

# TOSTRING, EQUALS, HASHCODE

---

CS302 – Introduction to Programming  
University of Wisconsin – Madison  
Lecture 23

By Matthew Bernstein – [matthewb@cs.wisc.edu](mailto:matthewb@cs.wisc.edu)

# toString method

- In Java, each object inherits (“has by default”) a **toString** method
- **toString** returns a String representation of the object
- You should define the **toString** method such that the String is meaningful
- By default, this method will return the object’s **Hash Code**

# Override the **toString** method

@Override



You need this '@Override'. You will learn about this notation in more detail later

```
public String toString()
```

```
{
```

```
    // Return a String that represents
```

```
    // the current object
```

```
}
```

-- See in class example --

# Example

- You can now simply print the object itself:

```
Car carA = new Car("Ford", Mustang");  
System.out.println(carA);
```

- Your object's **toString** method is called by default whenever your object is treated like a String

# hashCode method

- An object's **hash code** is an integer that is “like” its address in memory
- Each object's hash code maps to a memory address
- You can retrieve an object's hash code using its **hashCode()** method
- Example:

```
Car carA = new Car();
```

```
// Get the car's hash code
```

```
int code = carA.hashCode();
```

# equals method

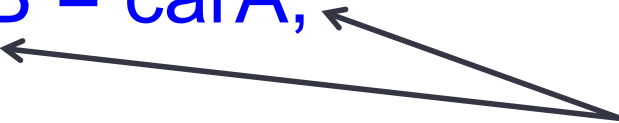
- Each object inherits a method called **equals()** that compares two objects
- If they are equal it returns true. If they are not equal it returns false
- What do we mean by equal?
- By default, the equals method compares the value returned by each method's **hashCode()** method
- Thus, by default, two objects are equal only if they share the same location in memory (i.e. they are the same object)

# Example of equals

```
Car carA = new Car("Volkswagen", "Beetle");
```

```
Car carB = carA;
```

Both carA and carB reference  
the same Car object



```
// This would return true
```

```
boolean isEqual = carA.equals( carB );
```

Compare carA with carB

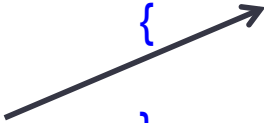


# Overriding the `equals` method

```
@Override
public boolean equals(Object o)
{
    if (! o instanceof Car.class)
    {
        return false;
    }

    if ( ((Car) o).getPrice() == this.price)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Cast the object  
parameter  
as a Car object



This method returns true  
whenever the two Car  
objects being compared  
have the same price



# Overriding the **equals** method

- You overwrite the **equals** method so that you can define how you want to compare two objects of your class
- For example, how do we would define whether two cars are equal?
  - Should they be equal if they have the same price?
  - Same make?
  - Same model?
  - It is up to you as the designer of the class

# Overriding hashCode

- The **hashCode** method and the equals method must be consistent.
- If two of your objects are evaluated to be equal by their **equals** method, then the two objects **MUST** return the same integer hash code
- How do we decide what hash code should we return?
- This is actually an involved topic...to make things simple...you should just return 0 from you **hashCode** method for all objects of your class