Java Programming 1

PROFESSOR: CÂI FILIAULT

Topics for this week

- Object oriented programming
- Inheritance
- Public, private protected
- Constructor
- Properties and actions

Object-Oriented Programming

Understanding Object Oriented Programming (OOP)

The key to learning OOP is understanding two fundamental concepts

- The difference between a class and an object
- Many programmers use these terms interchangeably, but this is incorrect. The next few slides we will begin to understand the relationship between class and object

What is a Class?

- A class is a template used to generate objects. A class can be thought of as a blueprint for an object. We will use the class every time we want to construct an object

We will use a car as an example:

Object-Oriented Programming

Imagine a car

Object-Oriented Programming

Car:

- I didn’t specify how many wheels the car had
- What color the car was
- What type of seats the car had
- How many seats the car had
- How fast the car was able to go
- What type of wheels the car had
- Or even if the car was manual or automatic.
- But every single one of you knew what a car was and what it can do

Object-Oriented Programming

A class is that abstract idea of a car

A class can have many properties and methods associated with them that can make them act in different ways:

- The amount of wheels, the color, the size etc. are all properties.
- Accelerating the car, shifting gears, applying the breaks etc. are all actions or methods that are associated with the class

An object is an implementation of that class.

When I asked you to imagine a car, the car you pictured came from the same class but had different properties about it.

Object-Oriented Programming

- Imagine you were tasked with creating a streaming service:
- You need to programmatically represent many different movies that your streaming service is to offer to subscribers.
- What are all the various pieces of information related to a movie?

Object-Oriented Programming

Movie info:

Movie	
Properties	Actions
• Title • Director • Publication date • Actors • Screenwriter • Genre • Rating	• Play() • Pause() • Stop() • Rewind() • Fast-forward() • Skip() • ChapterSelect()

Object-Oriented Programming

- What data type can represent all this information at once?
- The truth is there is no known data type that can store all this information and actions in a meaningful manner.
- This is where we use object-oriented programming
- Each movie can be constructed using a Movie class which defines all properties and behaviors of all movies.
- When we create a class, we are creating a definition/blueprint for the construction of a Movie datatype

Creating a class

Important concepts:

- What is a constructor?
- What is a method?
- What is public, private and protected?

Creating a class

What is a constructor?

A constructor is a magic method that you create within a class. The method has no return type and shares the same name as the class. The method is declared as follows:

```
public class Movie{
    public Movie(){

    }
}
public static void main(String[] args){
     Movie movie = new Movie();
}
```

Creating a class

When an object is created using the new keyword, the creating class automatically runs the movie() method. We can think of a constructor as the method that creates the object.

movie() is a method and as such we can pass parameters to this method. Any parameters provided when creating a new object will be passed to the movie().

Movie movie = new Movie(“The Hobbit: An unexpected journey”, “2012”);

The previous code would need the following constructor

```
public Movie(String title, String pubDate){


 }
```

Creating a class

What is private, public and protected information?

Encapsulation:

The process of collecting data and functions into a single unit. Encapsulation allows the user to selectively share information by changing its visibility utilizing public, private and protected information. Allowing for a layer of abstraction from the complexity of the object.

Scope

Private Information:

Private information is information that can only be accessed from within a class, or any instance of that class.

```
class Person {
    private int age;
}
```

Scope

Private Scope:

We can only reference this variable from within the class.

```
class Person {
    private int age;
    public void printAge(){
        System.out.println(this.age);
    }
}
```

Scope

Public Scope:

We can access the public information everywhere in our code:

```
class Person {
    public String name;
    public String printName(){
        System.out.println(this.name);
    }
}
```

Scope

Public Scope:

We can access the public information everywhere in our code:

```
public static void main(String[] args){
   Person cai = new Person();
   System.out.println(cai.name);
}
//this is valid code; the name member variable can be accessed from anywhere.
```

Scope

Protected information:

Protected information is information that can be accessed only from the class or any derived class. Also any instance from the class or its derived classes. Protected is almost an extension of private. You would use protected when you need to access the variable from sub class of a class. We have not yet covered these concepts and will not delve too deep.

Protected Scope:

The protected scope is outside of the scope of this lesson. We will rehash this when we cover super and sub classes.

Creating a new class file

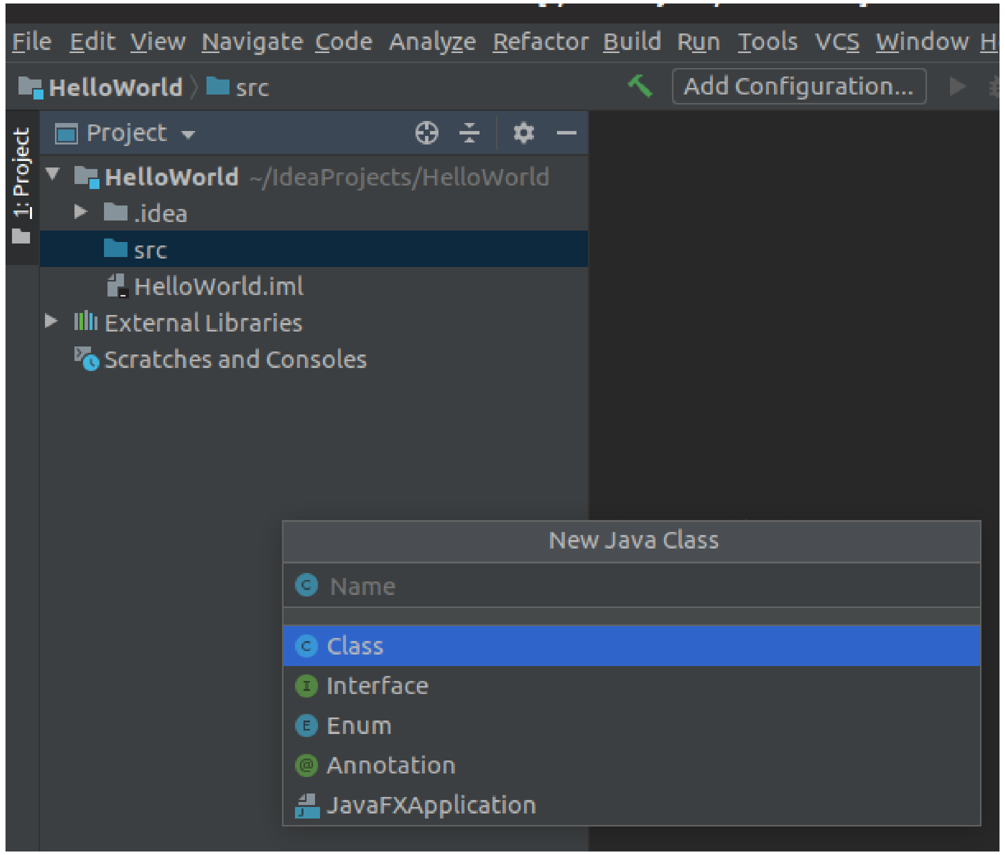
Next, we will need to create a new class file.

Right click on the **src** folder and select:

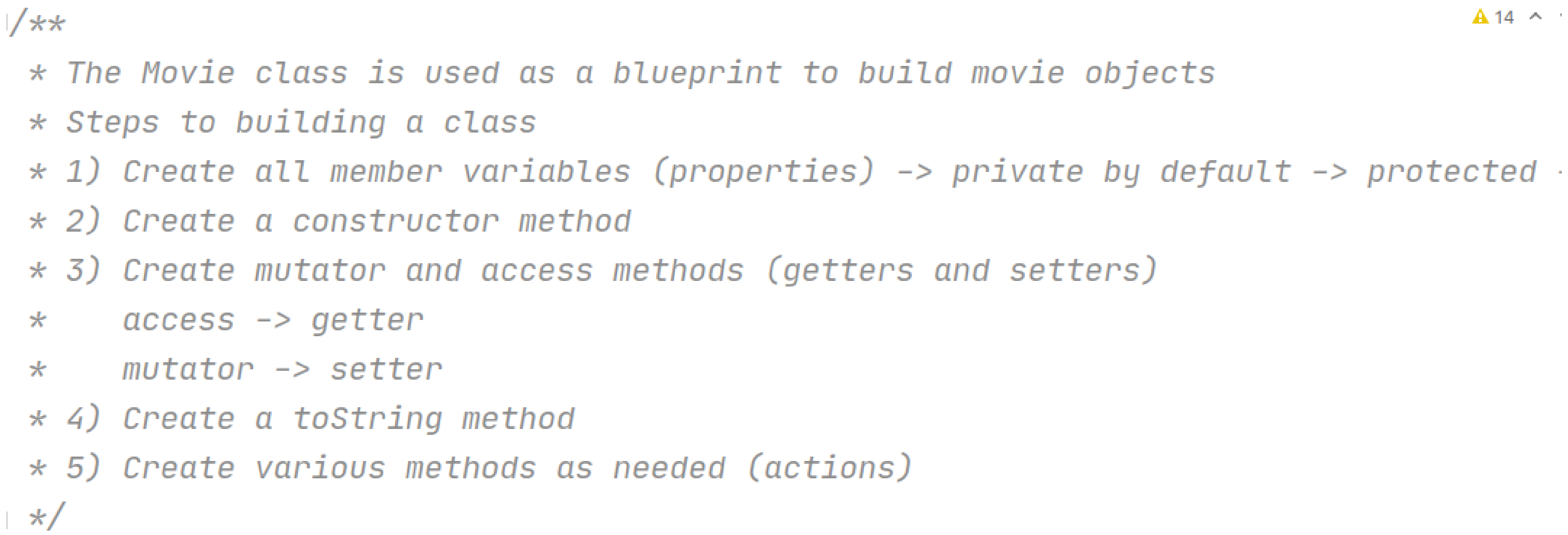
New> Java Class

Creating a new class is like creating a new program

Name the class **Movie** and hit **enter**



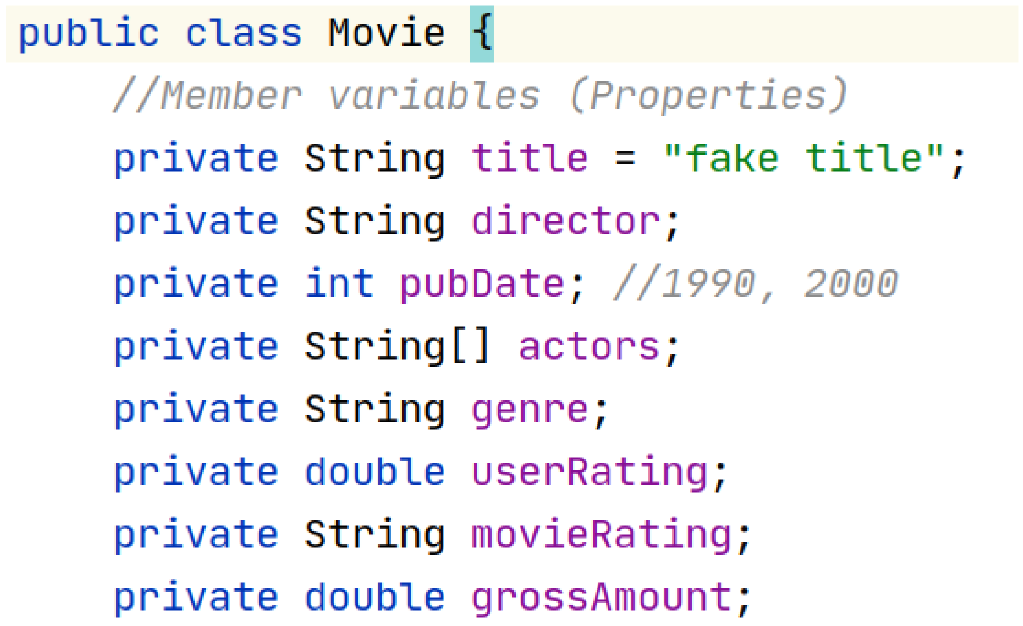
Creating a class



```
/**
 * The Movie class is used as a blueprint to build movie objects
 * Steps to building a class
 * 1) Create all member variables (properties) -> private by default -> protected
 * 2) Create a constructor method
 * 3) Create mutator and access methods (getters and setters)
 *    access -> getter
 *    mutator -> setter
 * 4) Create a toString method
 * 5) Create various methods as needed (actions)
 */
```

Creating a class

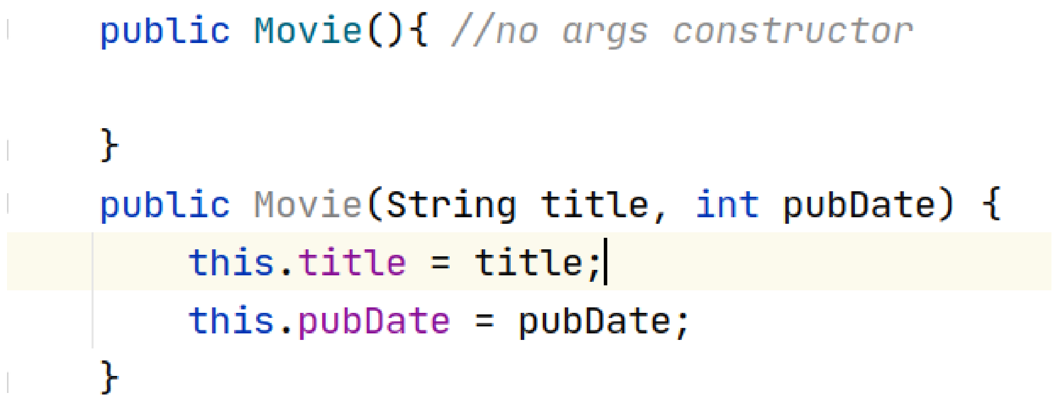
- The first thing we need to do when working on a class is to define the various properties that we would want an implementation of the class to have
- We can see that each movie will have varying information and data types from title to grossAmount.
- Each of these member variables are set to private by default



```java
public class Movie {
    //Member variables (Properties)
    private String title = "fake title";
    private String director;
    private int pubDate; //1990, 2000
    private String[] actors;
    private String genre;
    private double userRating;
    private String movieRating;
    private double grossAmount;
```

Creating a class

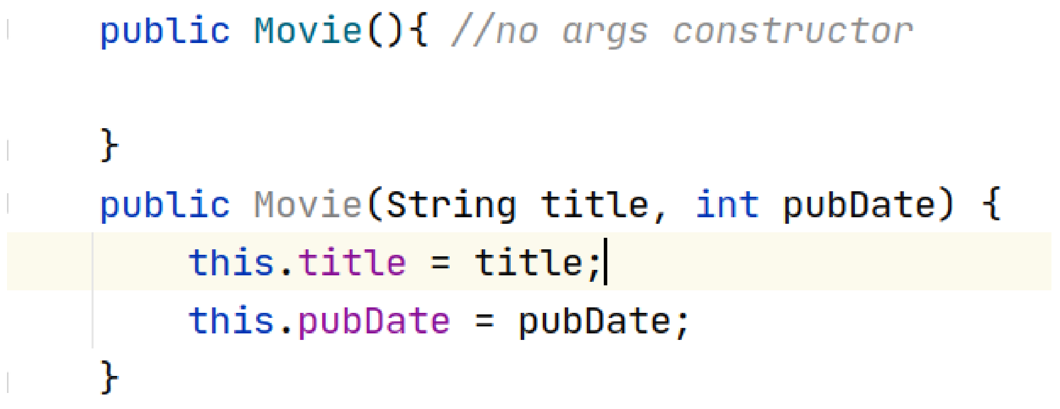
- The next thing that is needed are constructors.
- Constructors are used to help with the creation of new objects.
- Much like regular methods, constructors can have multiple implementations.
- We can see we have provided two different constructor methods
- The first is a no-args constructor (A constructor that contains no arguments)
- The second is a constructor that takes in the title and pubDate.



```
    public Movie(){ //no args constructor

    }
    public Movie(String title, int pubDate) {
        this.title = title;
        this.pubDate = pubDate;
    }
```

Creating a class

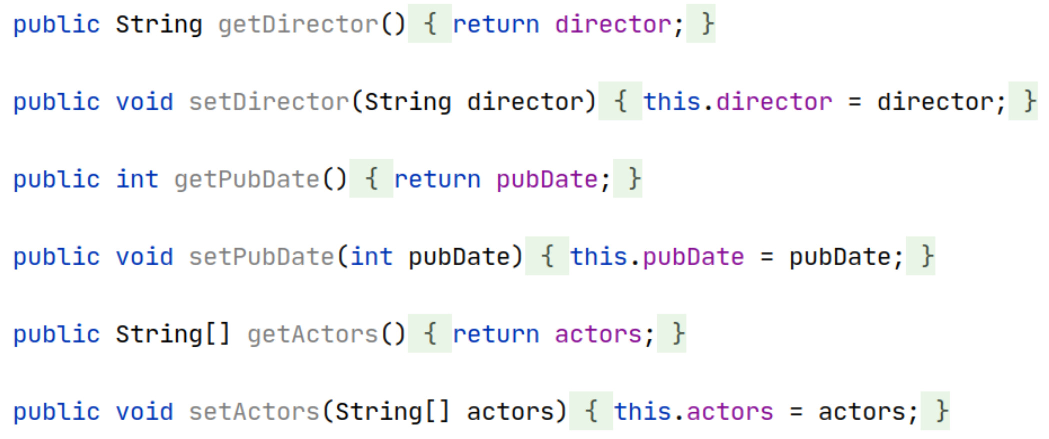
- Note that we can see the "this" keyword
- This represents the class itself.
- You will often see the "this" keyword used inside of classes.
- "this.title" refers to the title member variable that belongs to the class
- In other words we are saying take the member variable title and set it to the value that is the parameter title.



```
public Movie(){ //no args constructor

}
public Movie(String title, int pubDate) {
    this.title = title;
    this.pubDate = pubDate;
}
```

Creating a class

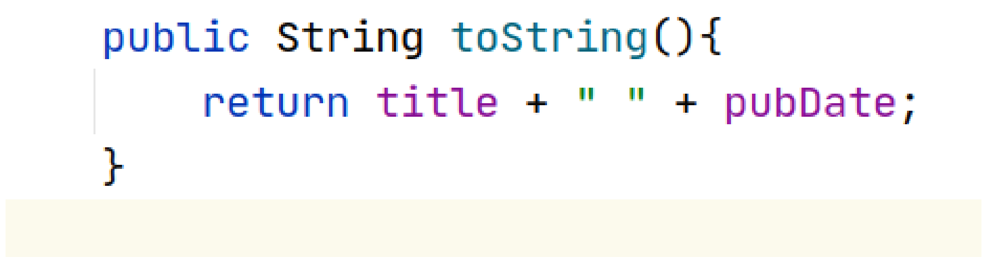
- We then begin to build accessor and mutator methods (Getters and setters)
- These are public methods that can be used to allow a user to read and write the various properties of an object
- Getters and setters permit us to provide controlled access to these values .
- We can provide code that will allow us to stop undesired values from being set to the properties.



```java
public String getDirector() { return director; }

public void setDirector(String director) { this.director = director; }

public int getPubDate() { return pubDate; }

public void setPubDate(int pubDate) { this.pubDate = pubDate; }

public String[] getActors() { return actors; }

public void setActors(String[] actors) { this.actors = actors; }
```

Creating a class

- Finally we are going to create a toString() method.
- We are going to store all information about the object in the toString method
- The toString method is another magic method that exists inside of a class.
- The magic method will be "magically" called whenever an instance of the class is treated like a string.
- Suppose you tried to concatenate an object into a string.
- The toString() method would be magically run and return its result.



```
public String toString(){
    return title + " " + pubDate;
}
```

Homework

Read chapter 9 of your textbook

Next Week

Super, Sub and Extending Classes,

Inheritance, Method Overriding, Overloading