Extra

Static Properties

Static

Definition of Static in Java

If a method or property is marked as static, it means **there is exactly one** copy of the method, or one copy of the data member shared across all objects of the class.

One way to think about this, is that the static member is a unique property of the “blueprint” that is the same for all objects created from that blueprint.

The non-static methods and properties we have defined so far are called instance properties or instance methods.

Static Members: Declaration

Static members and methods are declared by adding the keyword static.

```
public class Car {
        public static String carBrand = "Ford";

        public static void honkHorn() {
                System.out.println("beeep?");
        }
...
}
```

Static: Calling

Assuming we have the static member declarations from the previous slide, this is how you call them from a different class. Note that we use the class name (Car) as opposed to the name of an instance of a car (thisCar).

```
public class Garage {

	public static void main(String args[]) {

		System.out.println(Car.carBrand); // Correct way to refer to a static member.
		Car.honkHorn(); // Correct call to a static method.

		Car thisCar = new Car("Red", 2);

		System.out.println(thisCar.brand); // Not a valid way to call a static member.
		thisCar.honkHorn() // Not a valid way to call a static method.

	}}
```

Static: Assignment

Static data members can be reassigned to new values. Note that this will change the value for all instances of the Car object.

```
public class Garage {

        public static void main(String args[]) {

                Car.carBrand = “GM”;
        }
}
```

Static: Constants

Constants are variables that cannot change. The closest thing to a constant in Java is declaring a data member with **static final**.

```
public class Car {
        public static final String carBrand = "Ford";
...
}
```

Attempts to change the value of this data member will result in an error. This, for example is invalid:

```
public class CarDealership {

        public static void main(String args[]) {

                Car.carBrand = "GM";
}}
```

Static: Rules

There are some rules to observe when using static methods or data members:

- **Static** variables can be accessed by **Instance** methods.
- **Static** methods can be accessed by **Instance** methods.

The opposite of the above is not true:

- **Static** methods cannot access **Instance** properties.
- **Static** methods cannot call **Instance** methods.

Static: Rules

```
String someInstanceVariable;

public static void someStaticMethod() {
        System.out.printlnString (someInstanceVariable);
        someInstanceMethod();
}

public void someInstanceMethod() {

}
```

This is an instance (non-static data member)

We are inside a static method, but we are referencing an instance member, which is not allowed

We are inside a static method, but we are calling an instance method, which is not allowed.

You have encountered this issue before - recall that any method directly called by public static void main had to also be a static.