How to swap or exchange objects in Java?

How to swap objects in Java?
Let's say we have a class called "Car" with some attributes. And we create two objects of Car, say car1 and car2, how to exchange the data of car1 and car2?

A Simple Solution is to swap members. For example, if the class Car has only one integer attribute say "no" (car number), we can swap cars by simply swapping the members of two cars.

```
// A Java program to demonstrate that we can swap two
// objects be swapping members.

// A car with number class Car
class Car
{
    int no;
    Car(int no) { this.no = no; }
}

// A class that uses Car objects
class Main
{
    // To swap c1 and c2
    public static void swap(Car c1, Car c2)
    {
        int temp = c1.no;
        c1.no = c2.no;
        c2.no = temp;
    }

    // Driver method
    public static void main(String[] args)
    {
        Car c1 = new Car(1);
        Car c2 = new Car(2);
        swap(c1, c2);
        System.out.println("c1.no = " + c1.no);
        System.out.println("c2.no = " + c2.no);
    }
}
```

Output:

```
c1.no = 2
c2.no = 1
```

What if we don't know members of Car?
The above solution worked as we knew that there is one member "no" in Car. What if we don't know members of Car or the member list is too big. T*his is a very common situation as a class that uses some other class may not access members of other class*. Does below solution work?

```
// A Java program to demonstrate that simply swapping
// object references doesn't work

// A car with number and name
class Car
{
```

```
    int model, no;

    // Constructor
    Car(int model, int no)
    {
        this.model = model;
        this.no = no;
    }

    // Utility method to print Car
    void print()
    {
        System.out.println("no = " + no +
                           ", model = " + model);
    }
}

// A class that uses Car
class Main
{
    // swap() doesn't swap c1 and c2
    public static void swap(Car c1, Car c2)
    {
        Car temp = c1;
        c1 = c2;
        c2 = temp;
    }

    // Driver method
    public static void main(String[] args)
    {
        Car c1 = new Car(101, 1);
        Car c2 = new Car(202, 2);
        swap(c1, c2);
        c1.print();
        c2.print();
    }
}
```

Output:

```
no = 1, model = 101
no = 2, model = 202
```

As we can see from above output, the objects are not swapped. We have seen that parameters are passed by value in Java. So when we pass c1 and c2 to swap(), the function swap() creates a copy of these references.

Solution is to use Wrapper Class If we create a wrapper class that contains references of Car, we can swap cars by swapping references of wrapper class.

```
// A Java program to demonstrate that we can use wrapper
// classes to swap to objects

// A car with model and no.
class Car
{
    int model, no;

    // Constructor
```

```
    Car(int model, int no)
    {
        this.model = model;
        this.no = no;
    }

    // Utility method to print object details
    void print()
    {
        System.out.println("no = " + no +
                           ", model = " + model);
    }
}

// A Wrapper over class that is used for swapping
class CarWrapper
{
   Car c;

   // Constructor
   CarWrapper(Car c)   {this.c = c;}
}

// A Class that use Car and swaps objects of Car
// using CarWrapper
class Main
{
    // This method swaps car objects in wrappers
    // cw1 and cw2
    public static void swap(CarWrapper cw1,
                            CarWrapper cw2)
    {
        Car temp = cw1.c;
        cw1.c = cw2.c;
        cw2.c = temp;
    }

    // Driver method
    public static void main(String[] args)
    {
        Car c1 = new Car(101, 1);
        Car c2 = new Car(202, 2);
        CarWrapper cw1 = new CarWrapper(c1);
        CarWrapper cw2 = new CarWrapper(c2);
        swap(cw1, cw2);
        cw1.c.print();
        cw2.c.print();
    }
}
```

Output:

```
no = 2, model = 202
no = 1, model = 101
```

So a wrapper class solution works even if the user class doesn't have access to members of the class whose objects are to be swapped.