The Keyword: this

CSCI 111
There are two types of methods:

- **Instance Methods (non-static).** Need to be called on an instance of a class (on an object).

- **Static Methods.** Do not need to be called on an instance of a class.
public class Person
{
    private String name;
    public Person(String inName)
    {
        name = inName;
    }
    public String getName()
    {
        return name;
    }
}

public class Driver
{
    public static void main(String[] args)
    {
    }
}
public class Person
{
    private String name;
    public Person(String inName)
    {
        name = inName;
    }
    public String getName()
    {
        return name;
    }
}

public class Driver
{
    public static void main(String[] args)
    {
        Person p1 = new Person("Sally");
    }
}
public class Person {
    private String name;
    public Person(String inName) {
        name = inName;
    }
    public String getName() {
        return name;
    }
}

public class Driver {
    public static void main(String[] args) {
        Person p1 = new Person("Sally");
        Person p2 = new Person("Joe");
    }
}
public class Person {
    private String name;
    public Person(String inName) {
        name = inName;
    }
    public String getName() {
        return name;
    }
}

public class Driver {
    public static void main(String[] args) {
        Person p1 = new Person("Sally");
        Person p2 = new Person("Joe");
        Person p3 = new Person("Kelly");
    }
}
Example

```java
public class Person
{
    private String name;
    public Person(String inName)
    {
        name = inName;
    }
    public String getName()
    {
        return name;
    }
}
```

```java
name = "Sally"
public String getName()
{
    return name;
}
```

```java
name = "Joe"
public String getName()
{
    return name;
}
```

```java
name = "Kelly"
public String getName()
{
    return name;
}
```

```java
public class Driver
{
    public static void main(String[] args)
    {
        Person p1 = new Person("Sally");
        Person p2 = new Person("Joe");
        Person p3 = new Person("Kelly");
        //call getName();
```
Example

```
public class Person {
    private String name;

    public Person(String inName) {
        name = inName;
    }

    public String getName() {
        return name;
    }
}
```

```
public class Driver {
    public static void main(String[] args) {
        Person p1 = new Person("Sally");
        Person p2 = new Person("Joe");
        Person p3 = new Person("Kelly");
        // call getName();
    }
}
```

This exists in the computer and is executable (code we can run).

This is NOT executable (it is not code that we can run).
public class Person {
    private String name;
    public Person(String inName) {
        name = inName;
    }
    public String getName() {
        return name;
    }
}

public class Driver {
    public static void main(String[] args) {
        Person p1 = new Person("Sally");
        Person p2 = new Person("Joe");
        Person p3 = new Person("Kelly");
        System.out.println(p2.getName());
    }
}
public class Person
{
    private String name;
    public Person(String inName)
    {
        name = inName;
    }
    public String getName()
    {
        return name;
    }
}

public class Driver
{
    public static void main(String[] args)
    {
        Person p1 = new Person("Sally");
        Person p2 = new Person("Joe");
        Person p3 = new Person("Kelly");
        System.out.println(p2.getName());
    }
}

LESSON: If the method is not static, it MUST be executed INSIDE an object (a specific instance of the class).
“this” refers to the object (instance of the class) we are currently inside.
“this” refers to the object (instance of the class) we are currently inside.

Think of “this” as being a variable (hidden instance variable) of dataType equal to the class.
The value of the “this” variable changes depending on which object it is inside, but it is the address of that object in memory.
How do we know we are inside a specific object?

Because we are in a method that is not static.

define

```java
name = "Sally"
age = 19

public Person returnRef()
{
    Person a = this;
    return a;
}
```
How do we tell which object it is?

We look to see where the method was called from and what object it was called on.

```java
public class Person {
    String name;
    int age;

    public Person(String n, int a) {
        name = n;
        age = a;
    }

    public Person returnRef() {
        Person a = this;
        return a;
    }
}

public class Main {
    public static void main(String[] args) {
        Person p1 = new Person("Sally", 19);
        Person p2 = p1.returnRef();
    }
}
```
What if we call “this” from a static method?

You will get the error: “non-static variable this cannot be referenced from a static context”.

```java
name = "Sally"
age = 19

public static Person returnRef()
{
    Person a = this;
    return a;
}
```