Assess Yourself

Implement a class to represent a person *as part of an app to suggest and visualise outfits for an online clothing store*.
What attributes and methods might it have?

This depends entirely on the application.
As software engineers, you need to get in the habit of getting a complete picture of what you're building, particularly when dealing with "clients".

Assess Yourself

```
public class Person {
    public int height.
    public int shoeSize;

    public Colour favouriteColour;
    public boolean wearsHats = false;

    public Shirt shirt;
    public Pants pants;
    public Shoes shoes;
    ...
}
```

This is not a good design... We will revisit this later in the semester. Can anyone suggest why?

Assess Yourself

```
public class Person {

    public Person(int height, int shoeSize,
                  Colour favouriteColour, boolean wearsHats,
                  Shirt shirt, Pants pants, Shoes shoes) {
        ... sigh ...
    }

    public String toString() {
        return String.format("Height: %d; Shoe size: %d;...",
                             this.height, this.shoeSize);
    }

}
```

Thankfully IDEs can autogenerate constructors and other standard methods... Please figure out how to do this.

Assess Yourself

```
public class Person {

    public void changeOutfit(Shirt shirt, Pants pants,
                             Shoes shoes) {
        this.shirt = shirt;
        this.pants = pants;
        this.shoes = shoes;
    }

    public int getSimilarityValue() {
        return <some magic computation>;
    }

}
```

The `changeOutfit` method looks like a constructor, but it's not; why not?

SWEN20003
Object Oriented Software Development

Methods

Semester 1, 2019

The Road So Far

- Java Foundations
  - Classes and Objects
  - Strings and Wrappers
  - Formatting

Lecture Objectives

After this lecture you will be able to:

- Define and use the **main** method
- Perform *abstraction* with methods
- Describe a variable's "scope"
- Better explain what "static" means for
  - Constants
  - Variables
  - Methods
- Explain and use "overloading"

In-class code found here

Writing Programs

At the moment we have a bunch of classes, but they don't actually *do* anything... There isn't anything to **run/execute**.

This is where the main method comes in.

Main

```
public static void main(String args[])
```

- Let's take a closer look at `main`

Main

```
public static void main(String args[])
```

- We'll leave this for a bit longer

Main

```
public static void main(String args[])
```

- We'll look at this again today

Main

```
public static void main(String args[])
```

- Let's start here

Defining Methods

```
<privacy?> <static?> <return type> <method name>(<arguments>)
```

- For now, `public` is default
- We know when to use `static`... right?
- Every method needs a return type (`int`, `double[]`, `void`...)
- Method names follow the same conventions as variable names
- Methods can have zero or more arguments

```
public static String[] magicalComputation(String[] data)
```

Writing Methods

```
public static String[] magicalComputation(String[] data) {
    String newData[] = new String[data.length];

    for (int i = 0; i < data.length; i++) {
        newData[i] = data[i].toUpperCase();
    }

    return newData;
}
```

- Writing code in a method, same as everything so far
- You must include the `return` statement
- `void` methods **can** use `return`, but not required
- Must be defined in a **class**
- Represents the class **performing an action**, or **receiving a message**

Methods

Keyword

return: Indicates that a method **ends**, and in most cases will **output** a value. A method can have several `return` statements, but **all** paths must reach a `return`

Pitfall: Return Statements

What happens after a return statement?

What does this code do?

```
public static String[] splitString(String string) {
    String[] words;

    return words;

    words = string.split(" ");
}
```

Nothing.
Any code after a return statement will. Not. Execute.

Assess Yourself

Write a *program* that creates a `Person` object with height of 182cm, shoe size of 12, large shirt size, and pants size of 40.

You may assume that the `Person`, `Shirt`, and `Pants` classes all exist; the only thing you have to do is write the **main** method.

Why Methods?

Why should we use methods?

- Prevents **code duplication**
- Improves **readability**
- Makes code **reusable** and **portable**
- Easier to **debug**
- Gives "important code" a **useful name**

Assess Yourself

What *methods* could you define for the following example?

- What would you call them?
- What arguments would they take?
- What would they do?

Write a program that gets continuous input from the user, in the form `<p1>-><p2>`, indicating that `p1` is a parent of `p2`. Once the input has stopped, your program must print a family tree with the "root" parent on the first line, their children on the next, and so on.

Assess Yourself

Write a program that gets continuous input from the user, in the form <p1>-><p2>, indicating that p1 is a parent of p2. Once the input has stopped, your program must print a family tree with the "root" parent on the first line, their children on the next, and so on.

```
public static String[][] reallocateData(String[][] relationships)
public static String[] splitRelationship(String relationship)
public static String[][] createFamilyTree(String[][] relationships)
public static String findRoot(String[][] relationships)
public static void printFamilyTree(String[][] familyTree)
public static void printTreeLayer(String[] layer)
```

Methods

Keyword

Method: The basic unit of abstraction (after classes) in Java; represents performing an *action* or *receiving a message*

Static

Keyword

static: Indicates a constant, variable, or method exists **without an object**. In other words, you do not need to create a variable to use something defined as `static`

- Static Example:

```
double x = Math.sqrt(10);
```

- Non-static Example:

```
Scanner scanner = new Scanner(System.in);
String text = scanner.nextLine();
```

Static Constants

```
public class Program {
        public static final int N_ELEMENTS = 10;

        public static void main(String args[]) {
                int elements[] = new int[N_ELEMENTS];
        }
}
```

- `N_ELEMENTS` is a *class constant*, available to any (`static`) method defined in the *Program* class
- Defined **in the class** but **outside a method**
- Somewhat equivalent to C's *`#define`*

Static Variables

```
public class Program {
        public static int numMethodCalls = 0;

        public static void magicalComputation() {
                ...
                numMethodCalls += 1;
        }

        public static void evenMoreMagicalComputation() {
                ...
                numMethodCalls += 1;
        }
}
```

- `numMethodCalls` is a *class variable*, available to any (`static`) method defined in the *Program* class
- Defined **in the class** but **outside a method**
- Somewhat equivalent to "global" variables

Static Methods

```
public class Program {
        public static final int N_ELEMENTS = 10;

        public static int computeValue(int x) {
                ...
        }
}
```

- `computeValue` is a *class method*, available to any other (`static`) method defined in the *Program* class
- Defined **in the class**
- Can only call/use other `static` methods and variables

Scope

Keyword

Scope: Defines when a constant, variable or method can be "seen"

A new "scope" is created in

- every class/file
- every method
- every "block" (between { })

If something is "out of scope", it can't be used, called, manipulated, or accessed.

Assess Yourself

```
public class Program {
    public static int numMethodCalls = 0;

    public static int methodOne() {
        int x = 10;

        for (int i = 0; i < 10; i++) {
        }
    }

    public static void methodTwo() {
        numMethodCalls += 1;
    }
}
```

Where is the variable `numMethodCalls` "in scope"?

1. Just `methodOne`
2. Just `methodTwo`
3. All static methods of the Program class
4. All non-static methods of the Program class
5. None of the above

Assess Yourself

```
public class Program {
    public static int numMethodCalls = 0;

    public static int methodOne() {
        int x = 10;

        for (int i = 0; i < 10; i++) {
        }
    }

    public static void methodTwo() {
        numMethodCalls += 1;
    }
}
```

Where is the method `methodOne` "in scope"?

1. Just `methodOne`
2. Just `methodTwo`
3. All static methods of the Program class
4. All non-static methods of the Program class
5. None of the above

Assess Yourself

```
public class Program {
    public static int numMethodCalls = 0;

    public static int methodOne() {
        int x = 10;

        for (int i = 0; i < 10; i++) {
        }
    }

    public static void methodTwo() {
        numMethodCalls += 1;
    }
}
```

Where is the variable `x` "in scope"?

1. Just `methodOne`
2. Just `methodTwo`
3. All static methods of the Program class
4. All non-static methods of the Program class
5. None of the above

Assess Yourself

```
public class Program {
    public static int numMethodCalls = 0;

    public static int methodOne() {
        int x = 10;

        for (int i = 0; i < 10; i++) {
        }
    }

    public static void methodTwo() {
        numMethodCalls += 1;
    }
}
```

Where is the variable `i` "in scope"?

1. Just `methodOne`
2. Just `methodTwo`
3. All static methods of the Program class
4. All non-static methods of the Program class
5. None of the above

Assess Yourself

What is the output of this code?

```
import java.util.Arrays;

public class Program {
    public static void main(String args[]) {
        int[] nums = new int[10];
        nums = initialise(nums);
        System.out.print(Arrays.toString(nums));
    }

    public static int[] initialise(int[] nums) {
        for (int i = 0; i < nums.length; i++) {
            nums[i] = i;
        }
        return nums;
    }
}
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Assess Yourself

What is the output of this code?

```
import java.util.Arrays;

public class Program {
    public static void main(String args[]) {
        int[] nums = initialise();
        System.out.print(Arrays.toString(nums));
    }

    public static int[] initialise() {
        int[] nums = new int[10];
        for (int i = 0; i < nums.length; i++) {
            nums[i] = i;
        }
        return nums;
    }
}
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Assess Yourself

What is the output of this code?

```
import java.util.Arrays;

public class Program {
    public static void main(String args[]) {
        int[] nums = new int[10];
        initialise(nums);
        System.out.print(Arrays.toString(nums));
    }

    public static void initialise(int[] nums) {
        for (int i = 0; i < nums.length; i++) {
            nums[i] = i;
        }
    }
}
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Pitfall: Mutation

- Remember that objects are **pointers** in Java
- When we pass objects to methods, we pass references
- References allow us to "mutate" objects, despite being in a different scope
- Very important once we look at privacy

Overloading

Keyword

Method Signature: The name of a method, and the number and type of its parameters

Keyword

Overloading: When methods share the same name, but differ in the number, or type of arguments in the method signature

Overloading Example

Base Method

```
void magicalComputation(int n)
```

Overloading

```
void magicalComputation(double n)
```

```
void magicalComputation(int n1, int n2)
```

```
void magicalComputation(int n1, int n2, int n3)
```

Pitfall: Overloading

Base Method

```
void magicalComputation(int n)
```

Not Overloading Correctly

```
void magicalComputation(int num)
```

```
int magicalComputation(int n)
```

Pitfall: Overloading

Terrible Programming

```
void badMethod(int x, double y)
```

```
void badMethod(double x, int y)
```

What happens if you call `bad(6, 7)`?

Assess Yourself

Define the following methods:

- `max(int n1, int n2)` - Computes the maximum of two integers
- `max(String s1, String s2)` - Computes the maximum of two `Strings`
- `max(String s1, String s2, String s3, String s4)` - Computes the maximum of four `Strings`
- `max(String[] s)` - Computes the maximum of an array of `Strings`

Assess Yourself

```
public static int max(int n1, int n2) {
    if (n1 > n2) {
        return n1;
    } else {
        return n2;
    }
}
```

```
public static String max(String s1, String s2) {
    if (s1.length() > s2.length()) {
        return s1;
    } else {
        return s2;
    }
}
```

Assess Yourself

```
public static String max(String s1, String s2,
                         String s3, String s4) {
    return max(max(s1, s2), max(s3, s4));
}
```

```
public static String max(String[] s) {
    String maxString = s[0];

    for (int i = 1; i < s.length; i++) {
        maxString = max(maxString, s[i]);
    }

    return maxString;
}
```

Assess Yourself

```
public static void main(String[] args) {
    System.out.println(max(1, 2));
    System.out.println(max("Hello World", "This is me"));
    System.out.println(max("Hello World", "This is me", "Life could be",
                                    "Fun for everyone"));
    System.out.println(max(new String[]{"Hello World", "This is me",
                                    "Life could be", "Fun", "For everyone"}));
}
```

```
2
Hello World
Fun for everyone
Fun for everyone
```

Metrics

Add attributes to the `Person` class to represent the person's shirt and pants size.

Next, add a **constant** to represent a *comfortable* range of sizes for a person's pants. For example, some people maybe be comfortable in their pants size ± 2.

Finally, add a method called `willFit` that checks whether the input argument will fit on the person. This method should be **overloaded** so it works with a `Shirt`, `Pants`, or `Shoes` object. Why are we using overloading?