Encapsulation

(OOP Core Concept 1 - Encapsulation)

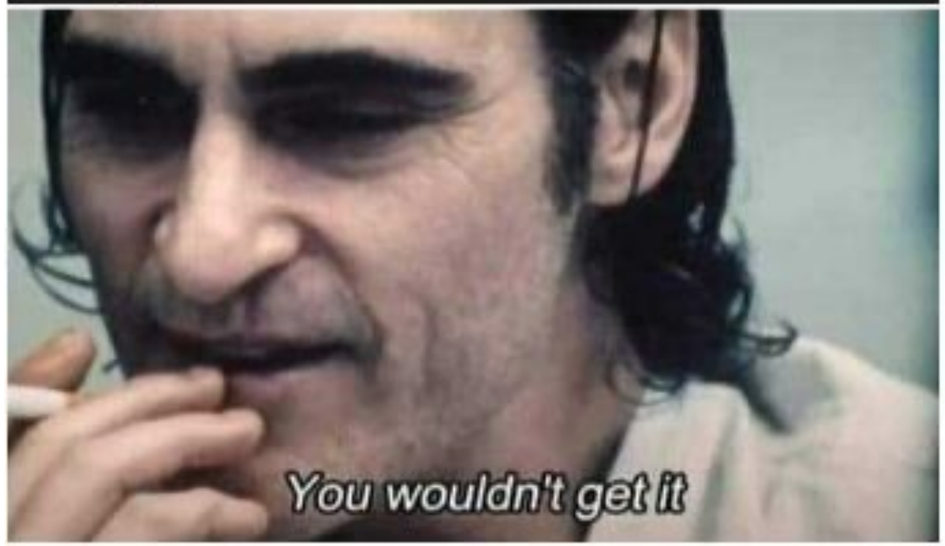
Learning objectives

- Encapsulation
- Constructors
- Getters and setters

Learning objectives

- You will get this joke:

```
1   public class Meme
2   {
3       private Joke joke;
4
5       public void setJoke(Joke newJoke)
6       {
7           this.joke = newJoke;
8       }
9   }
```



ACTLEA
Active Learning

Encapsulation

- Encapsulation is a fundamental OOP concept
- Encapsulation means hiding complexity (hiding data and behavior)
- It is accomplished by using access modifiers like private
- Only expose methods and variables with public access if they are meant to be accessed from the outside, otherwise keep them private

Benefits of encapsulation

- Improved maintainability - changes to internal code do not affect external code
- Complexity decreases with fewer allowed dependencies between classes
- Access to data is handled by getter and setter methods - variables can be read-only or write-only
- Enhances reusability - easier to reuse self-contained independent classes

Achieving encapsulation

- Declare all variables private
- Provide constructors to be able to initialize variables when the object is created (means read-only if combined with no public setter method)
- Provide public setter methods only to variables that should be able to be modified from the outside
- Provide public getter methods only to variables that should be visible from the outside
- Declare all methods as private except the ones that should be used from the outside

Constructors

- Constructors are special methods in Java that initialize objects
- They have the same name as the class and are called when an object is created with the new keyword
- Constructors have no return type
- If a class doesn’t have a constructor with no arguments, a default constructor with no arguments and no code is provided by Java

The purpose of constructors

- Usually constructors have input arguments that are used to initialize instance variables in the object
- Constructors can also be used for setting default values to instance variables
- Constructors can be used to set up necessary resources or perform other setup tasks that should be done when a new object is created

The this keyword

- this is a keyword in Java to specify calling a variable or a method in the same object
- this can be used to distinguish instance variables from local variables like input arguments with the same name
- this is often used when assigning values to instance variables in constructors

Demo 1 - Constructors

- Creating constructors
- Initializing instance variables
- The this keyword
- Using constructors to create new objects

Exercise 1 - Constructors

- Reuse the solution from the previous exercise with the Book class (make a copy of the previous solution and refactor the code in the copy)

- Create a constructor in the Book class that initializes the title, author and price

- Then use this constructor to create a few books in the main method instead of the default constructor, this will create a new initialized book in one line of code

Constructor overloading

- Constructors can be overloaded just like normal methods
- This means multiple constructors in the same class with different input arguments
- The appropriate constructor is chosen based on the input arguments used when calling the constructor with the new keyword when an object is created
- The this keyword can be used to call another constructor from within a constructor

Demo 2 - Constructor overloading

- Constructor overloading
- The default constructor
- Using this to refer to another constructor

Exercise 2 - Constructor overloading

- Reuse the solution from the previous exercise with the Book class
- Create a few overloaded constructors with different input arguments
- Use the different constructors from the main method
- Try to call another constructor from within a constructor with the this keyword

Getters and setters

- Making all variables private is good for encapsulation, but how will you then access these variables?

- Create public getter and setter methods that can be called from the outside and can also access the private variables in the same object

- But only create getters for the variables that should be visible from the outside and only create setters for variables that should be modified from the outside

Demo 3 - Getters and setters

- Making all variables private
- Creating getters and setters
- Using getters and setters

Exercise 3 - Getters and setters

- Reuse the solution from the previous exercise with the Book class
- Make all variables private
- Create getters and setters
- Use the getters and setters from the main method
- Do you need setters? Maybe you only need a constructor and getters?

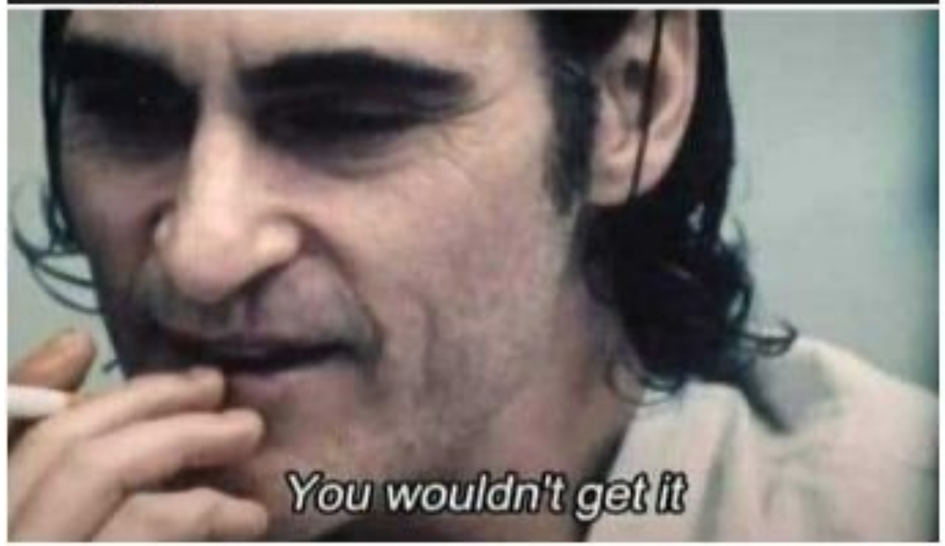
Learning objectives

- Encapsulation
- Constructors
- Getters and setters

Learning objectives

- You will get this joke:

```
1   public class Meme
2   {
3       private Joke joke;
4
5       public void setJoke(Joke newJoke)
6       {
7           this.joke = newJoke;
8       }
9   }
```



ACTLEA
Active Learning