Video Lecture Notes:

**ABSTRACT CLASSES:** an outline

* exist to be extended;
* when you know the methods you want to use, but now exactly how you want to use them
* to provide functionality we know all subclasses will use/must use (compile time safety) (code reuse)
* they cannot be instantiated; prevents you creating an instance
* force you to expose your class as inheritable & cant mark as final

Abstract Methods:

* methods signature, return type, and the keyword ‘abstract’, ends with ;

no block (curly braces w/ implementation)

* any class extended from the class must inherit these methods
* can’t have abstract and private methods

\*Don’t ‘need’ any abstract methods in an abstract class, but 1 abstract methods makes the class abstract

polymorphism: <https://www.youtube.com/watch?v=pt1S11yX-7k>

interfaces, defender methods

**OVERRIDING METHODS:**

* to replace/modify functionality in child class methods
* @override annotation 🡪 metadata hint for the compiler
* Can change variable to be typed as the parent class; the instance is still typed of the child class

ParentClass child = new Childclass ( );

* If you want to run the parent method, use the keyword ‘super’ 🡪 super.method-name
* Always add a comment when overriding methods to:

-Make clear that is an overridden method

-Document what is different with this implementation

Preventing user from overriding a method:

* Keyword ‘final’
* Cannot override a final methods
* Final class cannot have child classes

Create a child class & assign it to the parent class; create a class and use the child implementation

Shadowing: parent variable being shadowed by the child variable

* Must cast class to the parent class

\*Attributes should be private – accessed only with getters & setters

- using the getter will always get the parent version