

```
class Coord {
    public int row, col;
    public Coord(int rowVal, int colVal) {
        this.row = rowVal;
        this.col = colVal;
    }
}
class Car {
    public String color;
    public Coord location;
    public Car(String colorVal, Coord locVal) {
        this.color = colorVal;
        this.location = locVal;
    }
}
class Q1 {
    public static void g(Car c1, Car c2) {
        c2 = c1;
        c2.color = "blue";
    }
    public static String question () {
        Car redCar = new Car("red", new Coord(5, 6));
        Car greenCar = new Car("green", new Coord(7, 8));
        g(redCar, greenCar);
        return redCar.color + ", " + greenCar.color;
    }
    public static void main(String[] args) {
        System.out.println(question());
    }
}
```

Fields are associated with classes and objects. In the code above, `row`, `col`, `color`, and `location` are all **fields**. Also called **instance variables**, but we will use “field” to avoid confusion with other kinds of variables.

Stack

Method calls and variables

returns: nothing (void)	
returns:	
Q1. main(@Z)	
args	@Z
returns: nothing (void)	

Heap

Objects and their fields, arrays

@Z	[] An empty array for args, a detail not used in this example

Variables are associated with methods. In the code above, `c1`, `c2`, `redCar`, `greenCar`, `rowVal`, `colVal`, `colorVal`, and `locVal` are **variables**. Variables in the method signature (for example `c1` and `c2`) are also called **parameters**.

```
public class Q2 {
    public static void f(Coord c) {
        Car car = new Car("blue", c);
        car.location.row = 10;
        car.location.col = 9;
    }
    public static int question() {
        Coord unit = new Coord(1, 1);
        Car blackCar = new Car("black", unit);
        f(unit);
        return blackCar.location.row;
    }
    public static void main(String[] args) {
        System.out.println(question());
    }
}
```

returns:	
returns:	
Q1. main(@Z)	
args	@Z
returns: nothing (void)	

@Z	[] An empty array for args, a detail not used in this example

```
interface StringList {
```

```
    // We will fill this in together
```

```
}
```

```
class StringListIdea1 implements StringList {
```

```
    // How will it store the data?
```

```
    // How will it implement the methods?
```

```
}
```