The Keyword: `this`

CSCI 111

Method Calls

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.

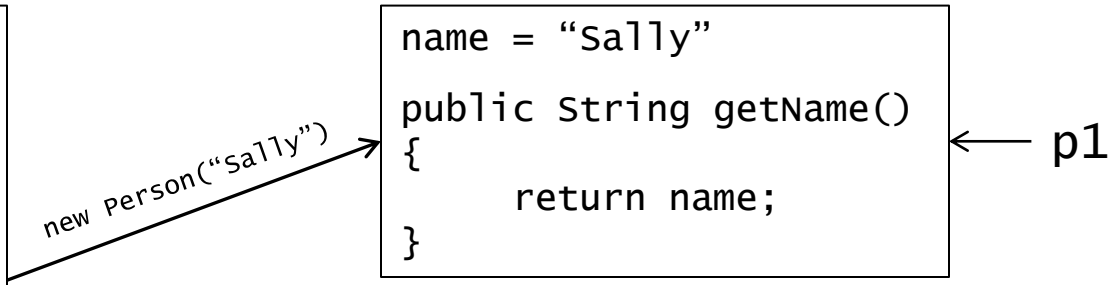
Example

```
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)
     {




     }
}
```

Example

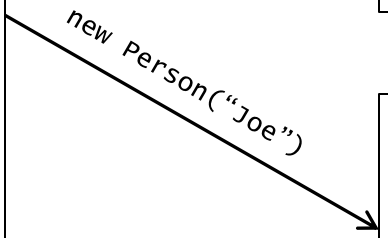
```
public class Person
{
     private String name;

     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");




     }
}
```

Example

```
public class Person
{
     private String name;

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

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




```
name = "Sally"

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

p1

```
name = "Joe"

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

p2

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




     }
}
```

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");


     }
}
```

new Person("Kelly")

```
name = "Sally"

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

p1

```
name = "Joe"

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

p2

```
name = "Kelly"

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

p3

Example

```
public class Person
{
     private String name;

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

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

```
name = “Sally”

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

p1

```
name = “Joe”

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

p2

```
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();
     }
}
```

```
name = “Kelly”

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

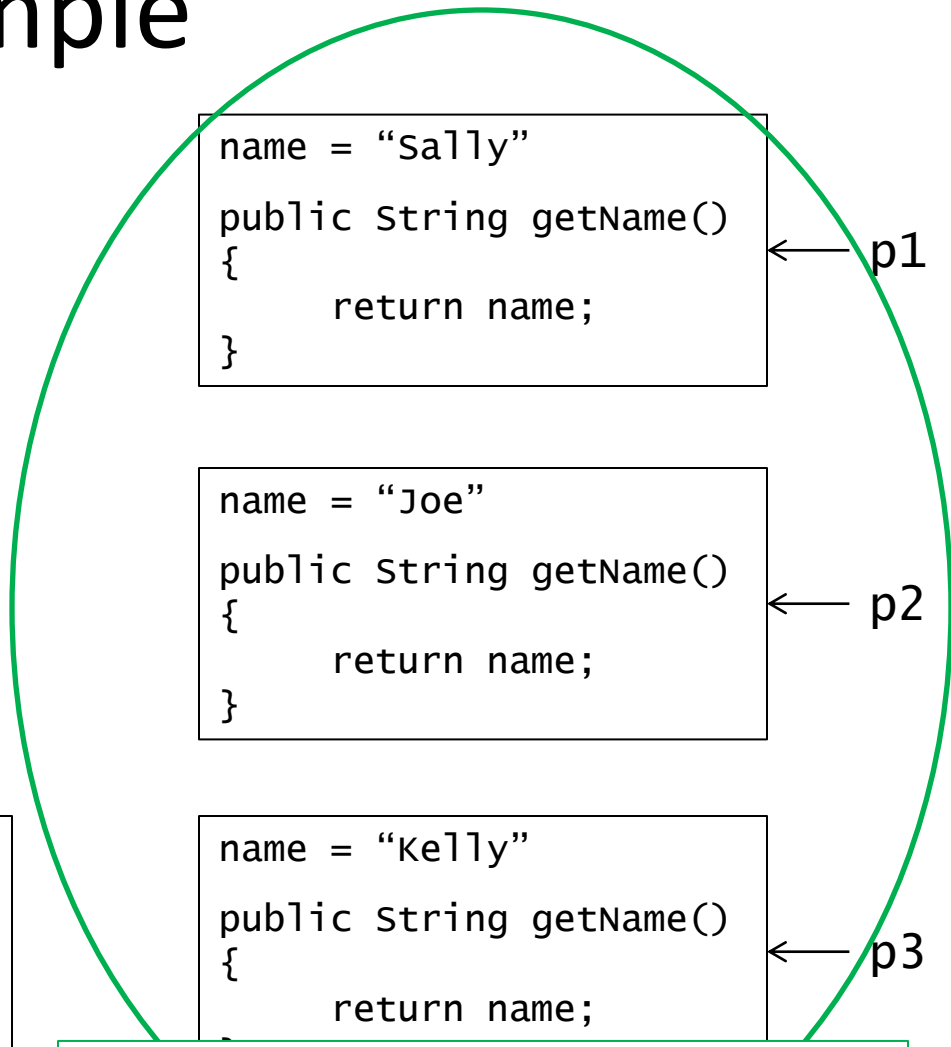
p3

Example

```
public class Person
{
      private String name;

      publ
      {

      }

      publ
      {
            return name;
      }
}
```

This is NOT executable (it is not code that we can run).

```
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).

Example

```
public class Person
{
      private String name;

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

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

```
name = "Sally"

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

← p1

```
name = "Joe"

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

← p2

```
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());
      }
}
```

```
name = "Kelly"

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

← p3

Example

```
public class Person
{
      private String name;

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

      public St
      {
            retu
      }
}
```

```
name = "Sally"

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

p1

me()

p2

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

p3

LESSON: If the method is not static, it MUST be executed INSIDE an object (a specific instance of the class).

`this`

"`this`" refers to the object (instance of the class) we are currently inside.

this

"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.

```
public class Person
{
      private String name;

      public void method()
      {
            //code
            a = this;
            //code
      }
}
```

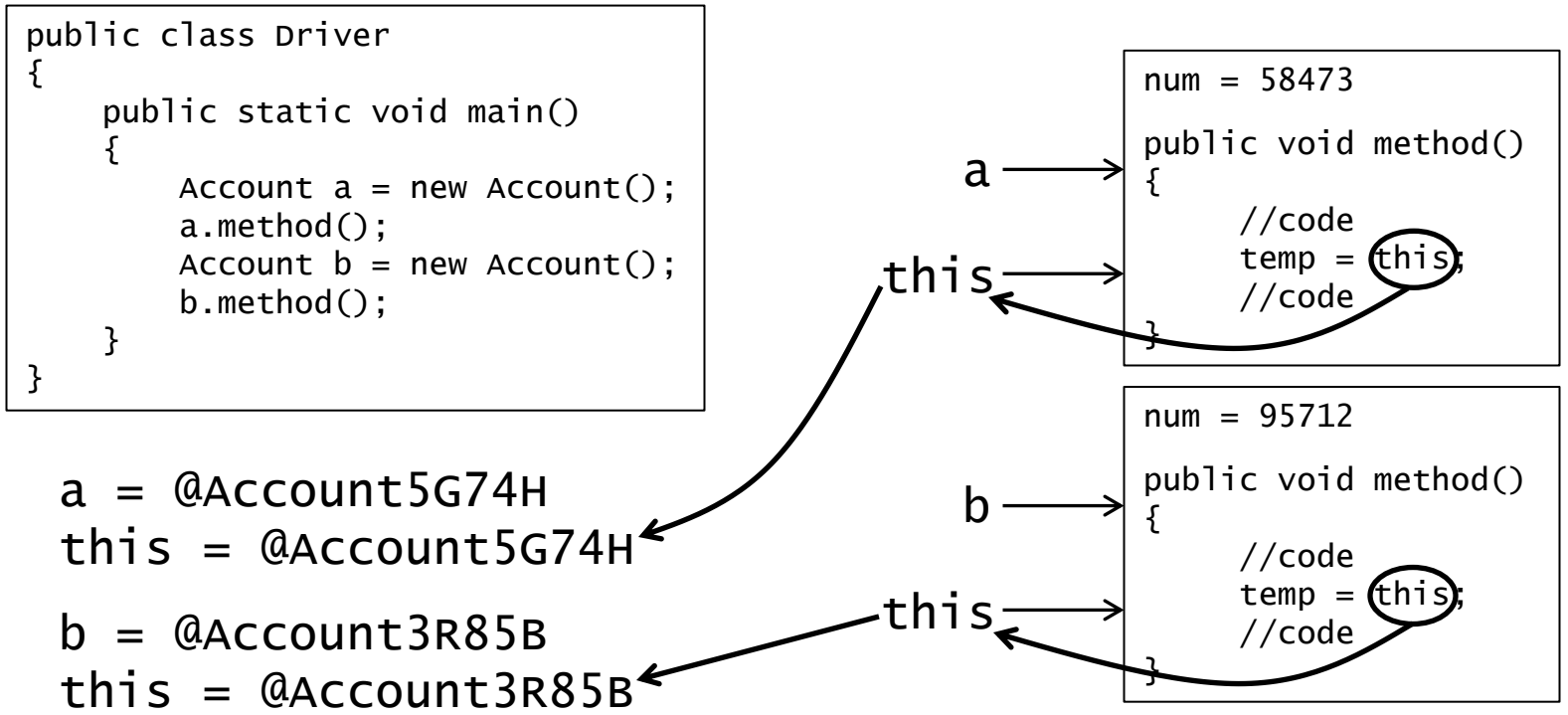
dataType Person

```
public class Account
{
      private int num;

      public void method()
      {
            //code
            a = this;
            //code
      }
}
```

dataType Account

this

The value of the "this" variable changes depending on which object it is inside, but it is the address of that object in memory.



this

How do we know we are inside a specific object?

Because we are in a method that is not static.

```
name = “Sally”
age = 19

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

this

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.

```
public void methodA()
{
      Person p1 = new Person("Sally", 19);
      Person p2 = p1.returnRef();
}
```

```
name = "Sally"
age = 19

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

this

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".

```
name = "Sally"
age = 19

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