Recap of OO concepts

Objects, classes, methods and more.

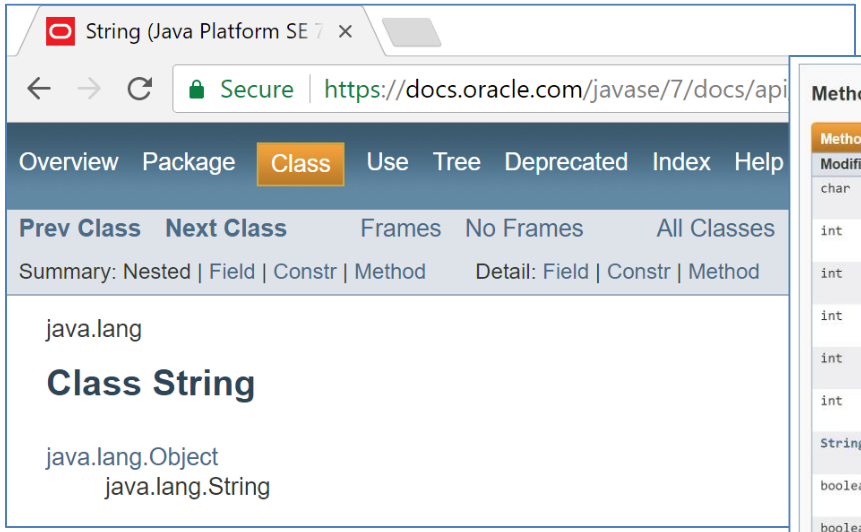
Produced by: Ms. Mairead Meagher
Dr. Siobhán Drohan

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Classes and Objects

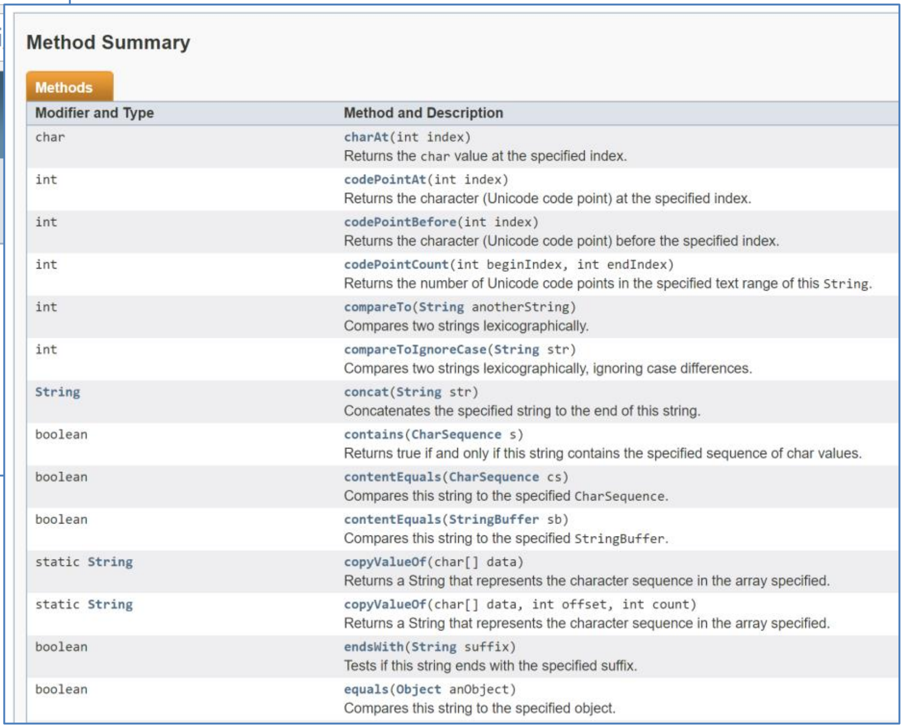
- A **class**
  - defines a group of related **methods** (functions) and **fields** (variables / properties).


String (Java Platform SE 7
Secure
https://docs.oracle.com/javase/7/docs/api
Overview Package Class Use Tree Deprecated Index Help
Prev Class Next Class Frames No Frames All Classes
Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.lang

Class String

java.lang.Object
    java.lang.String



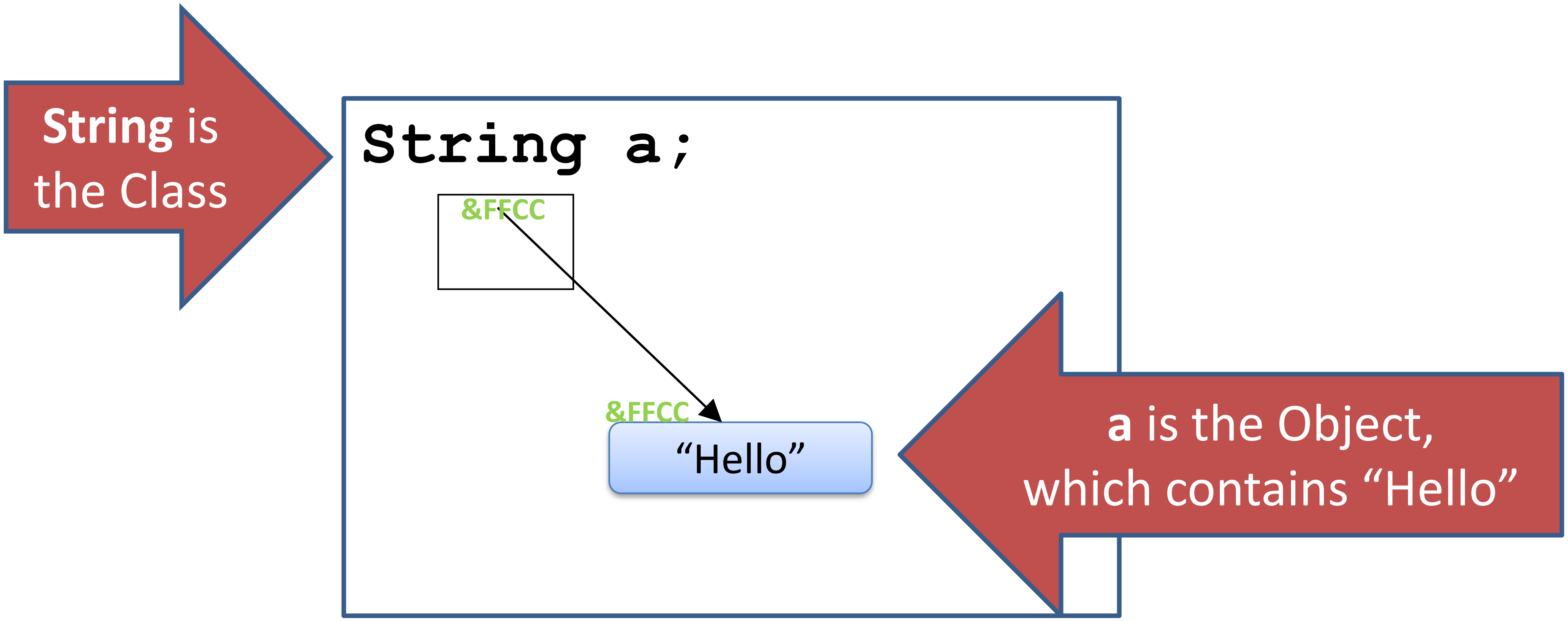
Method Summary

Methods

Modifier and Type	Method and Description
`char`	`charAt(int index)` Returns the `char` value at the specified index.
`int`	`codePointAt(int index)` Returns the character (Unicode code point) at the specified index.
`int`	`codePointBefore(int index)` Returns the character (Unicode code point) before the specified index.
`int`	`codePointCount(int beginIndex, int endIndex)` Returns the number of Unicode code points in the specified text range of this `String`.
`int`	`compareTo(String anotherString)` Compares two strings lexicographically.
`int`	`compareToIgnoreCase(String str)` Compares two strings lexicographically, ignoring case differences.
`String`	`concat(String str)` Concatenates the specified string to the end of this string.
`boolean`	`contains(CharSequence s)` Returns true if and only if this string contains the specified sequence of char values.
`boolean`	`contentEquals(CharSequence cs)` Compares this string to the specified `CharSequence`.
`boolean`	`contentEquals(StringBuffer sb)` Compares this string to the specified `StringBuffer`.
`static String`	`copyValueOf(char[] data)` Returns a String that represents the character sequence in the array specified.
`static String`	`copyValueOf(char[] data, int offset, int count)` Returns a String that represents the character sequence in the array specified.
`boolean`	`endsWith(String suffix)` Tests if this string ends with the specified suffix.
`boolean`	`equals(Object anObject)` Compares this string to the specified object.

Classes and **Objects**

- An **object**
  - is a single instance of a class
  - i.e. an object is created (instantiated) from a class.



Classes and Objects – **Many Objects**

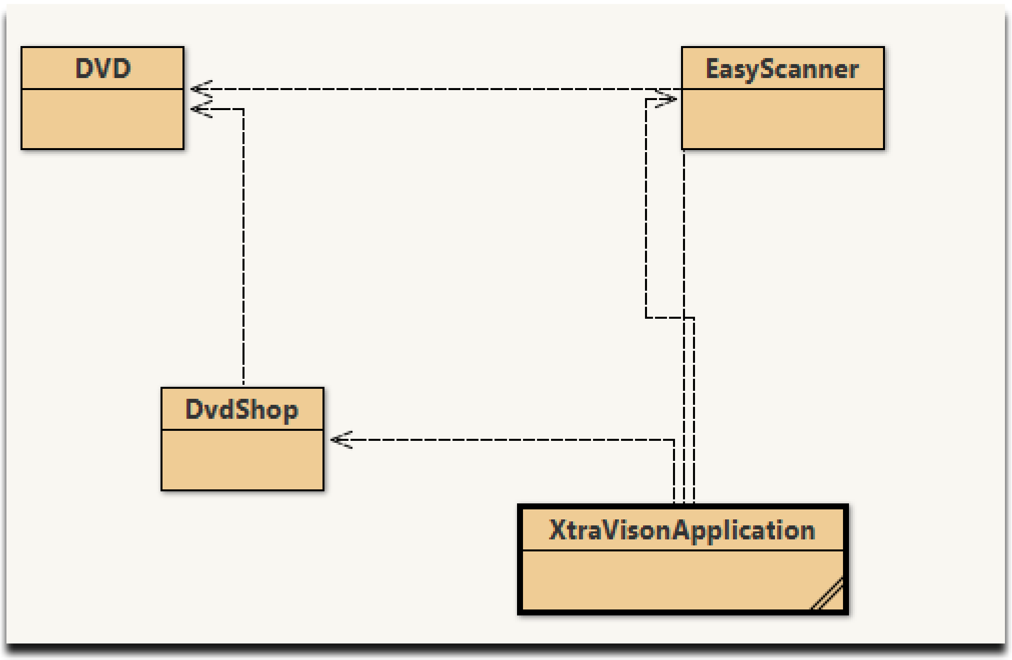
- Many **objects** can be constructed from a single **class** definition.
- Each **object** must have a unique name within the program.

Ver 1.0

XTRAVISION APPLICATION

XtraVisionApplication

- We will recap object oriented concepts through the study of your first year assignment XtraVisionApplication.



DVD

- The DVD class stores **details** about a product
  - id
  - name
  - ageClassification
  - category
  - numMinutes
  - lenofTime
  - rating

DvdShop

DvdShop

- The DvdShop class stores **a collection of dvds and other details about the collection**
  – dvdList
  – total (of dvds)

EasyScanner

- The **EasyScanner** class has the following methods to make reading from keyboard simple:
  - **int** nextInt()
  - **double** nextDouble()
  - **String** nextString**()**
  - **char** nextChar()

EasyScanner

A **DVD** Class… fields

```
private String dvdId;
private String dvdName;
private int ageClassification;
private String category;
private int numMinutes;
private static int lenOfTime;
private int rating;
```

A **DVD** Class… **fields** and **constructor**

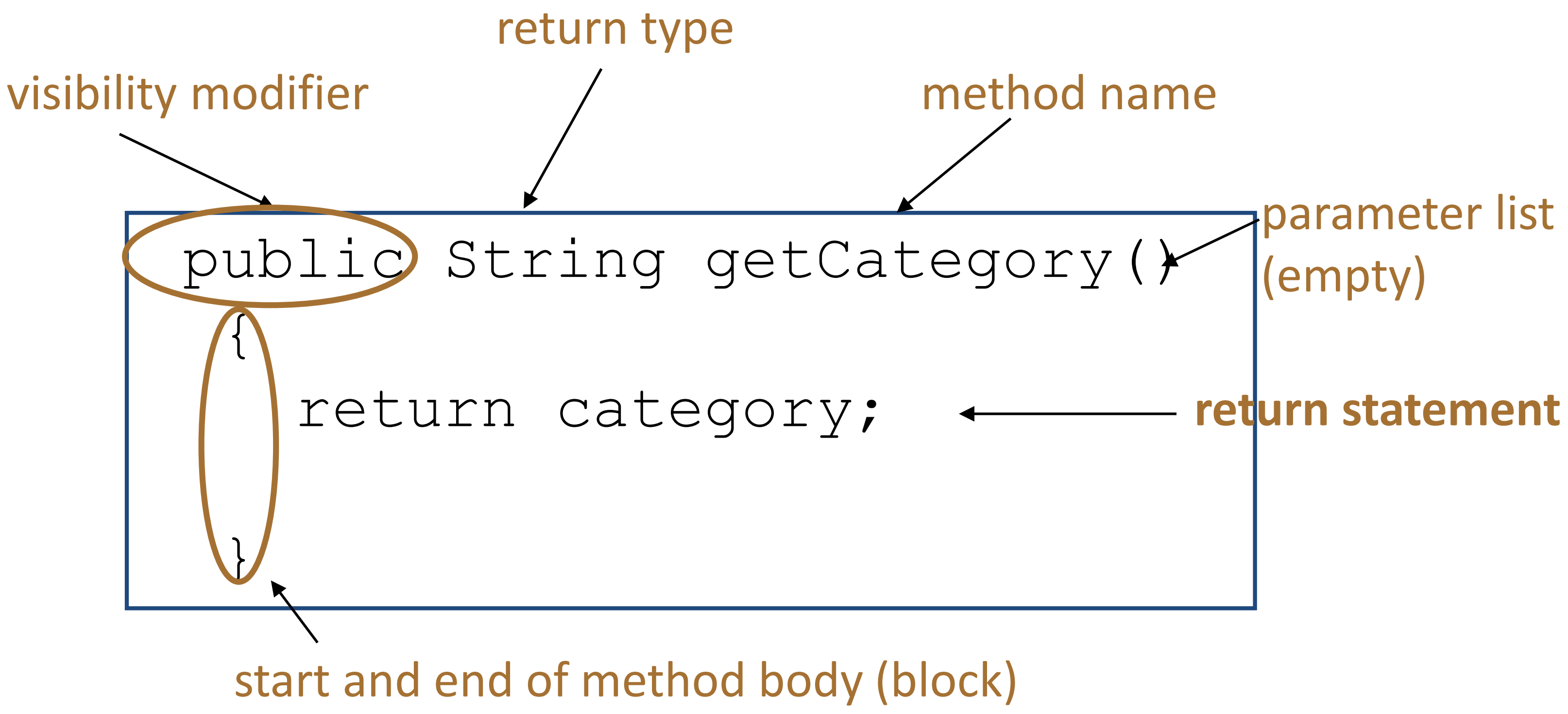
```
public class Class {

    private String dvdId;
        private String dvdName;
        private int ageClassification;
        private String category;
        private int numMinutes;
        private static int lenOfTime;
        private int rating;
 public DVD(String dvdIdIn, String nIn, String categoryIn,
           int ageClassificationIn, int numMinutesIn)
      {
         dvdId = dvdIdIn;
         dvdName = nIn;
         category = categoryIn;
         ageClassification = ageClassificationIn;
         numMinutes = numMinutesIn;
         rating = 0;                 //rating will be decided later
         lenOfTime = 0;       //lenOfTime will be changed later
```

Getters (Accessor Methods)

- **Accessor** methods
  - return information about the **state** of an object
    - i.e. **the values stored in the fields**.

- A 'getter' method
  - is a specific type of **accessor** method and typically:
    - **contains a return statement**
      (as the last executable statement in the method).
    - defines a **return type**.
    - **does NOT change the object state**.

Getters



```
public String getCategory()
{
    return category;
}
```

A Product Class...**get**ters

```
public String getDvdId()
       {
            return dvdId;
       }

public String getDvdName()
          {
             return dvdName ;

        }
 public int getAgeClassification()
        {
          return ageClassification;

        }
public String getCategory()
        {
             return category;

        }
```

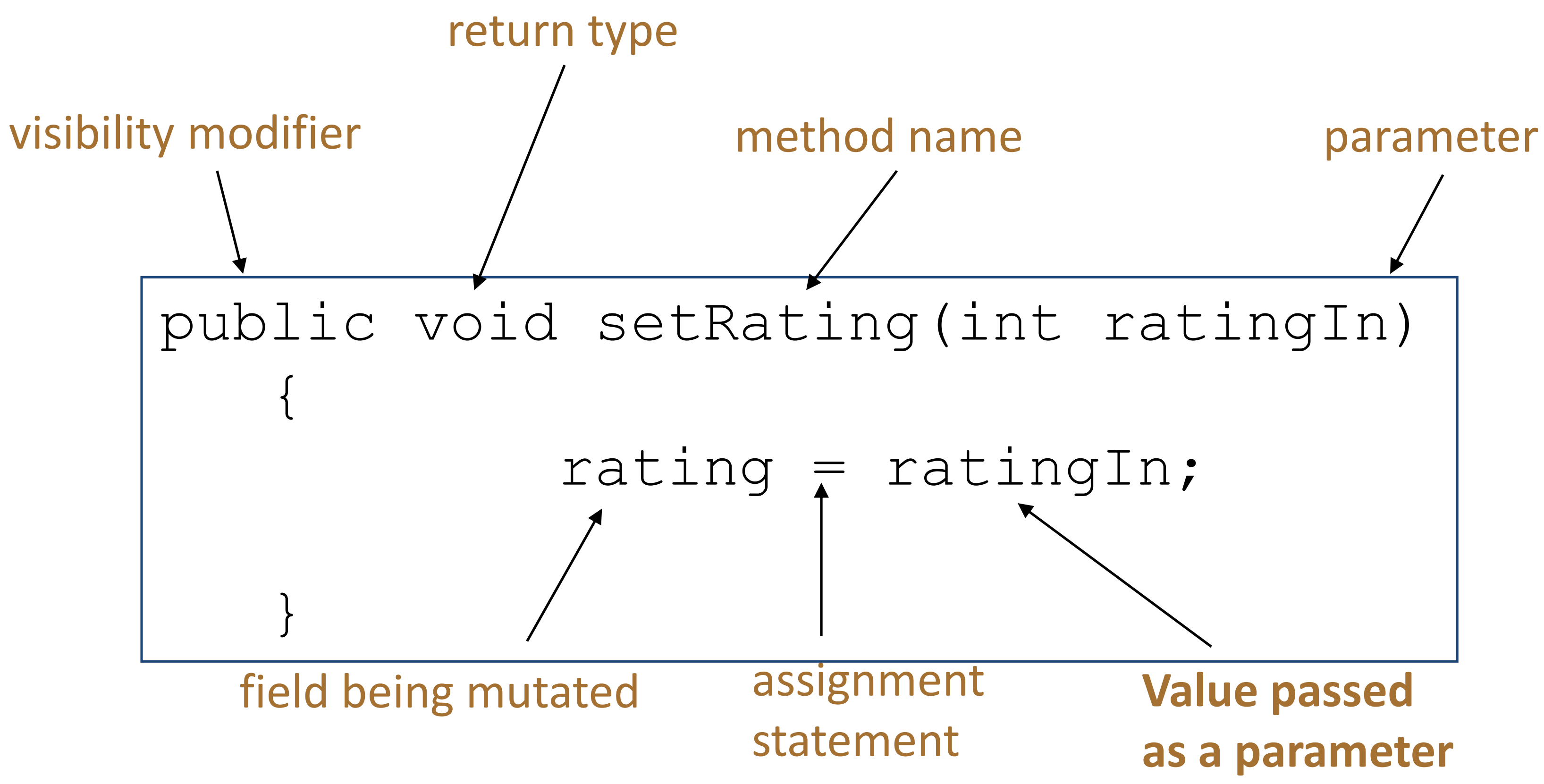
```
public int getNumMinutes()
        {
           return numMinutes;
        }
public static int getLenOfTime()
        {
            return lenOfTime;
        }
```

Setters (Mutator methods)

- **Mutator** methods
  - change (i.e. mutate!) an object's state.

- A 'setter' method
  - is a specific type of **mutator** method and typically:
    - contains an **assignment statement**
    - takes in a **parameter**
    - **changes the object state**.

Setters



```
public void setRating(int ratingIn)
    {
            rating = ratingIn;


    }
```

A DVD Class…setters

```
public void setDvdId(String dvdIdIn )
      {
         dvdId = dvdIdIn;
      }
public void setDvdName(String dvdNameIn)
      {
         dvdName = dvdNameIn;
      }
public void setAgeClassification(int ageClassificationIn)
       {
            ageClassification = ageClassificationIn;

       }

public void setCategory(String categoryIn)
       {
            category = categoryIn;

       }
```

A DVD Class…setters contd

```
 public void setRating(int ratingIn)
        {
             rating = ratingIn;

        }
public void setNumMinutes(int numMinutesIn)
        {
             numMinutes = numMinutesIn;

        }
 public  static void setLenOfTime()
        {
                lenOfTime = lenOfTime + 1;
        }
```

toString()

- toString() is a method that returns a string version of an object. E.g.

```
public String toString() {
        return ("DVD ID: " + dvdId  + "  Name : " + dvdName +
              "\nAgeClassification : " + ageClassification +
              "  Category : " + category + "  Rating : " + rating +
              "\nLength of running time :" + numMinutes +
              "\nNumber of years in store: " + lenOfTime );
           }
```

- This can then be re-used to print details

Getters/Setters

- For each instance field in a class, you are normally asked to write:
  - A **get**ter
    - Return statement
  - A **set**ter
    - Assignment statement

Encapsulation in Java – **steps 1-3**

Encapsulation Step	Approach in Java

1. **Wrap** the data (fields) and code acting on the data (methods) together as single unit.	`public class ClassName {`
	 `    Fields`
	 `    Constructors`
	 `    Methods (getters, setters, toString(), other methods )`
	 `}`

2. **Hide** the fields from other classes.	**Declare the fields of a class as private.**
3. **Access** the fields only through the methods of their current class.	**Provide public setter and getter** methods to modify and view the fields values.

http://www.tutorialspoint.com/java/java_encapsulation.htm

A DVD Class… **An Encapsulated Class**

Fields are **hidden** from other classes.

1. DVD class **wraps** the data (fields) and code acting on the data (methods) together as **single unit**.

```
public class DVD
 {
     private String dvdId;
            : fields

     public DVD ( …) { .. }
     public other methods
            :
 }
```

Methods are **available** from other classes.

Using the DVD Class

1

```
private DVD product;
```

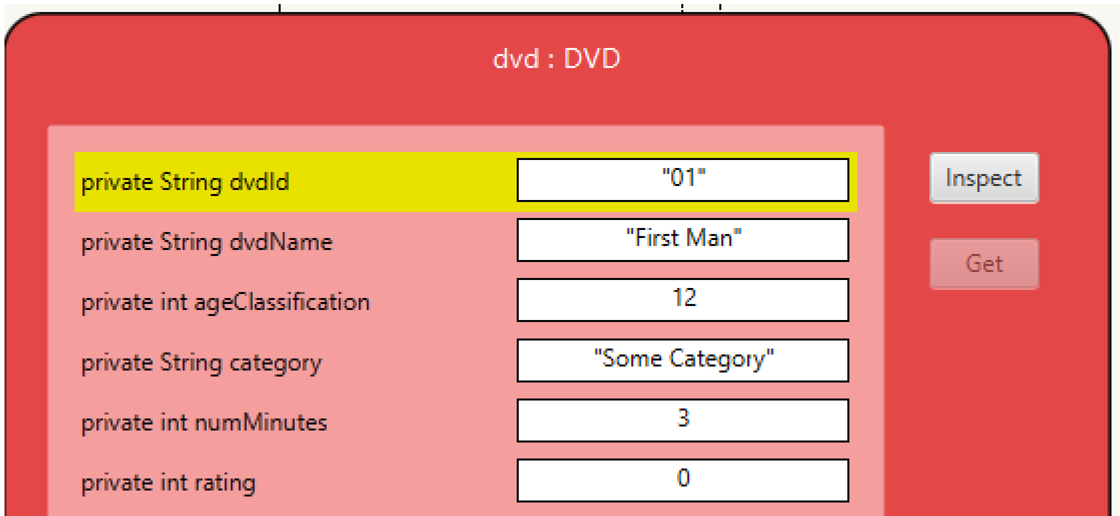
Declaring an object **dvd**, of type **DVD**.

dvd

2

```
dvd = new DVD("01","First Man", 12, "Some Category", 105, 3);
```

Calls the **DVD** *constructor* to build the **dvd** object in memory.

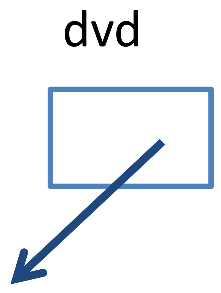


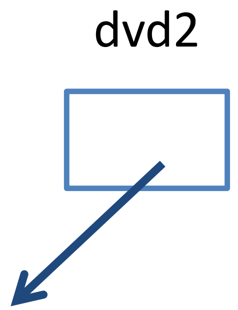
private String dvdId	"01"
private String dvdName	"First Man"
private int ageClassification	12
private String category	"Some Category"
private int numMinutes	3
private int rating	0

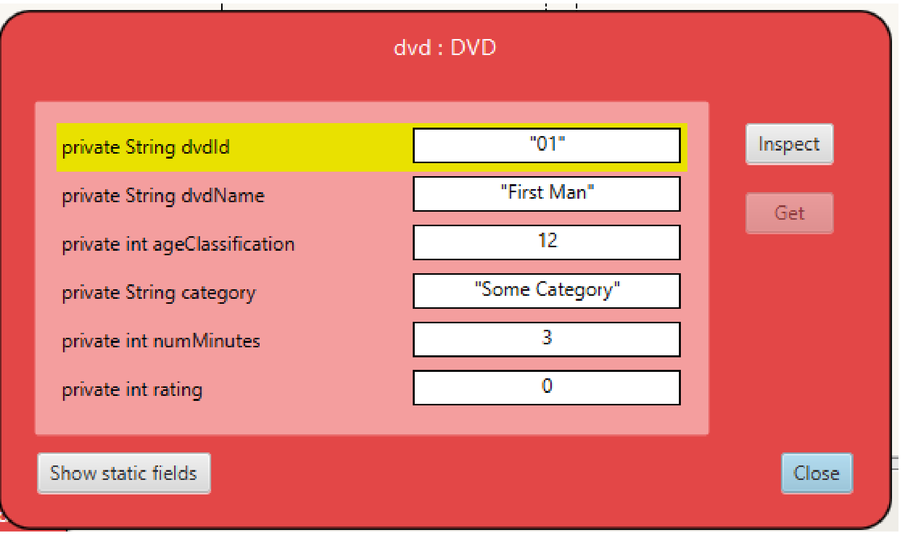
Multiple Product Objects

```
dvd = new DVD("01","First Man", 12, "Some Category", 12, 3);
```

```
dvd2 = new DVD("02","Avengers Endgame", 15, "SuperHeroes", 15, 4);
```

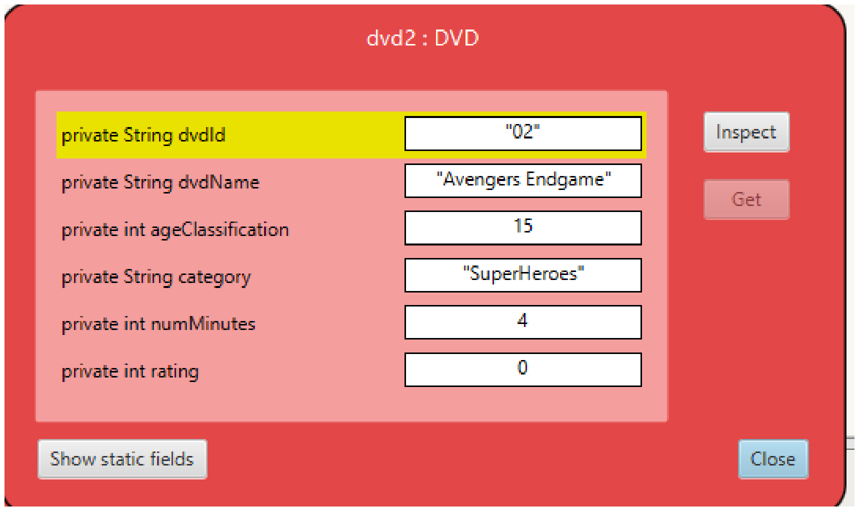






dvd : DVD

private String dvdId	"01"
private String dvdName	"First Man"
private int ageClassification	12
private String category	"Some Category"
private int numMinutes	3
private int rating	0

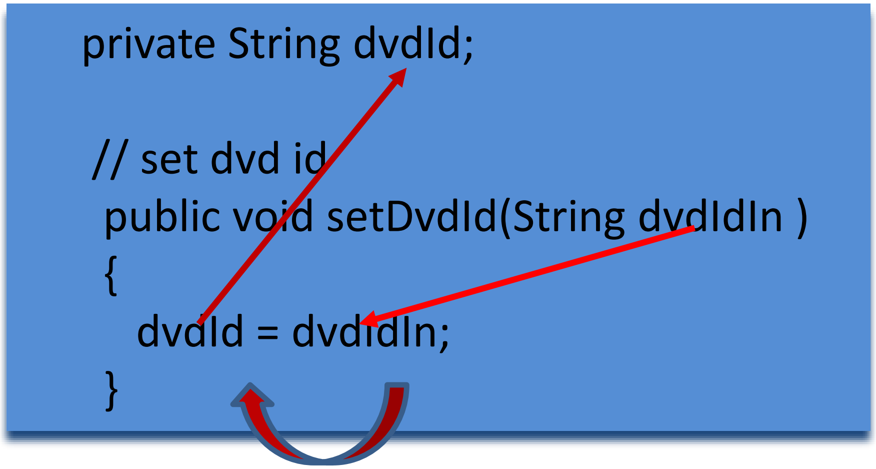


dvd2 : DVD

private String dvdId	"02"
private String dvdName	"Avengers Endgame"
private int ageClassification	15
private String category	"SuperHeroes"
private int numMinutes	4
private int rating	0

Use of the this keyword.

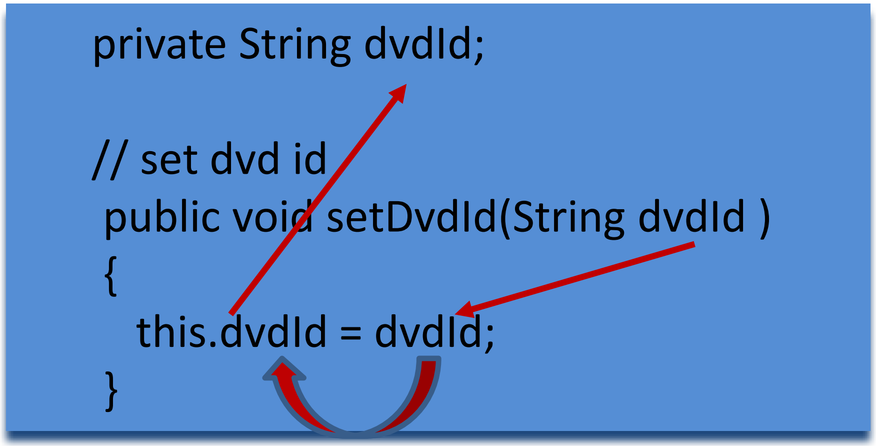
- Before



```
private String dvdId;

// set dvd id
 public void setDvdId(String dvdIdIn )
 {
    dvdId = dvdIdIn;
 }
```

- After

The this. construct always relates to the field variable



```
private String dvdId;

// set dvd id
 public void setDvdId(String dvdId )
 {
    this.dvdId = dvdId;
 }
```

Validation on fields

- Helps ensure lack of invalid data
- E.g. ‘age classifications should be between 12 and 18)

```
if (ageClassification >= 12 && ageClassification <=18)
        this.ageClassification = ageClassification;
else
        this.ageClassification = 18;
```

- We use this in the constructors and setters (and any time we are changing fields)

Questions?