

# CMSC131

## Static and Non-Static Methods

# The main Method

Every Java example we've seen so far has been a class with a meaningful name which contains a main method as the starting point.

```
public class MeaningfulName {  
    public static void main(String[] args) {  
        //code goes here...  
    }  
}
```

For some of the examples we've seen, I've added additional **static** methods which are called from that **main** method...

```
import java.util.Scanner;

public class SimpleDoWhileWithMethod {
    public static void main(String[] args) {
        int userValue;
        Scanner sc = new Scanner(System.in);
        do {
            System.out.print(
                "Enter an odd number to continue: "
            );
            userValue = sc.nextInt();
        } while (!isOdd(userValue));
        System.out.println("Thank you.");
    }

    public static boolean isOdd (int num) {
        return (!(num%2)==0);
    }
}
```

# Objects

We have seen a few examples where we instantiate an object of a particular class and access members of those classes, specifically with the **Scanner** class and the **String** class.

```
Scanner sc = new Scanner(System.in);  
int x = sc.nextInt();  
answerHolder = answerHolder.toLowerCase();
```

The variables **sc** and **answerHolder** are references to things called objects.

Each object is distinct from another, but they all have access to methods within their class.

# Static -vs- Non-static Methods

Static methods are associated with the ***class as a whole***, not specific object instances.

Non-static methods are associated with a ***specific object*** and act upon that one object when invoked.

# Non-static methods example:

Consider the following code segment:

```
String firstName = "Evan";  
String lastName = "Golub";  
  
...  
firstName = firstName.toLowerCase();  
lastName = lastName.toUpperCase();  
  
...
```

What do firstName and lastName now contain if we were to print them?

# Adding Methods

A class can have as many static or non-static methods as you desire.

As we continue, we will soon explore the differences between these in more detail.

For now, we will only add static methods.

# Why use methods?

We will add methods either to:

- Simplify the readability of our code.
- Reduce redundancy in our code (eg: if you have a large block of code doing the same thing in two places, maybe you should create a method with that code and invoke the method from the two places).

# Static Method Prototype

```
public static return_type method_name (parameter_list) {  
    body_of_method  
}
```

- All of our **static** method definitions will follow the above syntax.
- You choose the appropriate return type.
- You choose the meaningful method name.
- You choose what information needs to be passed into the method.
  - Note: If it isn't passed into the method, it doesn't know about it even if the **main** method does.

# Calling any method

Two key things to consider for now:

- If the method returns a value, then your calling statement should deal with it somehow.
- The calling statement's argument list needs to match up with the method's parameter list by data type.

Copyright © 2012 : Evan Golub