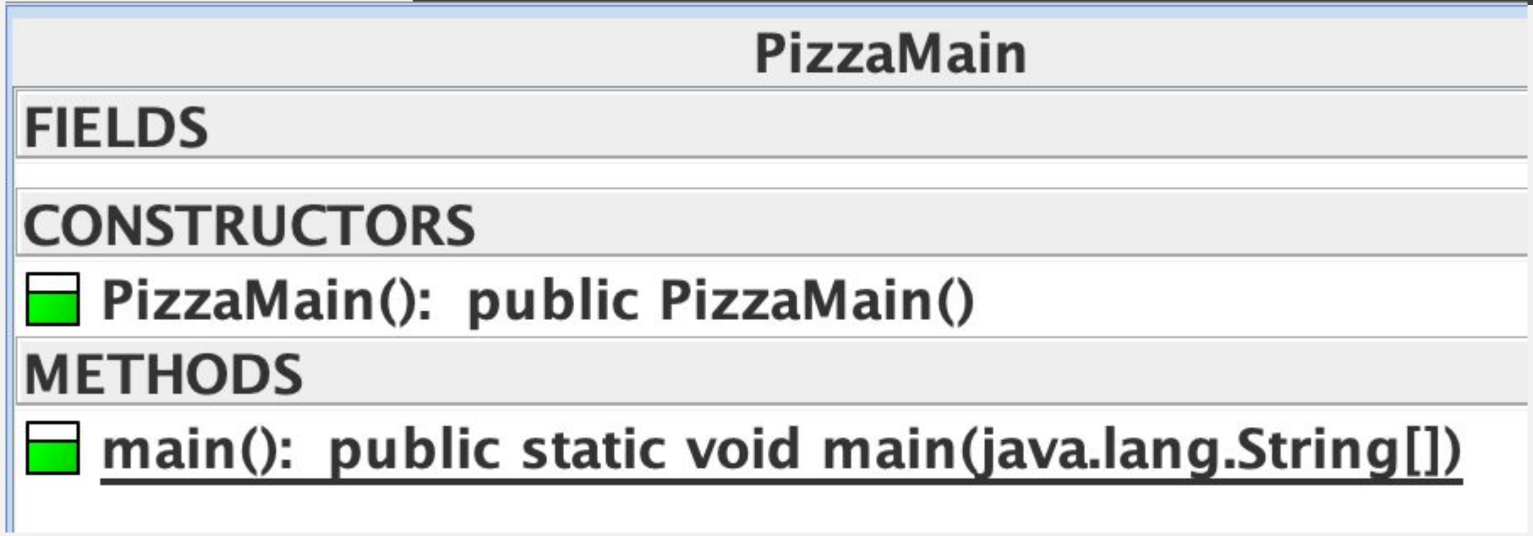


1. What does `PizzaCalculator` do?
2. What information does `PizzaMain` *send to / receive from* `PizzaCalculator`?

PIZZA PROJECT DESIGN - T-P-S ANSWERS

1. What does `PizzaCalculator` do?
   It calculates the volume and area of a pizza.
2. What information does `PizzaMain` send to / receive from `PizzaCalculator`?
   Sends: arguments for constructor: diameter and height.
   Receives: area, volume of pizza.

PiazzaMain _ User Interface


PizzaMain

FIELDS

CONSTRUCTORS

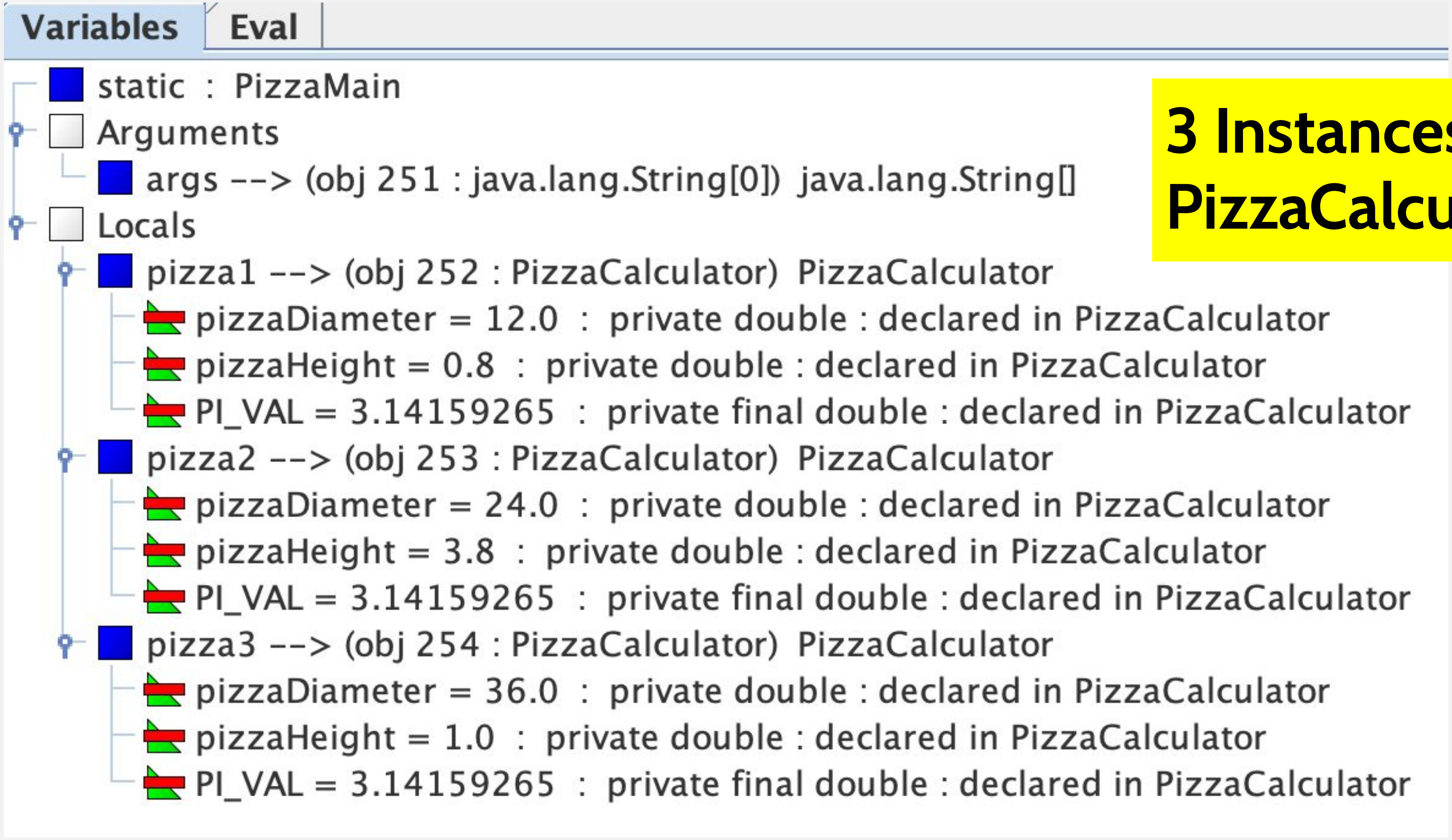
🟩 PizzaMain(): public PizzaMain()

METHODS

🟩 main(): public static void main(java.lang.String[])

The main class creates new instances of `PizzaCalculator`.
Calls the `calculatePiazzaArea,` `calculatePizzaVolume` and `printPizzaReport` methods.

DEBUGGER VIEW OF CLASS VARIABLES



```
static : PizzaMain
Arguments
    args --> (obj 251 : java.lang.String[0])  java.lang.String[]
Locals
    pizza1 --> (obj 252 : PizzaCalculator)  PizzaCalculator
        pizzaDiameter = 12.0  :  private double : declared in PizzaCalculator
        pizzaHeight = 0.8  :  private double : declared in PizzaCalculator
        PI_VAL = 3.14159265  :  private final double : declared in PizzaCalculator
    pizza2 --> (obj 253 : PizzaCalculator)  PizzaCalculator
        pizzaDiameter = 24.0  :  private double : declared in PizzaCalculator
        pizzaHeight = 3.8  :  private double : declared in PizzaCalculator
        PI_VAL = 3.14159265  :  private final double : declared in PizzaCalculator
    pizza3 --> (obj 254 : PizzaCalculator)  PizzaCalculator
        pizzaDiameter = 36.0  :  private double : declared in PizzaCalculator
        pizzaHeight = 1.0  :  private double : declared in PizzaCalculator
        PI_VAL = 3.14159265  :  private final double : declared in PizzaCalculator
```

3 Instances of PizzaCalculator

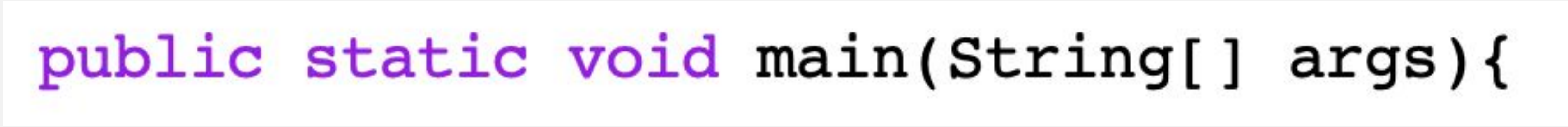
They each contain their own data.

DEMO: PIZZA PROJECT

You saw:

1. **The PizzaMain class need not know how the PizzaCalculator class' data and methods are implemented, but need only understand how each public member method behaves.**
2. **A programmer can create one or more objects of the same class**
   a. declare a reference variable of the class type.
   b. assign the variable with an instance of the class type.
3. **The `new` operator explicitly allocates an object of the specified class type.**
4. **The "." operator is used to invoke a method on an object.**
5. **Within a member method, the implicitly-passed object reference is accessible via the keyword `this`**

REMINDER: MAIN METHOD



```
public static void main(String[] args){
```

`main()` is a static method, which means `main()` does not have direct access to the class' instance members. A programmer <u>must</u> create objects within `main()` to call instance methods.

THE this. keyword

```
   /* Constructor that initializes diameter and height.
   Notice use of keyword "this".
    */
public PizzaCalculator(double pizzaDiameter, double pizzaHt){
   this.pizzaDiameter = pizzaDiameter;
   pizzaHeight = pizzaHt;
}
```

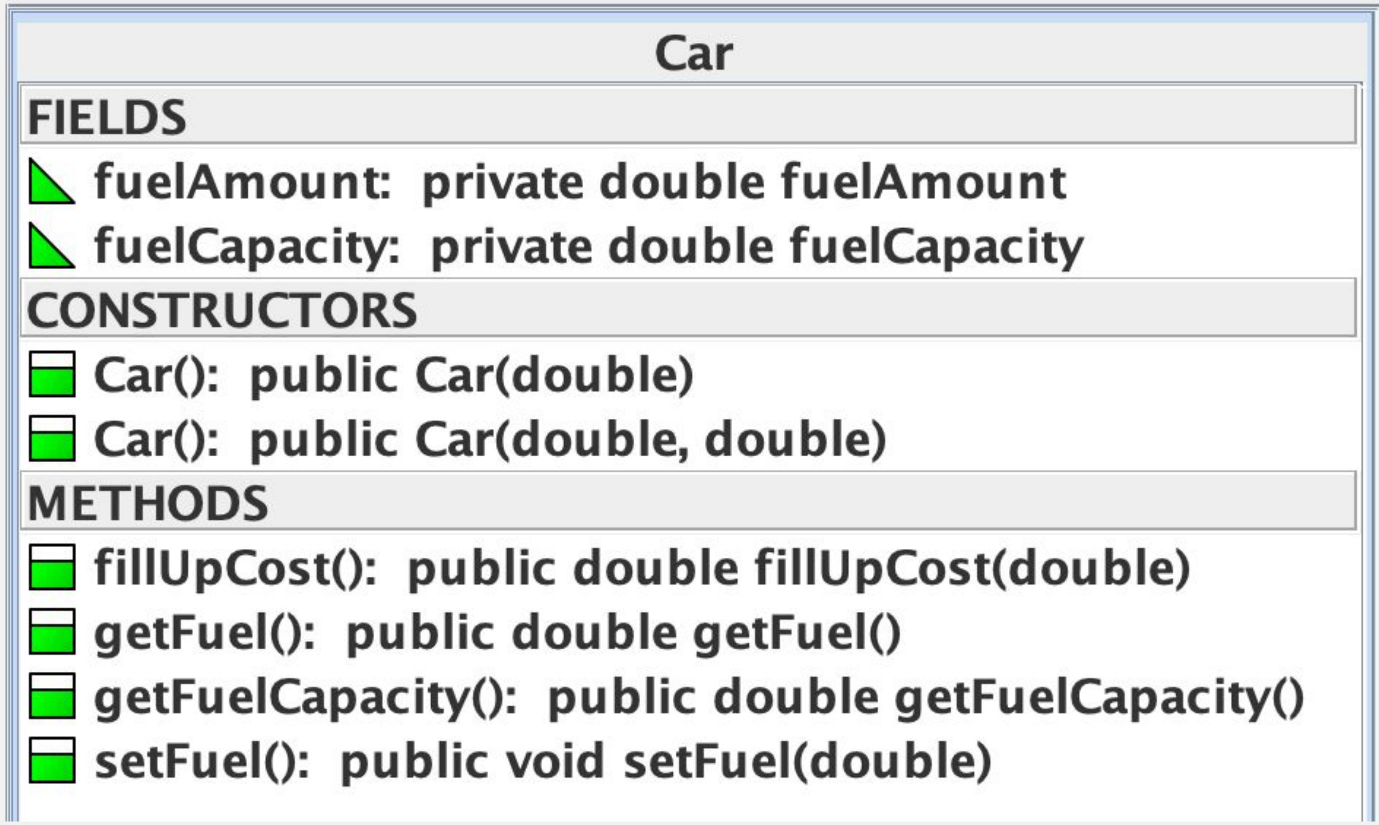
Using `this` makes clear that a class member is being accessed. Such use is essential if a field member and parameter have the same identifier because the parameter name dominates.

CLICKER QUESTION #1

A Car class (from last week) object is instantiated:

```
Car myCar = new Car(13.5, 5.0);
```

Which one of the following correctly accesses the fuel amount attribute of `myCar`?



Car

FIELDS

fuelAmount: private double fuelAmount

fuelCapacity: private double fuelCapacity

CONSTRUCTORS

Car(): public Car(double)

Car(): public Car(double, double)

METHODS

fillUpCost(): public double fillUpCost(double)

getFuel(): public double getFuel()

getFuelCapacity(): public double getFuelCapacity()

setFuel(): public void setFuel(double)

```
A. myCar.fuelAmount;
B. myCar.getFuelAmount();
C. myCar.getFuel;
D. myCar.getFuel();
```

CLICKER QUESTION #1

A Car class (from last week) object is instantiated:

```
Car myCar = new Car(13.5, 5.0);
```

Which one of the following correctly accesses the fuel amount attribute of `myCar`?

A. `myCar.fuelAmount;` private - can't access

B. `myCar.getFuelAmount();` not a method in Car class

C. `myCar.getFuel;` not a correct method call

D. `myCar.getFuel();` CORRECT

CLICKER QUESTION #2

Given the method definition below, indicate which is a **valid** `return` statement:

```
int calculate(int num1, int num2) { ... }
```

A. `return num1, num2;`

B. `return;`

C. `return num1 * num2;`

D. `return (num1, num2);`

CLICKER QUESTION #2 ANSWER

Given the definition below, indicate which is a valid `return` statement:

```
int calculate(int num1, int num2) { ... }
```

**A return statement can return only one value.

A. `return num1, num2;` ** Wrong

B. `return;` must return int value

C. `return num1 * num2;` CORRECT

D. `return (num1, num2);` ** Wrong

CLICKER QUESTION #3

Choose the incorrect statement

A. Creating methods helps `main` run faster.
B. Decomposing a program into methods aids program readability.
C. A method can be defined once, then called from multiple places in a program.
D. There can be only one `main` method in a class.

CLICKER QUESTION #3 ANSWER

Choose the incorrect statement

A. Creating methods helps `main` run faster.
B. Decomposing a program into methods aids program readability. CORRECT
C. A method can be defined once, then called from multiple places in a program. CORRECT
D. There can be only one `main` method in a class. CORRECT

More methods may cause slightly slower program execution - but improve readability.

CLICKER QUESTION #4

Choose the incorrect statement/s

1. Only one constructor can be declared in a class.
2. A constructor must have a return type.
3. A constructor must have at least one parameter.
4. A constructor must be called to make an instance of the class.

A. 1, 2, 3, 4
B. 1, 2, 3
C. 2, 3, 4
D. 1, 2

CLICKER QUESTION #4

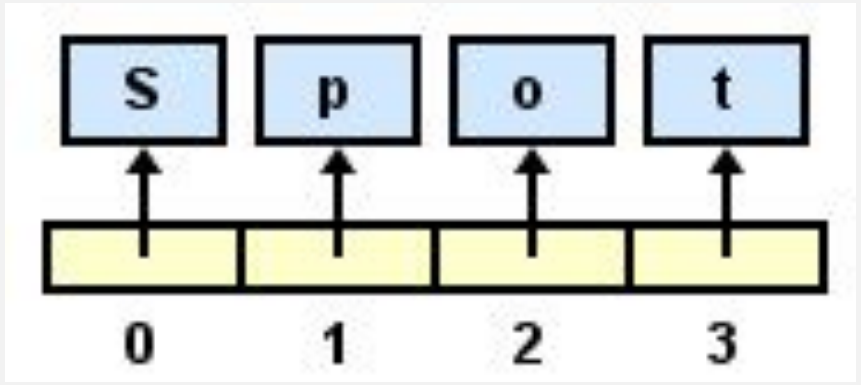
Choose the incorrect statement/s

1. Only one constructor can be declared in a class. INCORRECT
2. A constructor must have a return type. INCORRECT
3. A constructor must have at least one parameter. INCORRECT
4. A constructor must be called to make an instance of the class. CORRECT

A. 1, 2, 3, 4
B. 1, 2, 3
C. 2, 3, 4
D. 1, 2

INTRODUCTION TO STRING CLASS METHODS (1)

Strings are made up of characters. The characters in a particular string hold fixed positions in that string, beginning with position 0 and not at 1.



```
String pupName = “Spot”;

char ch = pupName.charAt(1);
// ch is assigned ‘p’

ch = pupName.charAt(0);
// ch  is assigned ‘S’
```

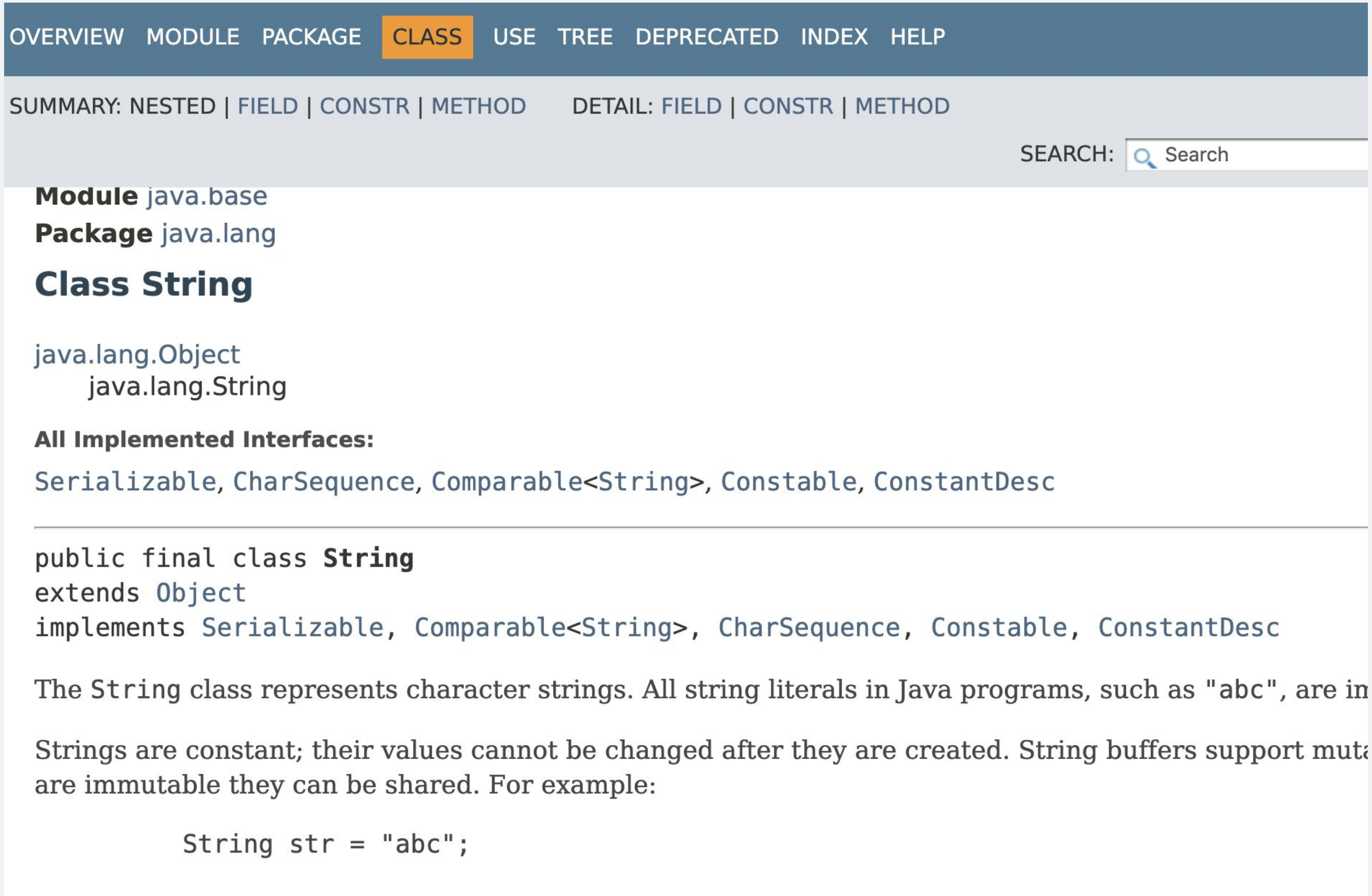
STRING CLASS METHODS (2)

```
String pupName = “Spot”;
int len = pupName.length();//len assigned 4
String huh = pupName.concat(“less”); //
huh is assigned to: “Spotless”
String bigHuh = pupName.toUpperCase();
bigHuh is assigned to: “SPOT”
```

Note: Strings are immutable – never change.
pupName is still “`Spot`”;
Some String methods need arguments.

JAVA API - STRING CLASS

https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/lang/String.html


OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH: Search

Module java.base
Package java.lang

Class String

java.lang.Object
    java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>, Constable, ConstantDesc

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence, Constable, ConstantDesc
```

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are in

Strings are constant; their values cannot be changed after they are created. String buffers support mut
are immutable they can be shared. For example:

```
        String str = "abc";
```

CLICKER QUESTION #5

Java's `String` class has a `length()` method that returns the length of the string.

What is returned by the call to the `length` method below?

```
String greetingStr = "Hello";
greetingStr.length();
```

A. 7

B. 5

C. 8

D. 6

CLICKER QUESTION #5 ANSWER

Java's `String` class has a `length()` method that returns the length of the string.

What is returned by the call to the length method below.

```
String greetingStr = "Hello";
greetingStr.length();
```

A. 7

B. 5

C. 8

D. 6

SUMMARY: PARAMETERS & ARGUMENTS

- *Parameter*: method input specified in method definition.
  - Upon a *call*, parameter's memory location is allocated, and parameter is assigned with argument's value.
  - Upon *return*, parameter is deleted from memory.
  - Method definition may have multiple parameters, separated by commas.
  - A method definition with no parameters must still have empty parentheses()- the call must include parentheses, with no argument.
- *Argument*: value provided to method's parameter during method call.
  - Parameters are assigned with argument values by position: First parameter with first argument, second with second, etc.

WEEK 3 TO-DO LIST:

- Check your iClicker grades in Moodle.
- Complete zyBook chapters 1-3 exercises (content for exam).
- Communicate with us using only Moodle Private Forum or Piazza.