**Week 1**

**Welcome and Project Overview: Visualizing Data**

**Computer Science:** The Science of using and processing large amount of information to automate useful tasks and learn about the world around us (using a computer)

A class is a type of data

An object is one such piece of data

The class is like the factory, and the object is what comes out of the factory. So you write a single class, and then you can use that class to create as many objects as you want.

Just a rule that Java enforces. That when you're creating a class, you have to put it in a file that has the same name as the class, and that's if the class is public

Member Variables: data the object need to store

Q. Where are member variables declared?

Ans. They are inside the class definition means Java will associate them with the class. But they are not local variables because they are not declared inside a method.

**This keyword:** This refers to the calling object, which is the object that called the method, or on which the method was called.

**Overloading of constructor**: There are two different copies of the constructor that take different numbers and types of arguments. They both do the same thing more or less in that they both create a new object, but how they assign values differ slightly

Note: You have to have some difference in the parameter list when you overload a method. You're not allowed to have a method with the same name, and the same parameter list, and a different return type

Q. When is it OK to have an overloaded method (i.e., method with the same name as another method) that has a different return type?

Ans: Only when the parameter list is also different. because The two parts of a method signature are the method name and the parameter list (number, type, and order of parameters). As long as the parameter list differs, Java allows you to overload the method, and even change the return type because the return type is not part of the method signature (it is ignored).

**Public Method:** I've declared my method distance to be public. Which means that I can access that method anywhere

**Private Method:** if I were to declare my member variables private, what that would mean is that those variables would only be accessible from within the class itself. So only inside this class definition. Nowhere else. No one else has access to either read or write those variables. Same thing goes for methods. If I make a method private, I can only call that method within the class I'm defining, and nowhere else.

Rule of thumb:

1. always make your member variables private.

2. methods can either be public or private, depending on whether they're for world use,Or whether they're maybe just helper methods that are designed to be used only within that class itself.

**Getters & Setters:**

let's focus on this rule of thumb that we should make our member variables private. So, if our member variables are private, nobody else can access them. They can't see their values, they can't change them. So, that seems like a bit of a problem. We probably sometimes want to give a little bit more access to the member variables in our class than that. So the way we get around that problem is through the use of what are called getters and setters.

**Getter:** The purpose of a **getter** is to take the value of a member variable that's private and expose it outside of that class. getter gets the value of the member variable latitude, which is private, and returns it to the outside world. So the variable itself is still private but now we can gain access to that variable by calling this getter.

if I try to access the variable directly say, print it out I get an error, but when I call that getter method, that's perfectly fine. So I can actually get access to the value of that variable through the getter.\

**Setter:** actually do want to allow somebody to change the value of the variable. In that case I would provide what's called a setter. A setter sets the value of our member variable, usually a private member variable.

**Getter and the Setter allows us more control as class designers.**

Q. Why private and public accessifiers are used?

Ans. If You make everything private and you end up with code that cannot interact with anything else. You make everything public and you end up with code that's extremely vulnerable, maybe even useless.