Naming

Naming things (= *variables, properties, functions, methods, classes*) correctly and in an understandable way if **an extremely important part of writing clean code.**

Indeed - if poor names are chosen - pretty much all other concepts taught throughout the course will **not help that much**.

Be Descriptive

Names have **one simple purpose**: They should **describe** what's stored in a variable or property or what a function or method does. Or what kind of object will be created when instantiating a class.

If you keep that in mind, coming up with good names should actually be straightforward - though coming up with the **best** name for a given variable/ property/ function/ ... will of course still require some practice and often **multiple iterations**. That's normal though - clean code is written by iterating and improving code over time!

Naming Rules

Variables & Properties

Variables and properties hold data - numbers, text (strings), boolean values, objects, lists, arrays, maps etc.

Hence **the name should imply which kind of data is being stored**.

Therefore, variables and properties should typically receive a **noun** as a name. For example: `user`, `product`, `customer`, `database`, `transaction` etc.

Alternatively, you could also use a **short phrase with an adjective** - typically for storing **boolean values**. For example: `isValid`, `didAuthenticate`, `isLoggedIn`, `emailExists` etc.

Typically, if you can be more specific, you **should** be more specific.

For example, prefer `customer` over `user` if the code at hand is doing customer-specific operations with that data. This makes your code easier to read and understand.

Functions & Methods

Functions and methods can be called to then **execute some code**. That means that they perform **tasks and operations**.

Therefore, functions and methods should typically receive a **verb** as a name. For example: `login()`, `createUser()`, `database.insert()`, `log()` etc.

Alternatively, functions and methods can also be used to primarily produce values - then, especially when producing **booleans**, you could also go for **short phrases with adjectives**. For example: `isValid(...)`, `isEmail(...)`, `isEmpty(...)` etc.

You should try to **avoid** names like `email()`, `user()` etc. These names sound like properties. Prefer `getEmail()` etc. instead.

As with variables and properties, if you can **be more specific**, it typically makes sense to use such more specific names. For example: `createUser()` instead of just `create()`.

Classes

Classes are used to **create objects** (unless it's a static class).

Hence the class name should **describe the kind of object it will create**. Even if it's a static class (i.e. it won't be instantiated), you will still use it as some kind of container for various pieces of data and/ or functionality - so you should then describe that container.

Good class names - just like good variable and property names - are therefore **nouns**.

For example: `User`, `Product`,`RootAdministrator`, `Transaction`, `Payment` etc.

Avoid Generic Names

In most situations, you should **avoid generic names** like `handle()`, `process()`, `data`, `item` etc.

There can always be situations where it makes sense but typically, you should either make these names more specific (e.g. `processTransaction()`) or go for a different kind of name (e.g. `product` instead of `item`).

Be Consistent

An important part of using proper names is **consistency**.

If you used `fetchUsers()` in one part of your code, you should also use `fetchProducts()` - and not `getProducts()` - in another part of that same code.

Generally, it doesn't matter if you prefer `fetch...()`, `get...()`, `retrieve...()` or any other term but you should be consistent!