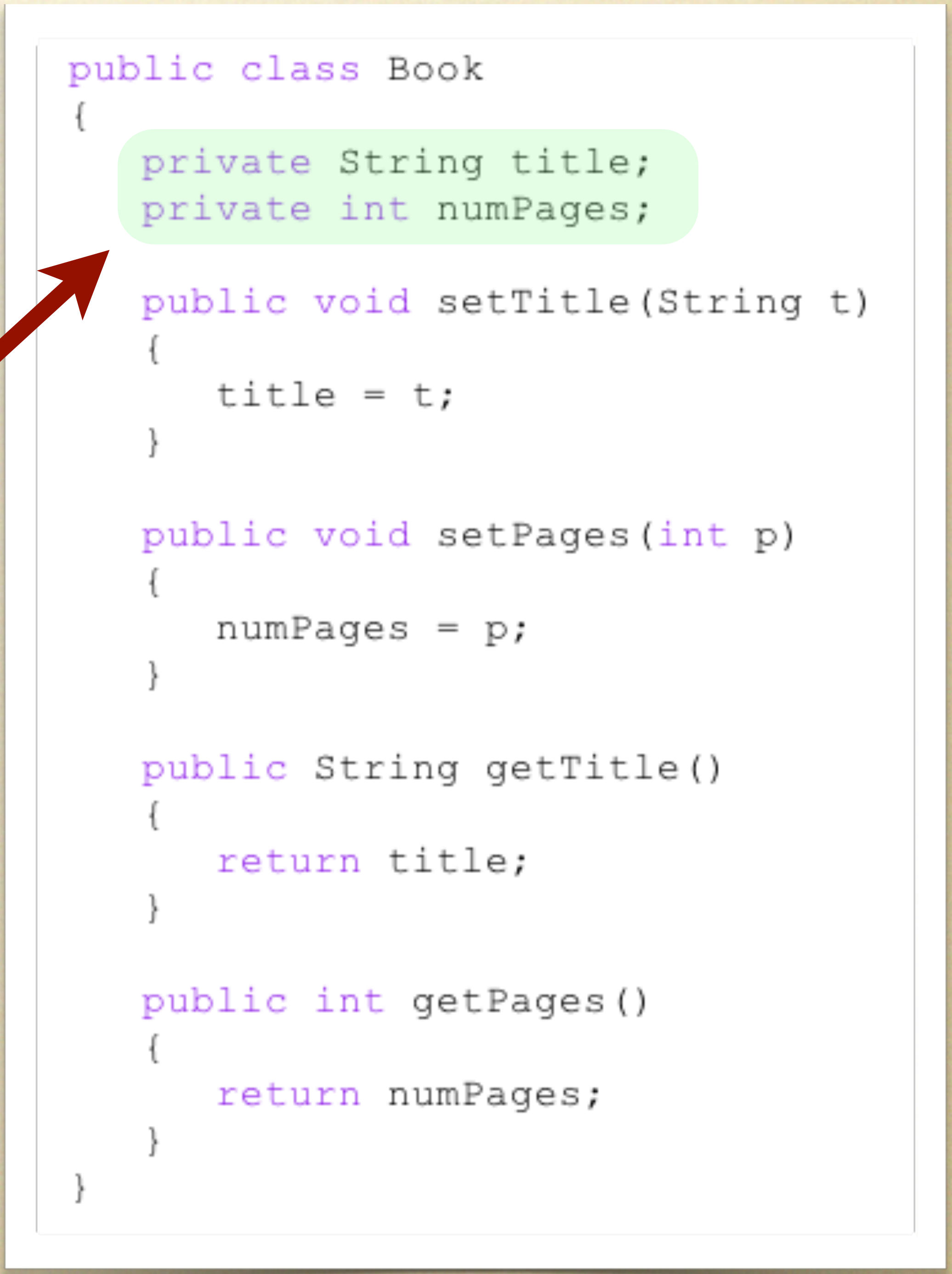
Image Credit: Julius Schorzman

Java Lesson 1

Concept Review

Inheritance

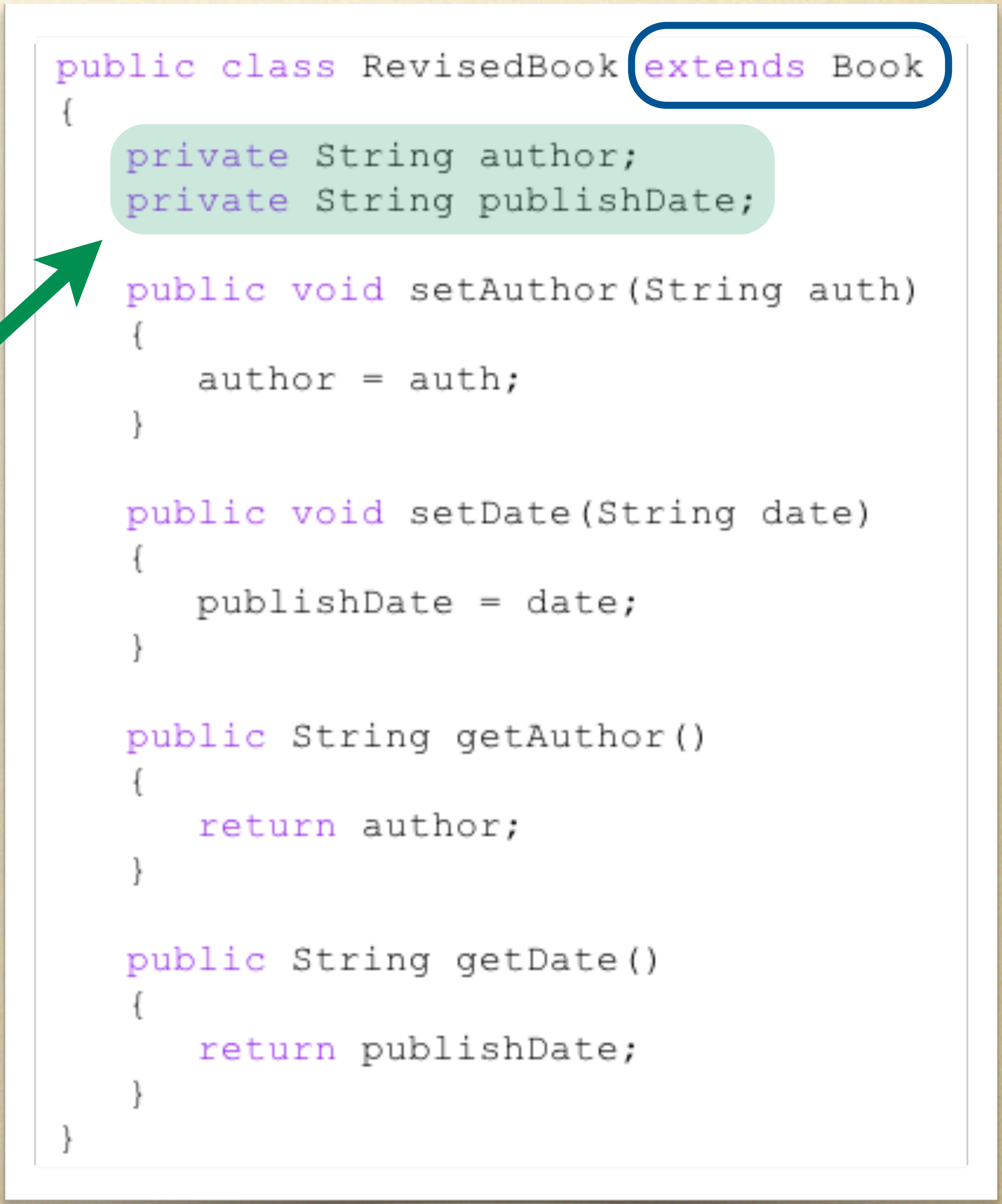
- Inheritance is a powerful feature in Java that allows you to "extend" the abilities of another Java class.
- Consider the following Book class. Notice that it stores both a title and the number of pages.
- Yet, what if we wanted to improved this class, without changing the original Book class? With inheritance, it's easy to do!



```java
public class Book
{
    private String title;
    private int numPages;

    public void setTitle(String t)
    {
        title = t;
    }

    public void setPages(int p)
    {
        numPages = p;
    }

    public String getTitle()
    {
        return title;
    }

    public int getPages()
    {
        return numPages;
    }
}
```

Inheritance

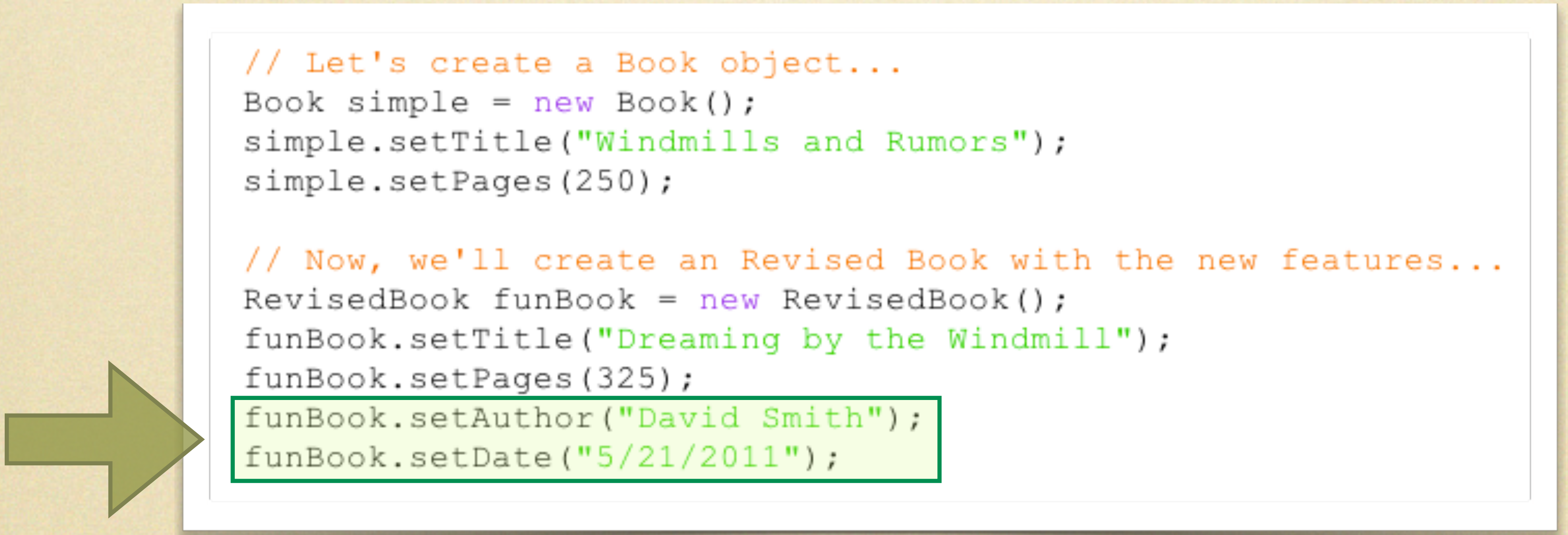
- By "extending" a class, we can add features to the Book class, as desired.
- For example, say that we wanted to add a data field to store the author's name and publishing date. It's easy to do!
- Notice that our revised book now "extends" the original Book class.



```
public class RevisedBook extends Book
{
    private String author;
    private String publishDate;

    public void setAuthor(String auth)
    {
        author = auth;
    }

    public void setDate(String date)
    {
        publishDate = date;
    }

    public String getAuthor()
    {
        return author;
    }

    public String getDate()
    {
        return publishDate;
    }
}
```

Inheritance

- Notice that the RevisedBook has the same features as the original Book class, but with the additional improvements, as shown below:



```
// Let's create a Book object...
Book simple = new Book();
simple.setTitle("Windmills and Rumors");
simple.setPages(250);

// Now, we'll create an Revised Book with the new features...
RevisedBook funBook = new RevisedBook();
funBook.setTitle("Dreaming by the Windmill");
funBook.setPages(325);
funBook.setAuthor("David Smith");
funBook.setDate("5/21/2011");
```

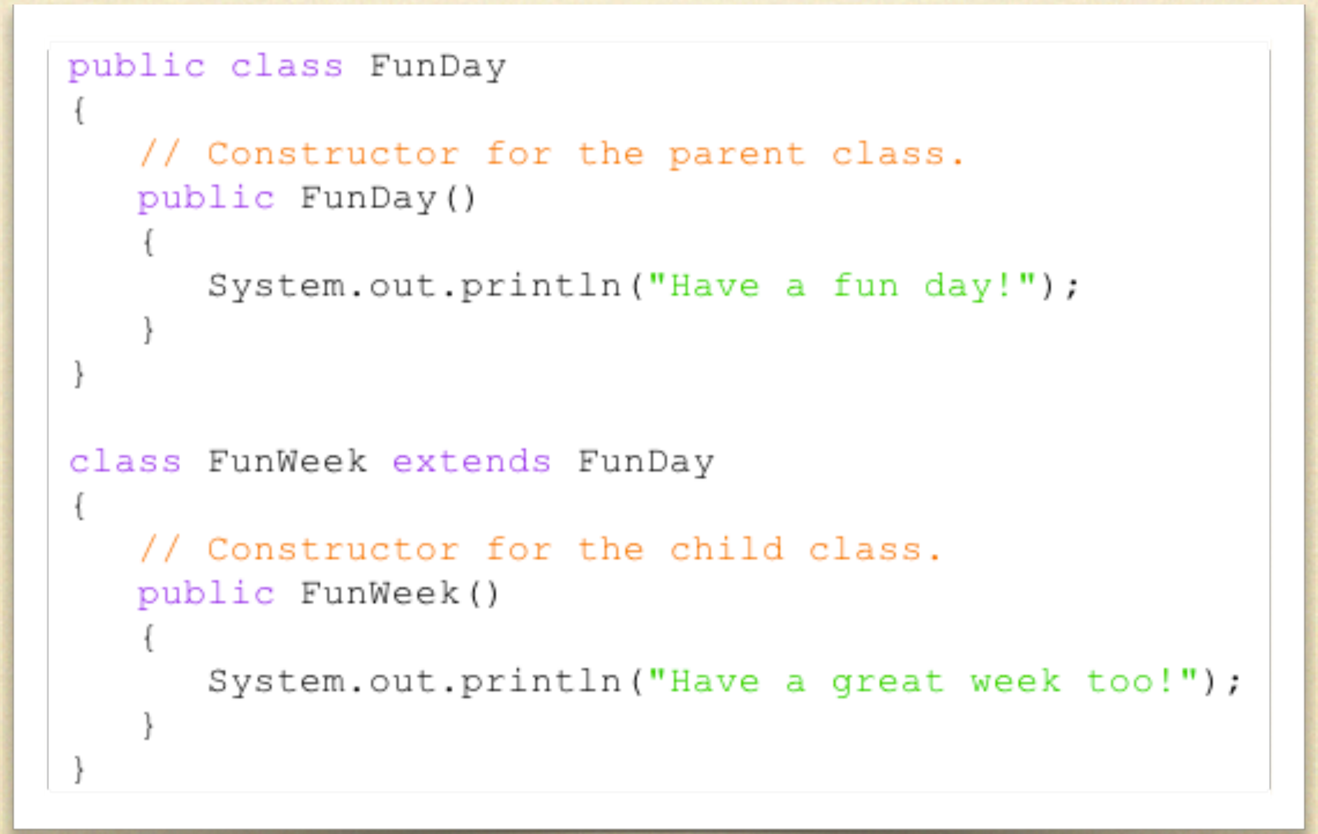
- All of these improvements were made without changing one line of code in the original Book class -- simply by "extending" its abilities.

Inheritance

- A few things that are important to realize when extending a class:
  - The original class is considered the parent or superclass.
  - The class that contains the additional features is called the child or subclass, as it's "extending" the original parent's code.
  - If the parent class contains a constructor that requires arguments, you'll have to pass these arguments on by using the super() method.
  - The parent's constructor always runs first. Once that's finished, the subclass constructor will then proceed.

Constructors

- To demonstrate that the parent's constructor runs first, consider the following example.



```java
public class FunDay
{
   // Constructor for the parent class.
   public FunDay()
   {
      System.out.println("Have a fun day!");
   }
}

class FunWeek extends FunDay
{
   // Constructor for the child class.
   public FunWeek()
   {
      System.out.println("Have a great week too!");
   }
}
```

- When you instantiate FunWeek as an object, the parent's constructor will run first, displaying the words, "Have a fun day!" on the screen.

Protected

- As you learn to use inheritance with your programs, you might encounter a situation where a subclass needs to access one of the superclass's data fields.
- Luckily, Java provides a solution to this -- it's called the protected access specifier. (*Quick note: Access specifiers are also known in the programming world as "access modifiers" too.*)
- Variables that use the "protected" access specifier can be directly accessed and modified by their subclasses. Take a look at the online lesson, as it provides an example showing how it is used.