Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Today's Lecture

- toString
- equals

toString

- First we will cover the toString method…

Object toString Implementation

Object.toString() Method

toString() – Returns a string representation of the object.

- There is a default implementation of toString() defined on the Object class.
- **The default implementation will return a string that contains the object type concatenated with an integer.**

Object

```
String toString() {
        Returns string containing class name
        and integer
}
```

Object toString Implementation

```
public class Employee {
    public String firstName;
    public String lastName;
}

public class Driver {
    public static void main(String[] args) {
        Employee e1 = new Employee();
        e1.firstName = "Arthur";
        e1.lastName = "Hoskey";
        System.out.println(e1.toString());
    }
}
```

You do not have to override toString. If there is no toString override it will use the base class methods toString (Object in this case)

Prints the class name and a number (uses the base class toString method which is Object in this case)

Prints something like:

```
csc211.hoskey.compare.tostring.Employee@4fee225
```

Using Default toString Implementation

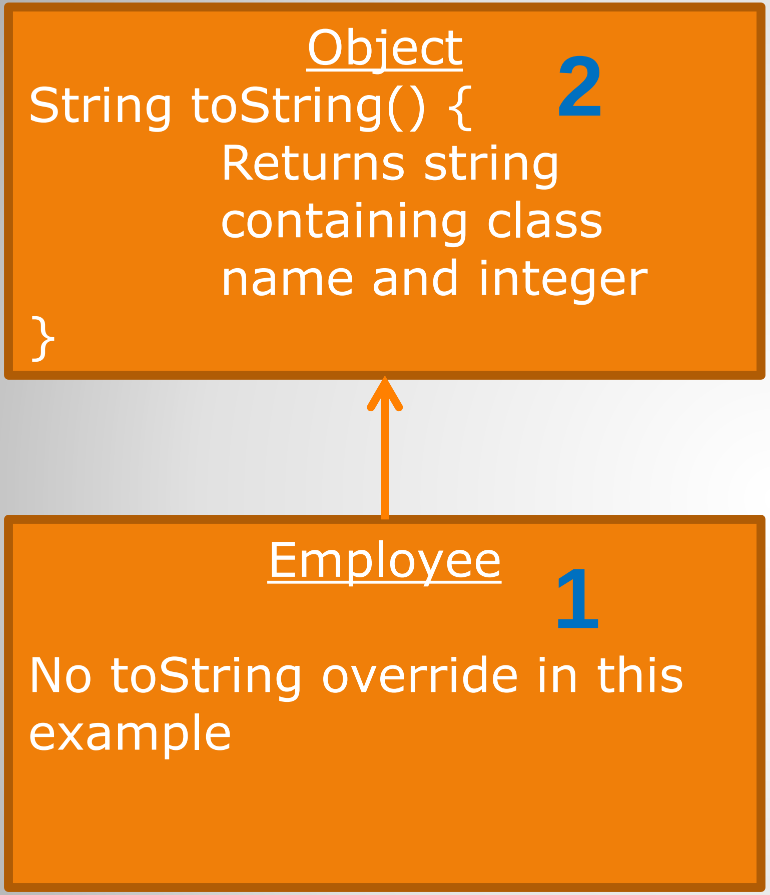
```
public class Employee {
    public String firstName;
    public String lastName;
}

public class Driver {
    public static void main(String[] args) {
        Employee e1 = new Employee();
        e1.firstName = "Arthur";
        e1.lastName = "Hoskey";
        System.out.println(e1);  ← Calls toString automatically!!!
    }
}
```

Prints something like:

```
csc211.hoskey.compare.tostring.Employee@4fee225
```

Passing a reference type to print will automatically call toString

toString Method Resolution



Object

```
String toString() {
        Returns string
        containing class
        name and integer
}
```

2

Employee

No toString override in this example

1

toString Method Resolution Example

1. **Compiler checks Employee for a toString method implementation (does not exist).**
2. **Compiler checks base class (Object in this case) for a toString method implementation (finds it)**

Compiler will use the Object toString method implementation.

Override toString

Overrides toString in Employee

```
public class Employee {
    public String firstName;
    public String lastName;


    @Override
    public String toString()  {
        String s = firstName + " " + lastName;
        return s;
    }
}
```

Use @Override annotation when overidding

- toString will return a string that contains the first and last names separated by a space.

```
public class Driver {
    public static void main(String[] args) {
        Employee e1 = new Employee();
        e1.firstName = "Arthur";
        e1.lastName = "Hoskey";
        System.out.println(e1.toString());
    }
}
```

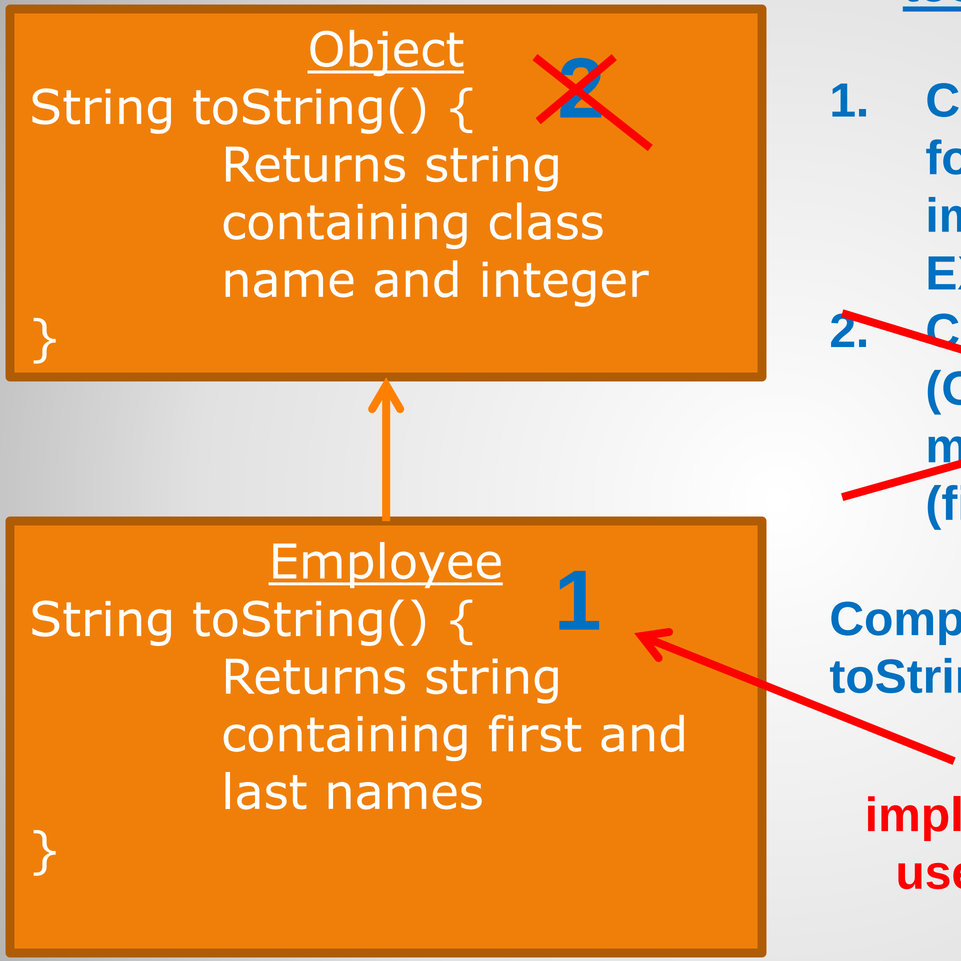
Prints:

```
Arthur Hoskey
```

Uses Employee override of toString (we added it on previous slide)

Using toString Override

toString Method Resolution



<u>Object</u> ~~2~~

```
String toString() {
        Returns string
        containing class
        name and integer
}
```

<u>Employee</u> 1

```
String toString() {
        Returns string
        containing first and
        last names
}
```

<u>toString Method Resolution Example</u>

1. Compiler checks Employee for a toString method implementation (DOES EXIST!!!).
2. ~~Compiler checks base class (Object) for a toString method implementation (finds it)~~

Compiler will use the Employee toString method implementation.

Finds an Employee implementation of toString so it uses it. No need to check the base class.

- Now we will cover the equals method…

equals

Object Compare (==)

```
public class Driver {
    public static void main(String[] args) {
        Employee e1 = new Employee();
        Employee e2 = new Employee();
        e1.firstName = "Arthur";
        e2.firstName = "Arthur";
        e1.lastName = "Hoskey";
        e2.lastName = "Hoskey";
        if (e1 == e2) {
            System.out.println("Equal");
        } else {
            System.out.println("Not equal");
        }
    }
}
```

Compares addresses

Prints:

```
Not equal
```

Object.equals() Method

equals() – Indicates whether some other object is "equal to" this one.

- There is a default implementation of equals() defined on the Object class.
- **The default implementation will test if the addresses of the objects are equal.**

Object equals Implementation

```
public class Driver {
    public static void main(String[] args) {
        Employee e1 = new Employee();
        Employee e2 = new Employee();
        e1.firstName = "Arthur";
        e2.firstName = "Arthur";
        e1.lastName = "Hoskey";
        e2.lastName = "Hoskey";
        if (e1.equals(e2)) {
            System.out.println("Equal");
        } else {
             System.out.println("Not equal");
        }
    }
}
```

No Employee equals method so it calls Object equals (address compare is used)

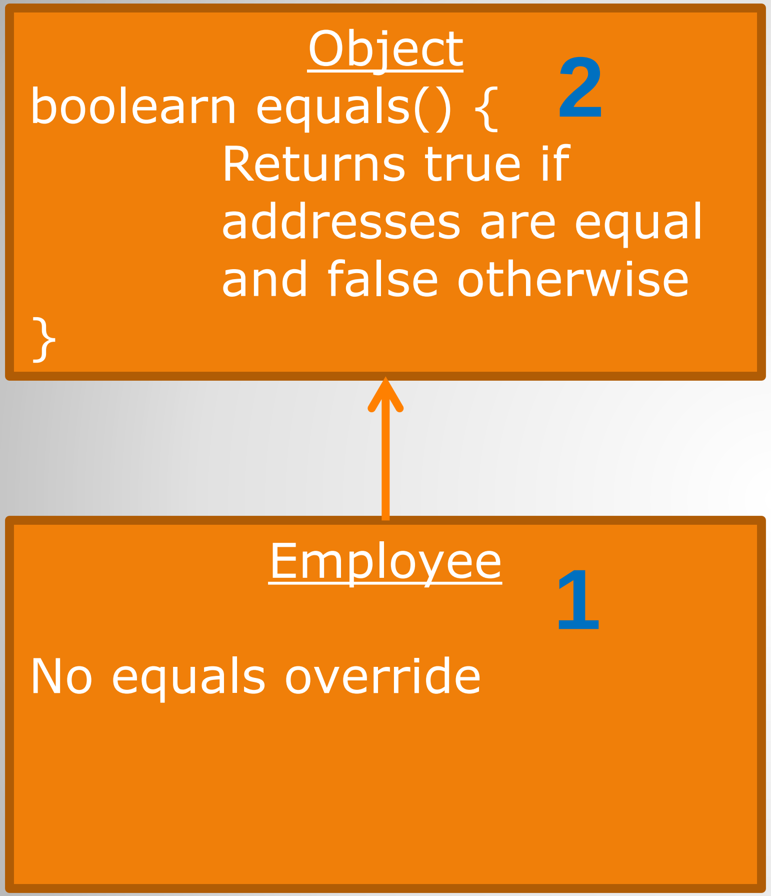
Prints:

```
Not equal
```

Note: The call to equals is being done on an Employee object (NOT A STRING OBJECT)

Object Compare (default equals)

Equals Method Resolution



```
boolearn equals() {
        Returns true if
        addresses are equal
        and false otherwise
}
```

No equals override

equals Method Resolution Example

1. **Compiler checks Employee for an equals method implementation (does not exist).**
2. **Compiler checks base class (Object) for an equals method implementation (finds it)**

Compiler will use the Object equals method implementation.

Override equals in Employee

```
public class Employee {
    public String firstName;
    public String lastName;

    @Override
    public boolean equals(Object obj) {
        Employee other = (Employee) obj; // Copy to Employee var

        if (firstName.equals(other.firstName) == false)
            return false;

        if (lastName.equals(other.lastName) == false)
            return false;

        return true;
    }
}
```

firstName and lastname are Strings so it is calling the String.equals method which performs a string value comparison

- equals will return true if both firstname and lastname string values are the same and false otherwise.

Override equals

```
public class Driver {
    public static void main(String[] args) {
        Employee e1 = new Employee();
        Employee e2 = new Employee();
        e1.firstName = "Arthur";
        e2.firstName = "Arthur";
        e1.lastName = "Hoskey";
        e2.lastName = "Hoskey";
        if (e1.equals(e2)) {
            System.out.println("Equal");
        } else {
             System.out.println("Not equal");
        }
    }
}
```

We added equals implementation to Employee so that one will be used!!!

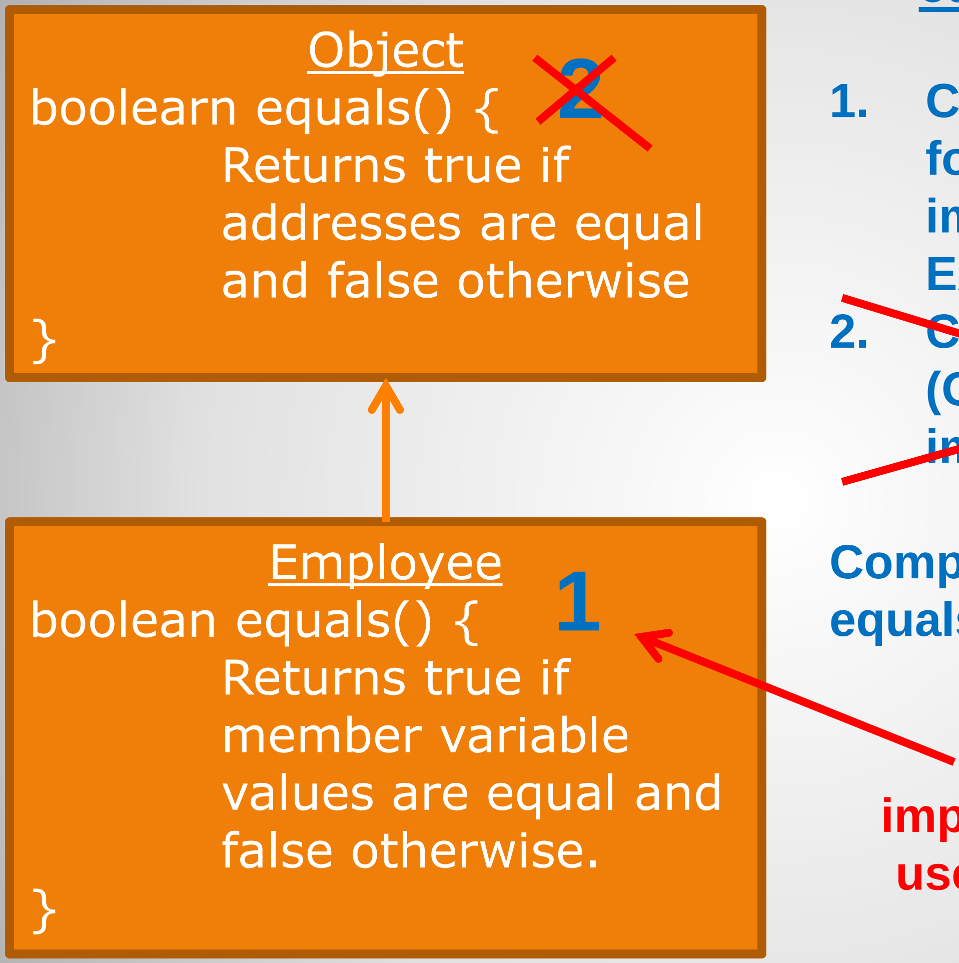
Prints:

Equal

Note: The call to equals is being done on an Employee object (NOT A STRING OBJECT)

Value Compare

equals Method Resolution



<u>Object</u> ~~2~~

```
boolearn equals() {
        Returns true if
        addresses are equal
        and false otherwise
}
```

<u>Employee</u> 1

```
boolean equals() {
        Returns true if
        member variable
        values are equal and
        false otherwise.
}
```

equals Method Resolution Example

1. Compiler checks Employee for an equals method implementation (DOES EXIST!!!).
2. ~~Compiler checks base class (Object) for an equals method implementation (finds it)~~

Compiler will use the Employee equals method implementation.

Finds an Employee implementation of equals so it uses it. No need to check the base class.

String equals

String.equals() Method

equals() – The String class provides an override of the equals method. This override compares string values (not addresses).

- This is why the equals method works on strings.
- **For equals to work on classes that you create you must override it yourself. You will have to add code to compare the member variable values or whatever else you want to test.**

- End of Slides

End of Slides