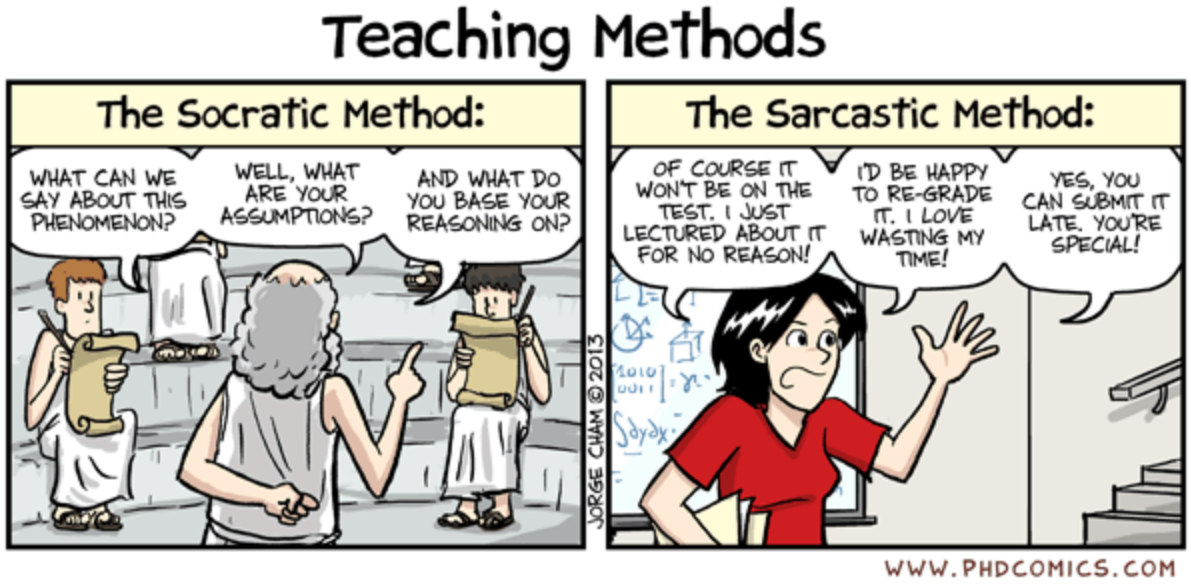
Mr. Fahrenbacher
AP Computer Science
Unit 3A - Methods

List of Java Concepts and Vocabulary From This Unit

- Methods
- Parameters
- Returning
- Pass By Value
- Methods in Other Classes
- Returning Early



Subroutines / Methods / Functions / Procedures

Read through the program below - it introduces some new, useful syntax. Find what's new and think about how the code works.

```
public class NurseryRyhme {

                                    //parameter list
public static void singAnimal(String name, String sound) {  //2 parameters (2 arguments),
        System.out.println("And on his farm he had a " + name);
        System.out.println("E-I-E-I-O");
        System.out.println("With a " + sound + " " + sound + " here");
        System.out.println ("And a " + sound + " " + sound + " there");
        System.out.println("Here a " + sound + ", there a " + sound);
        System.out.println("Everywhere a " + sound + " " + sound);
}

        public static void main(String[] args) {            //3 methods
                refrain();          //method call
                singAnimal("cow", "moo");
                refrain();

                refrain();
                singAnimal("pig", "oink");
                refrain();

                refrain();
                singAnimal("duck", "quack");
                refrain();
        }

        public static void refrain() {      //method declaration (no paramaters)
                System.out.println("Old MACDONALD had a farm");
                System.out.println("E-I-E-I-O");
        }


}
```

Questions:

1) What is the output of singAnimal("moo" "cow");

2) What about this code segment?

```
String name = "cow";
```

3) Is the following code legal?

```
String name = "cow";
```

```
String sound = "moo"
singAnimal(sound, name);
```

```
String sound = "moo"I
singAnimal();
```

Order of Methods

Does the output of the program below suprise you?

```
public class DoesOrderMatter {

	public static void methodA() {
		System.out.println("A");
	}

	public static void main(String[] args) {
		methodB();
		methodA();
		System.out.println("D");
		methodC();
	}

	public static void methodC() {
		System.out.println("C");
	}

	public static void methodB() {
		System.out.println("B");
	}

}
```

Output: B
A
D
C

Pass by Value

Look at the program below and then consider the output. Does it surprise you?

```
public class PassByValue {

        public static void main(String[] args) {

        double money = 0;

        int quarters = 1, dimes = 4, nickels = 5, pennies = 3;
        calculateValue(quarters, dimes, nickels, pennies, money);

        System.out.println("Money after method call: " + money);
        }

        public static void calculateValue(int q, int d, int n, int p, double money) {
                money = q * 0.25 + d * 0.10 + n * 0.05 + p * 0.01;
                System.out.println("Money in method: " + money);
        }
}
```

Program Output: **Money in method: 0.93**
Money after method call: 0.0

Return of the Method

Read through the game program below - it introduces some new, useful syntax. Find what's new and think about how the code works.

```
public class PassByValue {

        public static void main(String[] args) {

                int quarters = 1, dimes = 4, nickels = 5, pennies = 3;
                double money = calculateValue(quarters, dimes, nickels, pennies);

                System.out.println("Money after method call: " + money);
        }
                        //return type
        public static double calculateValue(int q, int d, int n, int p) {
                double result  = q * 0.25 + d * 0.10 + n * 0.05 + p * 0.01;
                System.out.println("Money in method: " + result);
                return result;
        }
}
```

Questions:

1) Which of the following commands are valid?

(a) double a = calculateValue(1, 2, 3, 4); YES!

(b) double b = calculateValue(1, 2, **3.0**, 4); NO!

(c) double c = calculateValue(1, 2, 3); NO! (missing the 4th parameter)

(d) **int** d = calculateValue(1, 2, 3, 4); NO! (return type is a double)

(d) if(calculateValue(1, 2, 3, 4) > 0.50) { YES!
        System.out.println("You have more than 50 cents!");
    }

(e) int q = 4, d = 3, n = 7, p = 2;
    double e = calculateValue(); NO!

A Frustrating Attempt at Writing a Method

What two mistakes us the coder making in the below program?

```
public class Average {

	//program should print out the average of 5 and 7
	public static void main(String[] args) {

		int avg = getAverage(5, 7);
		System.out.println(avg);

		//System.out.println(getAverage(5, 7));
	}

	public static int getAverage(int x, int y) {
		int average = (x + y) / 2;
		return average;
	}


}
```

Method Overloading

Look at the program below and then consider the output. Does it surprise you? Do you think it would be legal to write another area method that takes 1 integer parameter and returns an int?

```
public class AreaMadness {

	public static void main(String[] args) {
		double a1 = area(4);
		double a2 = area(5.0, 6.0);
		double a3 = area(3, 4);
		double a4 = area(1.0, 2);
		System.out.println(a1 + " " + a2 + " " + a3 + " " + a4);
	}

	public static double area(int side) {
		return 4 * side;
	}

	public static double area(int width, int height) {
		return width * height;
```

```
        }

        public static double area(double width, double height) {
            return width * height / 2;
        }
    }
```

Output: 16 15.0 12 1.0

Methods In Other Classes

Look at the program below that consists of two classes. Pay close attention to how the two classes interact with each other.

```
public class HaveSomeClass {

    public static void main(String[] args) {

        double area = Geometry.triangleArea(5, 6);
        int answer = theAnswerToLifeTheUniverseAndEverything();

        System.out.println("Area: " + area);
        System.out.println("Answer: " + answer);
    }

    public static int theAnswerToLifeTheUniverseAndEverything() {
        return 42;
    }
}
```

```
public class Geometry {

    public static double triangleArea(double h, double w) {
        return multiply(w, h) / 2;
    }

    private static double multiply(double a, double b) {
        return a * b;
    }

}
```

Questions:

1) Which of the lines of code do you think are invalid if they are written in the **HaveSomeClass** class? Why?

(a) double times = Geometry.multiply(4, 5);

(b) int answer = HaveSomeClass.theAnswerToLifeTheUniverseAndEverything();

(c) double area = triangleArea(7, 8);

Returning Before the End

The code on the left represents methods with more than one return statement that works properly, while the code has more than one return statement and does not work properly (syntax or logical errors). Try to identify why the code on the left works and what is wrong with the code on the right.

Works Properly	Does Not Work Properly

public static int max(int a, int b) {
  if(a >= b) {
  return a;
  }
  else {
  return b;
  }
}
	public static int max(int a, int b) {
	  int answer = 0;
	  if(a >= b) {
	  answer = a;
	  }
	  if(a < b) {
	  answer = b;
	  }
	  return answer;
	}

public static boolean isPrime(int x) {
 	
  for(int i = 2; i < x; i++) {
  if(x % i == 0) {
  return false;
  }
  }
  return true;
}
	public static boolean isPrime(int x) {
	
	  for(int i = 2; i < x; i++) {
	  if(x % i == 0) {
	  return false;
	  }
	  else {
	  return true;
	  }
	  }
	}

Some Rules / Notes to be Aware Of

1. The return command ends a method immediately

```
public static int sumOfDivisors(int n) {
	if(n < 1)
		return 0; //none of the code below executes if n is less than 1

	int sum = 0;
	for(int i = 1; i <= n; i++)
		if(n % i == 0)
			sum += i;
	return sum;
}
```

2. Because a return command ends a method immediately, no code can follow a return statement (breaking this rule is a syntax error)

```
public static int sumOfDivisors(int n) {
	if(n < 1) {
		return 0;
		System.out.println("Invalid input"); //syntax error, unreachable code
	}

	int sum = 0;
	for(int i = 1; i <= n; i++)
		if(n % i == 0)
			sum += i;
	return sum;
}
```

3. You can return in a void method

```
public static void printDivisors(int n) {
	if(n < 1)
		return; //method will stop if n is less than 1

	int sum = 0;
	for(int i = 1; i <= n; i++)
		if(n % i == 0)
			System.out.print(i + " ");
}
```