



**Computer Science Virtual Learning**

# **HS Computer Science A**

**May 21st, 2020**



Lesson: **The Equals Method**

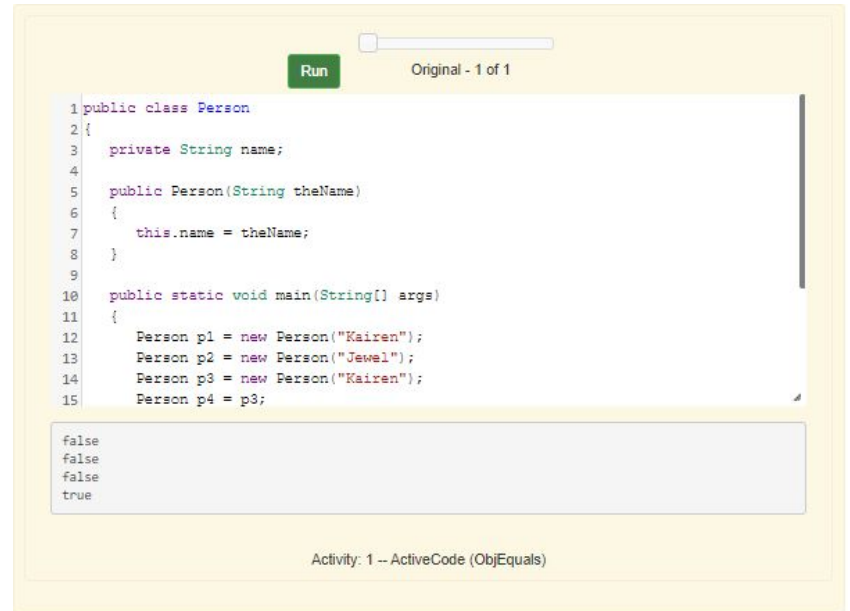
**Objective/Learning Target:**

Understanding what the Equals Method is

# The Equals Method

If a parent class isn't specified using the extends keyword, the class will inherit from the Object class. What does a class inherit from the Object class? One of the important things that gets inherited is the equals(Object obj) method. This method is used to test if the current object and the passed object called obj are equal.

The equals method that is inherited from the Object class only returns true if the two objects references refer to the same object.

A screenshot of a Java IDE window titled "Original - 1 of 1". It shows a code editor with the following Java code:

```
1 public class Person
2 {
3     private String name;
4
5     public Person(String theName)
6     {
7         this.name = theName;
8     }
9
10    public static void main(String[] args)
11    {
12        Person p1 = new Person("Kairen");
13        Person p2 = new Person("Jewel");
14        Person p3 = new Person("Kairen");
15        Person p4 = p3;
```

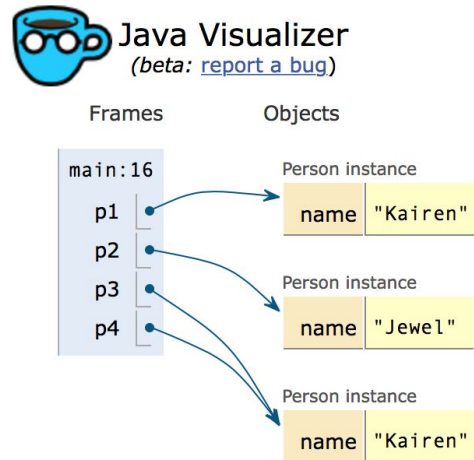
Below the code editor is an output window showing the results of the equals method calls:

```
false
false
false
true
```

At the bottom of the IDE window, it says "Activity: 1 -- ActiveCode (ObjEquals)".

# The Equals Method

The equals method inherited from the Object class only returns true when the two references point to the same object as shown in the code above and figure 1 below.



# String Overrides Equals

If you want to change how the inherited `equals` method works you can **override** it so that the new method is called instead of the inherited one. The `String` class **overrides** the inherited `equals` method to return true when the two objects have the same characters in the same order as shown in the code below.

A screenshot of an IDE window titled "Original - 1 of 1" with a "Run" button. The code defines a class `StringTest` with a `main` method. It creates three strings: `s1 = "hi"`, `s2 = "Hi"`, and `s3 = new String("hi")`. It then prints the results of `s1.equals(s2)`, `s2.equals(s3)`, and `s1.equals(s3)`. The output shows `false`, `false`, and `true` respectively.

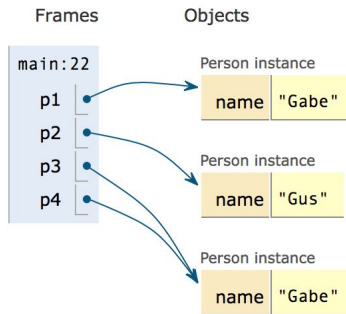
```
1 public class StringTest {
2 {
3     public static void main(String[] args)
4     {
5         String s1 = "hi";
6         String s2 = "Hi";
7         String s3 = new String("hi");
8         System.out.println(s1.equals(s2));
9         System.out.println(s2.equals(s3));
10        System.out.println(s1.equals(s3));
11    }
12 }
13
```

false  
false  
true

Activity: 2 -- ActiveCode (StringTest1)

# Overriding the Inherited Equals Method

A class can override the inherited `equals` method by providing a method with the same method signature (method name and parameter list) and return type. The provided method will be called instead of the inherited one, which is why we say that the new method **overrides** the inherited method. The `Person` class below **overrides** the inherited `equals` method.



Run Original - 1 of 1

```

1 public class Person
2 {
3     private String name;
4
5     public Person(String theName)
6     {
7         this.name = theName;
8     }
9
10    public boolean equals(Object other)
11    {
12        Person otherPerson = (Person) other;
13        return this.name.equals(otherPerson.name);
14    }
15

```

```

false
false
true
true

```

# Overriding vs Overloading

**Overriding** an inherited method means providing a method in a child class with the same method signature (method name and parameter type list) and return type as a method in the parent class. The method in the child class will be called *instead of* the method in the parent class. In the following example the `MeanGreeter` inherits the `greet` method from `Greeter`, but then overrides it.

To override an inherited method, the method in the child class must have the same name, parameter list, and return type (or a subclass of the return type) as the parent method.

A screenshot of a Java IDE interface. At the top right, there is a green "Run" button and a progress indicator labeled "Original - 1 of 1". The main area contains a code editor with the following Java code:

```
1 public class Greeter
2 {
3     public String greet()
4     {
5         return "Hi";
6     }
7
8     public static void main(String[] args)
9     {
10        Greeter g1 = new Greeter();
11        System.out.println(g1.greet());
12        Greeter g2 = new MeanGreeter();
13        System.out.println(g2.greet());
14    }
15 }
```

Below the code editor is a console window showing the output:

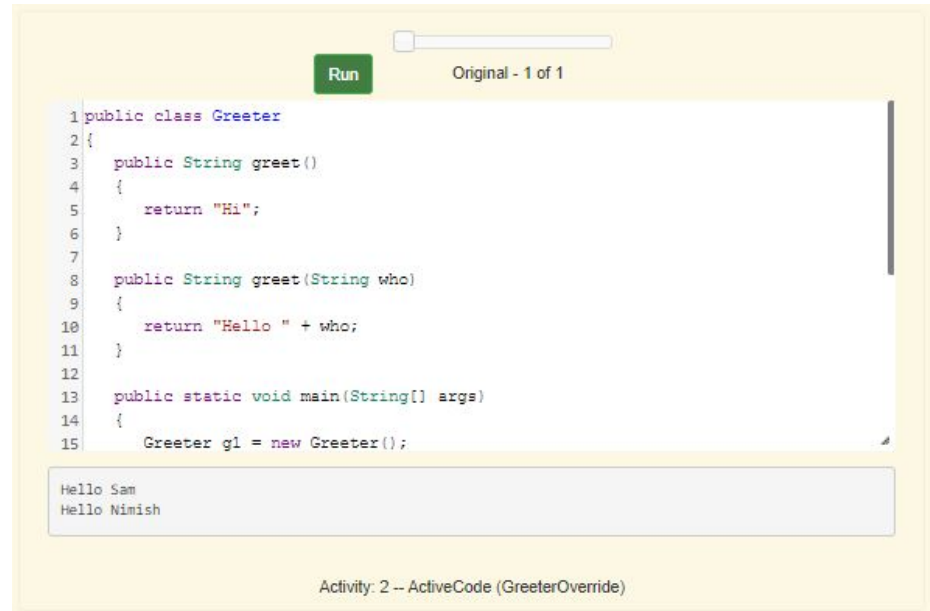
```
Hi
Go Away
```

At the bottom of the IDE, it says "Activity: 1 -- ActiveCode (GreeterEx)".

# Overriding vs Overloading

**Overloading** a method is when several methods have the same name but the parameter types, order, or number are different. In the example below the `greet(String who)` method overloads the `greet()` method of `Greeter`. Notice that `MeanGreeter` inherits this method and it isn't overridden.

To overload a method the method must have the same name, but the parameter list must be different in some way. It can have a different number of parameters, different types of parameters, and/or a different order for the parameter types. The return type can also be different.

A screenshot of a Java IDE. At the top right, there is a green "Run" button and a progress indicator showing "Original - 1 of 1". The main area contains a code editor with the following Java code:

```
1 public class Greeter
2 {
3     public String greet()
4     {
5         return "Hi";
6     }
7
8     public String greet(String who)
9     {
10        return "Hello " + who;
11    }
12
13    public static void main(String[] args)
14    {
15        Greeter g1 = new Greeter();
```

Below the code editor is a text area containing the output:

```
Hello Sam
Hello Nimish
```

At the bottom right, the text "Activity: 2 -- ActiveCode (GreeterOverride)" is visible.





## Check Your Understanding

Which of the following declarations in Student would correctly override the getFood method in Person?

```
public class Person
{
    private String name = null;

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

    public String getFood()
    {
        return "Hamburger";
    }
}
```

- A. public void getFood()
- B. public String getFood(int quantity)
- C. public String getFood()

Which of the following declarations in Person would correctly overload the getFood method in Person?

```
public class Person
{
    private String name = null;

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

    public String getFood()
    {
        return "Hamburger";
    }
}
```

- A. public void getFood()
- B. public String getFood(int quantity)
- C. public String getFood()



## For More Resources and to Check Answers

Go to: <https://runestone.academy/runestone/books/published/apcsareview/OOBasics/ooOverrideInherited.html>

<https://runestone.academy/runestone/books/published/apcsareview/OOBasics/ooOverrideVsOverload.html>