COMPSCI 121: BASIC OBJECTS

SPRING 2020 TUESDAY LECTURE

BYTE FROM COMPUTING HISTORY

James Gosling is known as the '*Father of the Java programming language*'.

Listen to an interview.

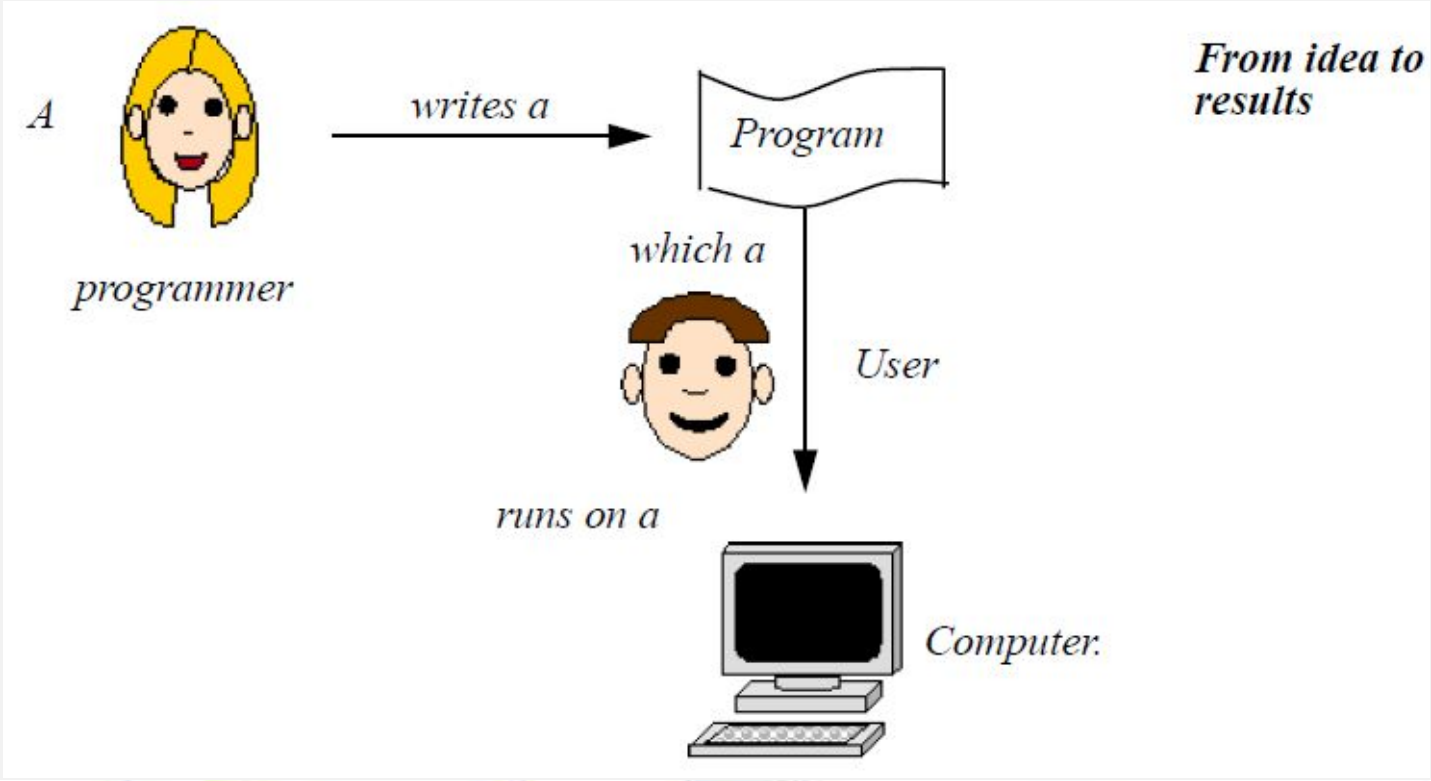
https://mappingthejourney.com/single-post/2017/06/29/episode-3-interview-with-james-gosling-father-of-java/

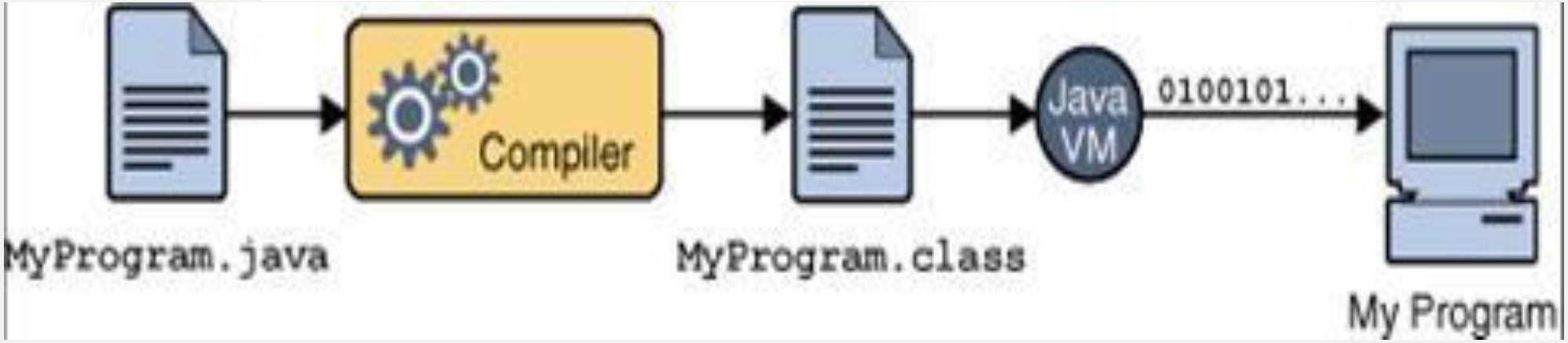
GOALS FOR TODAY'S CLASS

Lecture & Demo in JGRASP

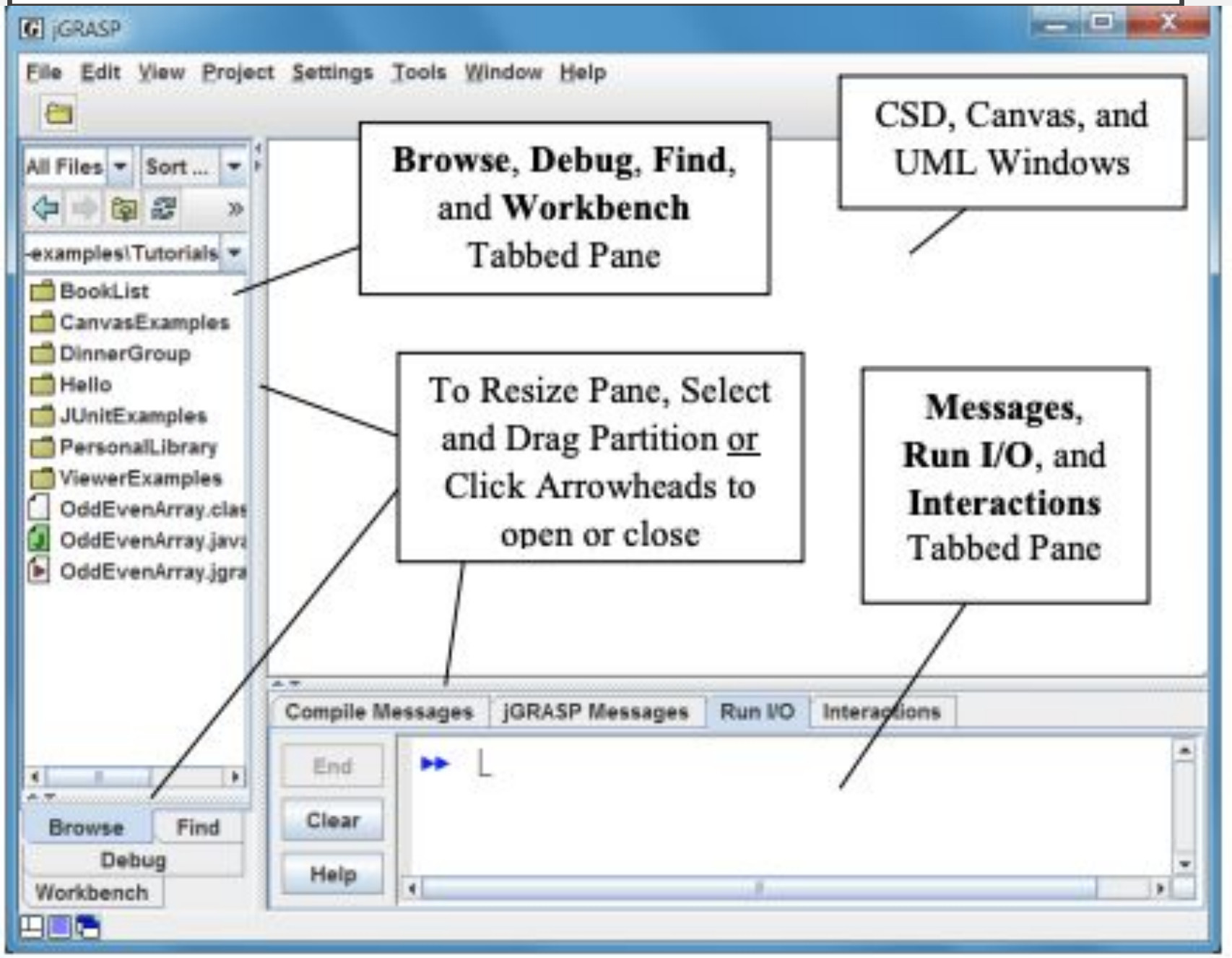
- Introduction to Objects and Classes
- Working with Constructors and Methods
- Using the Debugger
- Using Style Guidelines

REVIEW: HOW DOES PROGRAMMING IN JAVA WORK?





REVIEW: jGRASP DESKTOP



PROBLEM STATEMENT

Create a program for a car. The program must simulate the attributes of the car and its fuel operations.

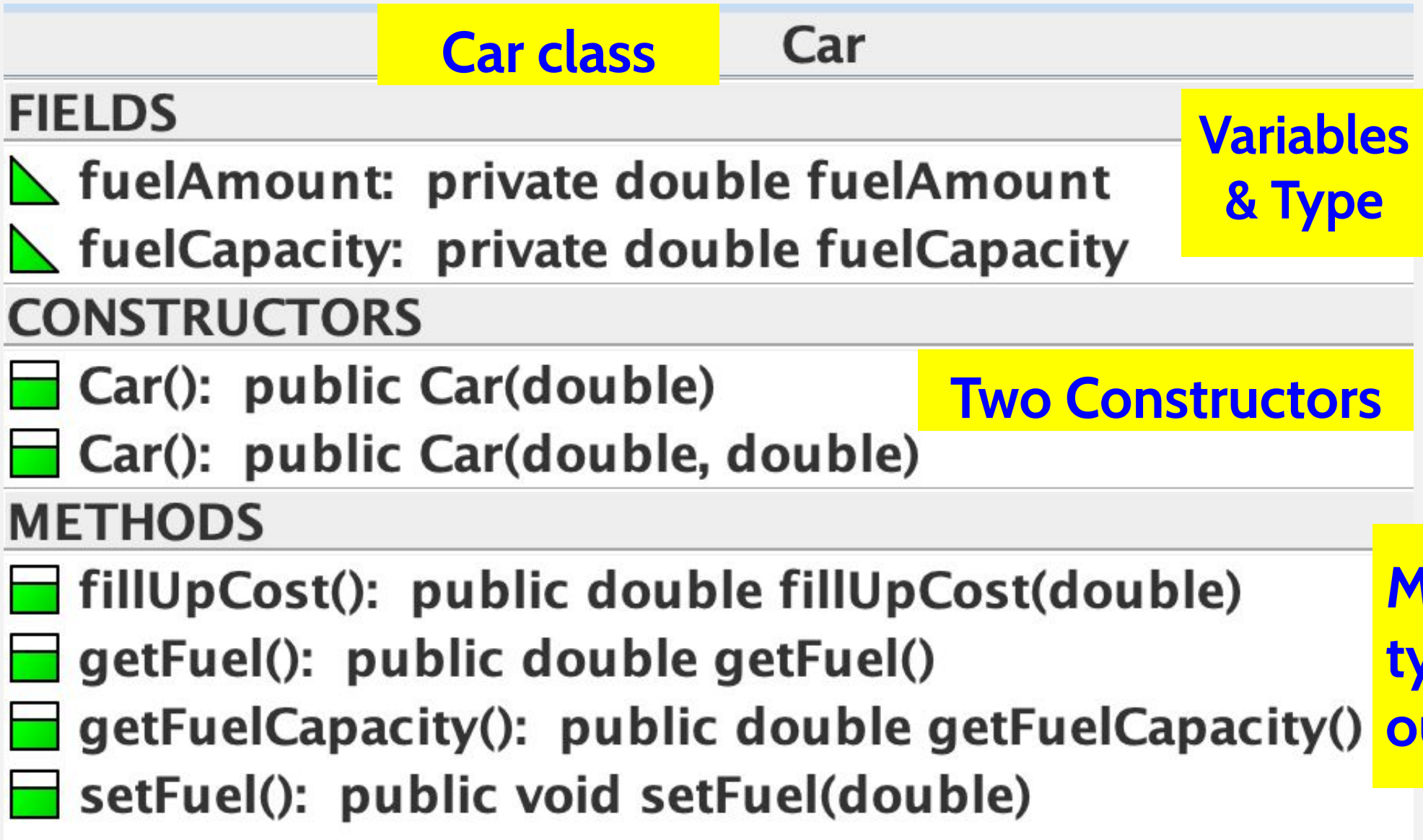
We decide to

In Java, only Class names are capitalized!

1. Create a `Car` class (note capital first letter).
2. Store fuel capacity and amount (in gallons).
3. Give the user a way to update, or set the fuel and capacity.
4. Give the user a way to read, or get the current fuel level and cost for filling up the tank.

Class definitions are meant to model real-world entities, such as a car or an employee. This makes code design easier to understand.

DESIGNING OUR SOLUTION



Note camelCase naming conventions!

DEMO IN JGRASP - Car PROJECT

- Using the Canvas and Interaction pane in jGRASP
- Running Main method
- Calling Constructors and Methods with Parameters and Arguments
- Using the Debugger

TESTING WITH TEST CASES FROM THE MAIN CLASS

Given the cost of the fuel is $2.44.

Test the `fillUpCost` method from `CarMain` for the `yourCar` object.

	Fuel Capacity	Fuel Amount	Expected Cost	
Normal case	10.0	5.0	$12.2	Test fails! ERROR
Edge case	0	0	0	Test passes

REVIEW: WORKING WITH CONSTRUCTORS

```
public class Car{
 // the Car attributes
 private double fuelCapacity;//must be known when creating a car.
 private double fuelAmount;// default value is zero.

 // the Car constructors

 //creates a car with capacity and default fuel level.
 public Car(double fuelCap){
   fuelCapacity = fuelCap;
   fuelAmount = 0.00;
  }

 //creates a car with capacity and fuel level
 public Car(double cap, double amt){
   fuelCapacity = cap;
   fuelAmount = amt;
  }
```

Declare private variable

Initialize variable in constructor

A class may define several constructors with different parameter types.

Create a new Car instance in main method

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

   //use a variable for this instead of hard-coding in lines 26 and
   double currentGasCostPerGallon = 2.44;

   //create an instance of a Car, with fuel capacity, fuel amount p
   Car myCar = new Car(13.5, 5.0);

   //create another Car instance using the default constructor.
   Car yourCar = new Car(10.0);
```

CLICKER QUESTION 1

Select the statement to declare and create a new object of type `Bus` named `bus1`

```
1  public class Bus {
2
3      // Variable to store number of people in bus
4      private int numPeople;
5
6      // Default constructor initializes count to 0
7      public Bus() {
8          numPeople = 0;
9      }
```

A. `Bus bus1;`

B. `Bus bus1 = new Bus();`

C. `Bus = new bus1;`

D. `bus1 = new Bus();`

READY FOR THE ANSWER?

CLICKER QUESTION 1 ANSWER

Select the statement to declare and create a new object of type Bus named bus1

```
1  public class Bus {
2
3      // Variable to store number of people in bus
4      private int numPeople;
5
6      // Default constructor initializes count to 0
7      public Bus() {
8          numPeople = 0;
9      }
```

A. `Bus bus1;` Declares variable bus1 for an object, but does not create an object.

B. `Bus bus1 = new Bus();`

C. `Bus = new bus1;` incorrect syntax

D. `bus1 = new Bus();` incorrect syntax

REVIEW: WORKING WITH METHODS

```
public double getFuelCapacity(){
   return fuelCapacity;
}

public double getFuel(){
   return fuelAmount;
}

public void setFuel(double amt){
   fuelAmount = amt;
}
```

parameter

Methods have return type or void

Mutator methods modify object - changing object's internal data.

Accessor methods access object's data but do not modify internal data.

Notice return types!

Invoke methods on Car object from main

```
//use a variable for this instead of hard-c
double currentGasCostPerGallon = 2.44;

//create an instance of a Car, with fuel ca
Car myCar = new Car(13.5, 5.0);

//create another Car instance using the def
Car yourCar = new Car(10.0);

//using set method to change fuel amount.
yourCar.setFuel(5.00);
```

arguments

CLICKER QUESTION 2

Call the `incrementCount()` method on a `Bus` object named `bus1`

```
    // Variable to store number of people in bus
    private int numPeople;

    // Default constructor initializes count to 0
    public Bus() {
        numPeople = 0;
    }

     // Increments counts by 1
    public void incrementCount() {
        numPeople = numPeople + 1;
    }
}//end class
```

A. `Bus.incrementCount();`
B. `bus1.incrementCount;`
C. `bus1.incrementCount();`
D. `Bus.bus1.incrementCount();`

READY FOR THE ANSWER?

CLICKER QUESTION 2 ANSWER

Call the `incrementCount()` method on a `Bus` object named `bus1`

```
    // Variable to store number of people in bus
    private int numPeople;

    // Default constructor initializes count to 0
    public Bus() {
        numPeople = 0;
    }

     // Increments counts by 1
    public void incrementCount() {
        numPeople = numPeople + 1;
    }
}//end class
```

A. `Bus.incrementCount();`
B. `bus1.incrementCount;`
C. `bus1.incrementCount();`
D. `Bus.bus1.incrementCount();`

Object name comes before the period.
Method being called on that object comes after the period.

CLICKER QUESTION 3

```
public void setNumPeople(int val) {
    numPeople = val;
}

public int getNumPeople() {
    return numPeople;
}

public static void main(String[] args) {
    Bus bus1 = new Bus(20);
    int num;
    //?????
    System.out.println("Number of people on bus = " + num);
}
```

Assume there is a Bus constructor that takes the number of people as an argument.
Which is the missing statement?

A. `num = bus1.setNumPeople(20);`
B. `num = bus1.getNumPeople();`
C. `num = bus1. getNumPeople(20);`
D. `num = 20;`

READY FOR THE ANSWER?

CLICKER QUESTION 3 ANSWER

```
public void setNumPeople(int val) {
    numPeople = val;
}

public int getNumPeople() {
    return numPeople;
}

public static void main(String[] args) {
    Bus bus1 = new Bus(20);
    int num;
    //?????
    System.out.println("Number of people on bus = " + num);
}
```

A. num = bus1.setNumPeople(20);

B. num = bus1.getNumPeople();

C. num = bus1.getNumPeople(20);

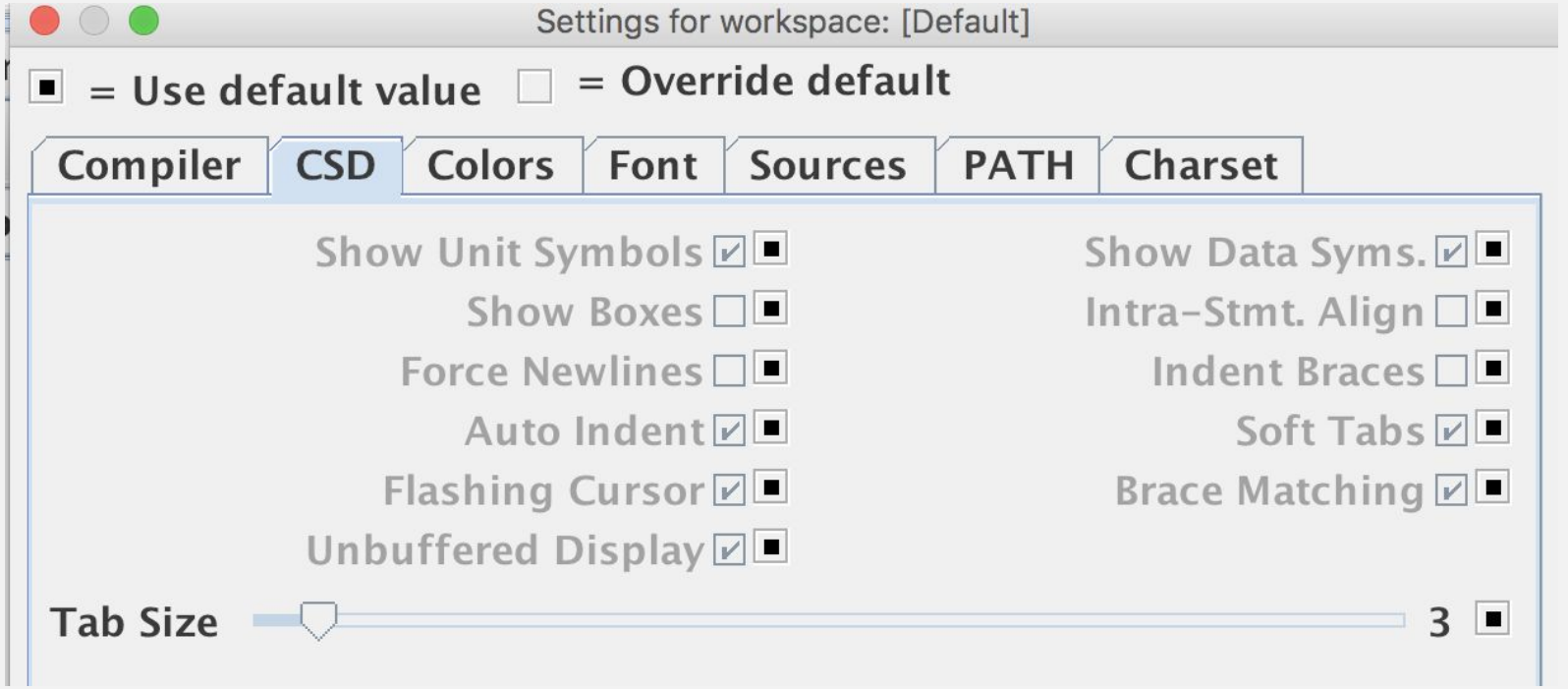
D. num = 20

Which is the missing statement?

RULES & STYLE GUIDES FOR YOUR PROJECT

We'll use Table 2.14.1: Sample style guide from zyBook Chapter 2.14 as our guide for all projects!

In JGrasp "`Settings->Font->CSD`" tab check that `AutoIndent` and `SoftTabs` have a tick mark and Tab size is set to 3 spaces.



GOOD CODE FORMATTING THINK-PAIR-SHARE

```
1  public class firstprogram {
2  public static void main(String[] args){
3  String x = "Hello and welcome to all ";
4  int y = 450;
5  String z = " students in CS121!";
6  System.out.print(x);
7  System.out.print(y);
8  System.out.print(z);
9  }
10 }
```

Version 1

In jGRASP:
Java "keywords" in purple.
String literals in green.
Comments in orange.

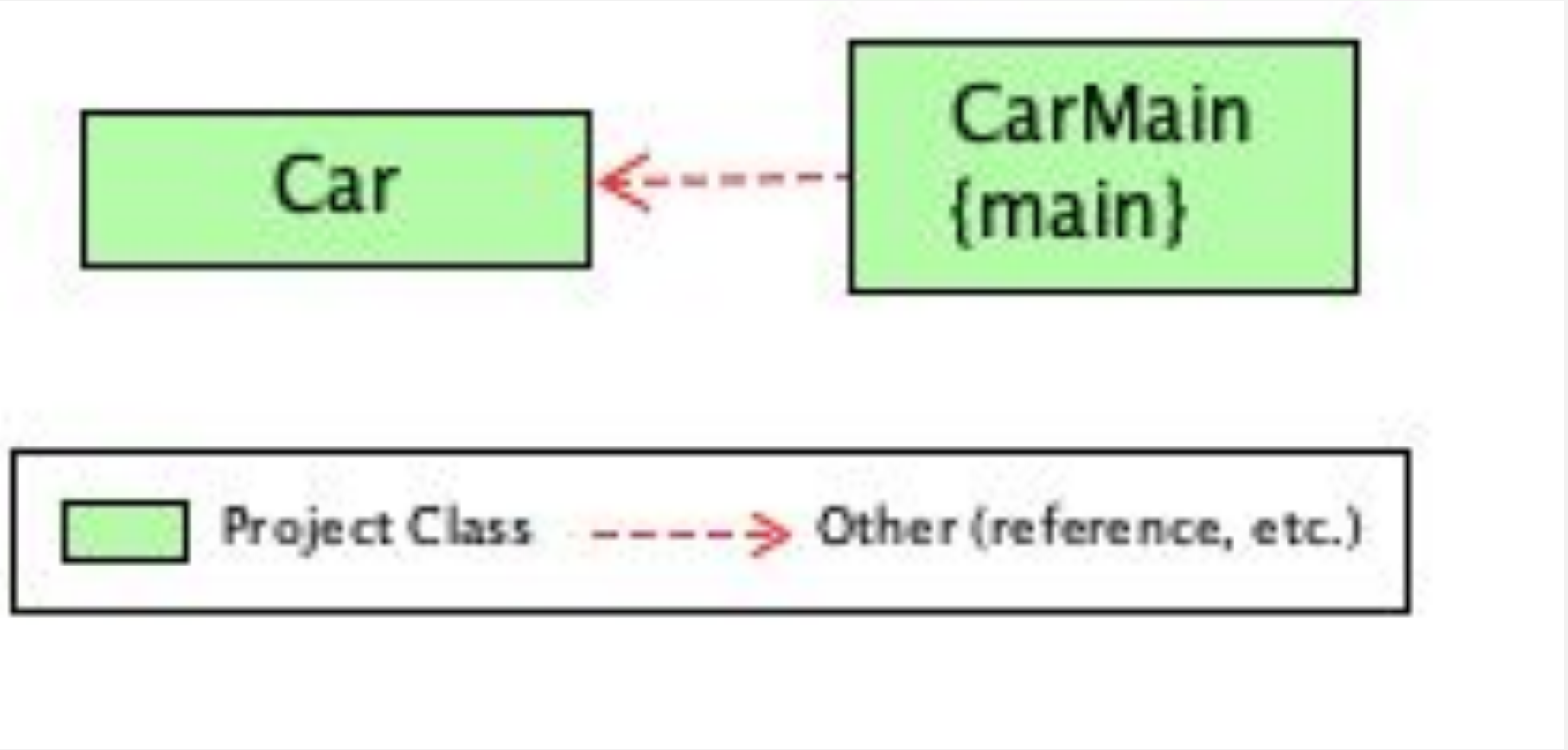
How many changes can you spot?

```
1  /* This class implements a simple program that
2     prints a greeting message to the console.
3  */
4  public class FirstProgram {
5     public static void main(String[] args){
6
7        String str1 = "Hello and welcome to all ";
8        int num = 450;
9        String str2 = " students in CS121!";
10
11       System.out.print(str1);
12       System.out.print(num);
13       System.out.print(str2);
14    }
15 }
```

Version 2

REVIEW: USING 2 CLASSES

Why two classes?
`Car` encapsulates data about each specific car.
`CarMain` deals with the various cars and prints info.



The dotted line means `CarMain` uses an instance of Car.

REVIEW: USING 2 CLASSES

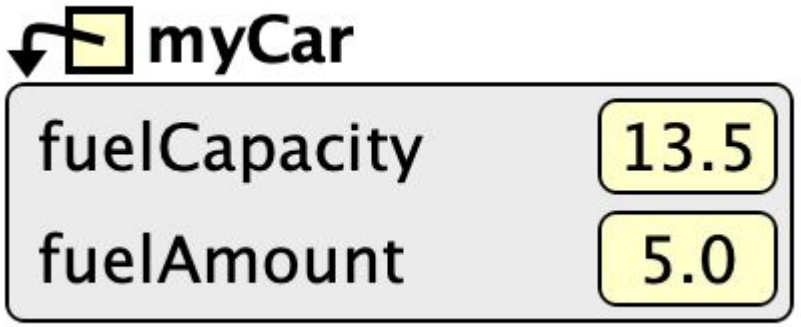
These diagrams show the design of the 2 classes.

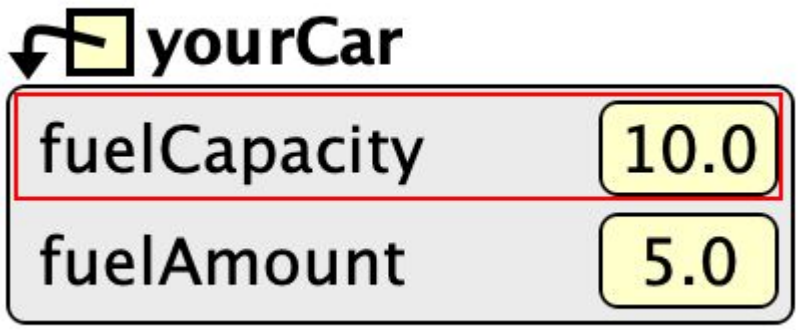
Car
FIELDS
fuelAmount: private double fuelAmount
fuelCapacity: private double fuelCapacity
CONSTRUCTORS
Car(): public Car(double)
Car(): public Car(double, double)
METHODS
fillUpCost(): public double fillUpCost(double)
getFuel(): public double getFuel()
getFuelCapacity(): public double getFuelCapacity()
setFuel(): public void setFuel(double)

CarMain
FIELDS
CONSTRUCTORS
CarMain(): public CarMain()
METHODS
main(): public static void main(java.lang.String[])

REVIEW: USING 2 CLASSES

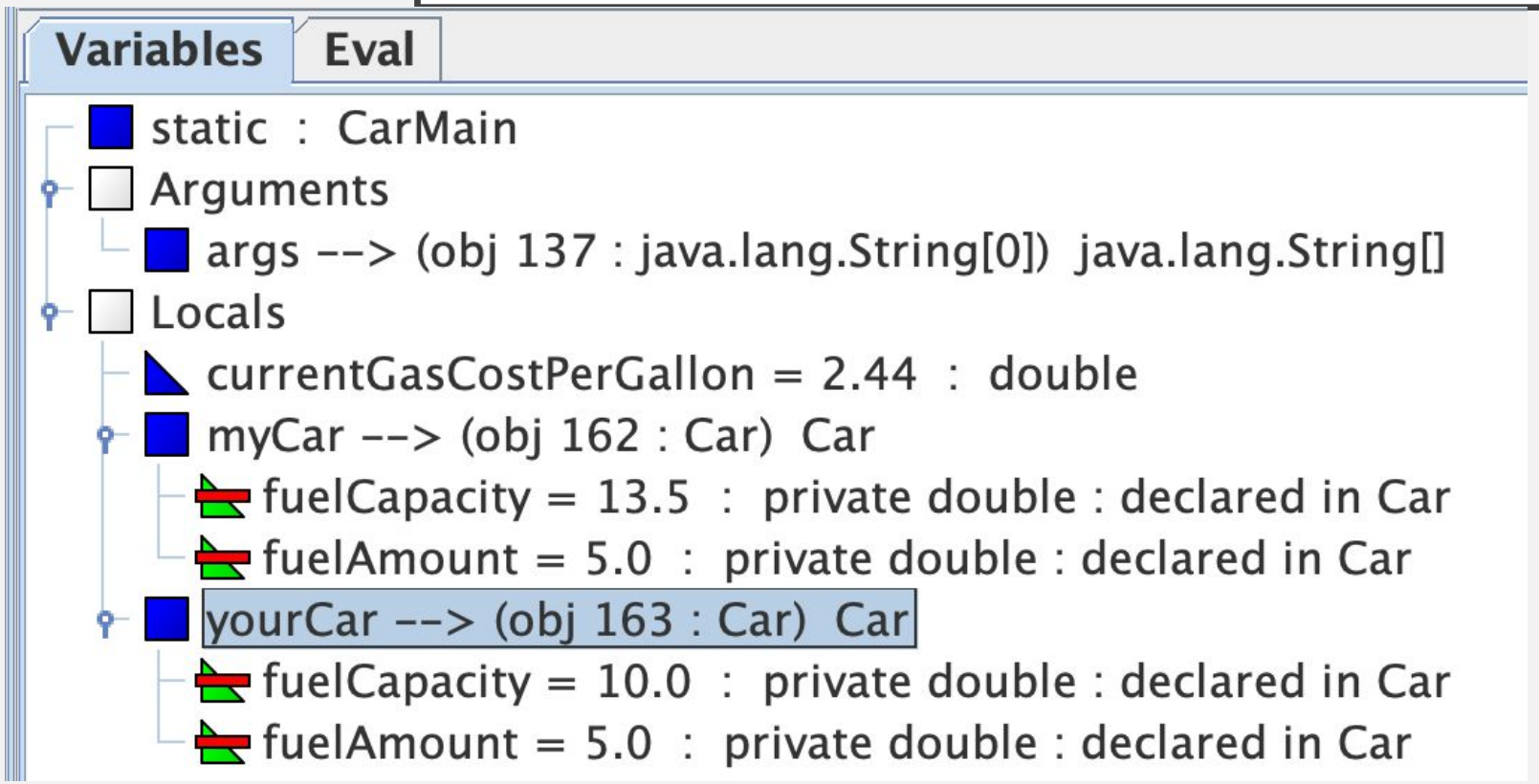
This diagram shows the objects that are created when the `CarMain` program is run.





The program could deal with any number of cars. Data for each specific car is contained in a `Car` object. This keeps the code modular and easier to work with. `CarMain` manages all `Car` objects.

REVIEW: USING 2 CLASSES



Shows the state of the variables on completion of the program.

```
myCar capacity: 13.5
yourCar fuel: 5.0
yourCar capacity: 10.0
myCar fill-up cost: 20.74
yourCar fill-up cost: 20.74
```

Shows the output when the `CarMain` program is run.

Week 2 TODO List:

- Register your iClicker in Moodle.
- Complete zyBook chapter 2 exercises.
- Join us on Piazza - sign-up in Moodle.
- Attend Lab 2 on Friday.
- Start Project 1 early! Upload often in codePost.