Type Hierarchies, Polymorphism & Dispatching

Type Hierarchy

What's Important?

Object-oriented languages let us extend the programming language with types related to the problem we are trying to solve. Putting these types in a hierarchy lets us record the similarities between the types.

Abstract Class/Inheritance

What's Important?

Inheritance lets us reuse code between two different implementations. Putting code in one place saves duplicating the code. When we fix an error, we don't have to remember the code was in two places.

Inheritance with super

What's Important?

We can refine or override code that we are using through inheritance. This ability broadens how we can use inheritance to capture similarities.

Multiple implements

What's important?

We can use one object from multiple perspectives by declaring more than one type for a Java class. We can use this ability to state that some code only uses an object from a particular perspective.

Type Substitution

What's important?

Lets us abstract code that manipulates objects.
Lets us introduce new functionality through subtypes without changing the main parts of the codebase

Overloading

What's important?

In Java, we can have multiple methods of the same name on a type to capture problem domain. The methods must differ in the types of the parameters.