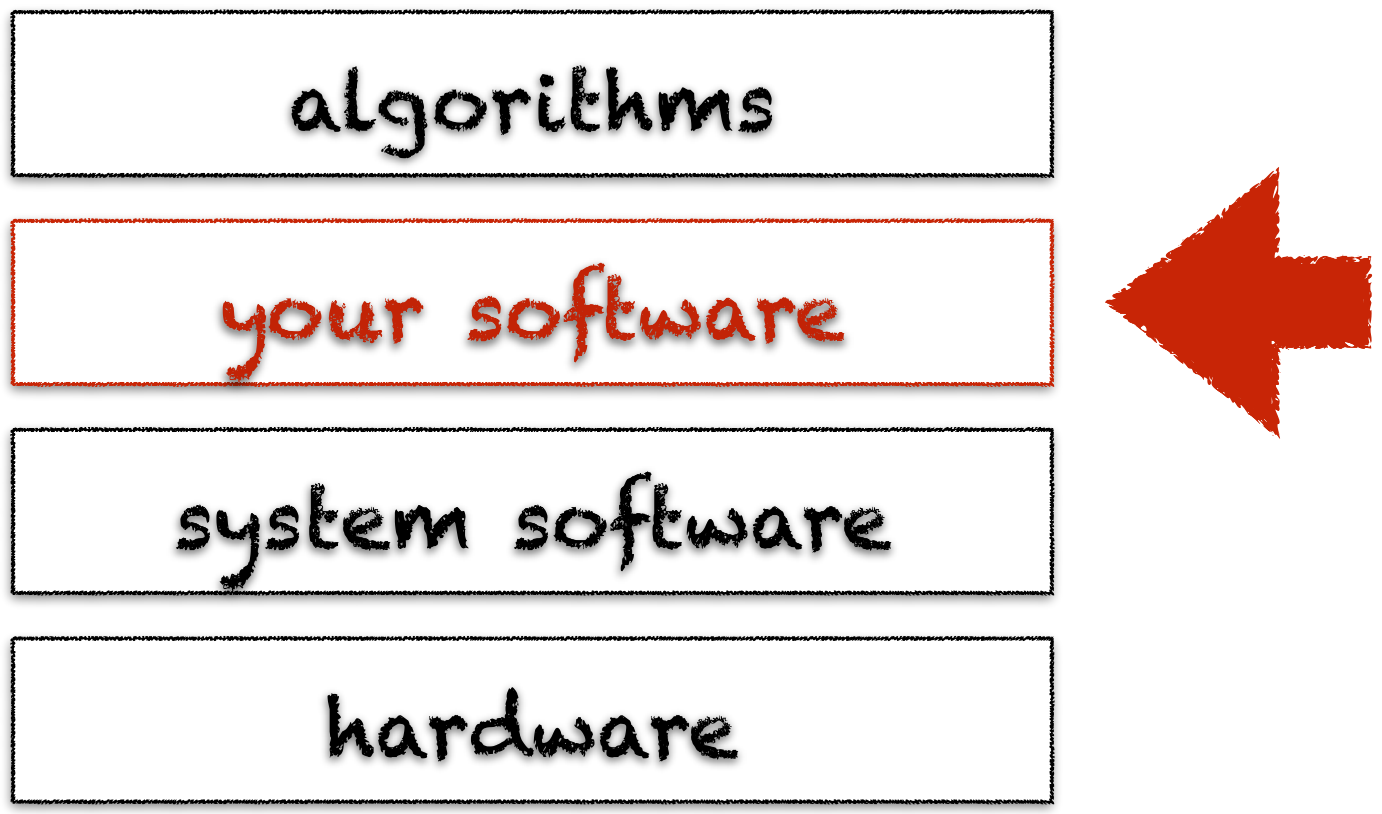
inheritance & polymorphism

learning objectives



- learn how to factor code thanks to inheritance
- learn how to override constructors and methods
- learn about subclassing, subtyping & polymorphism

a vintage example

catalog of books & vinyls

title
author
rating

title
artist
duration
rating

list of books
list of vinyls

a vintage example

catalog of books & vinyls

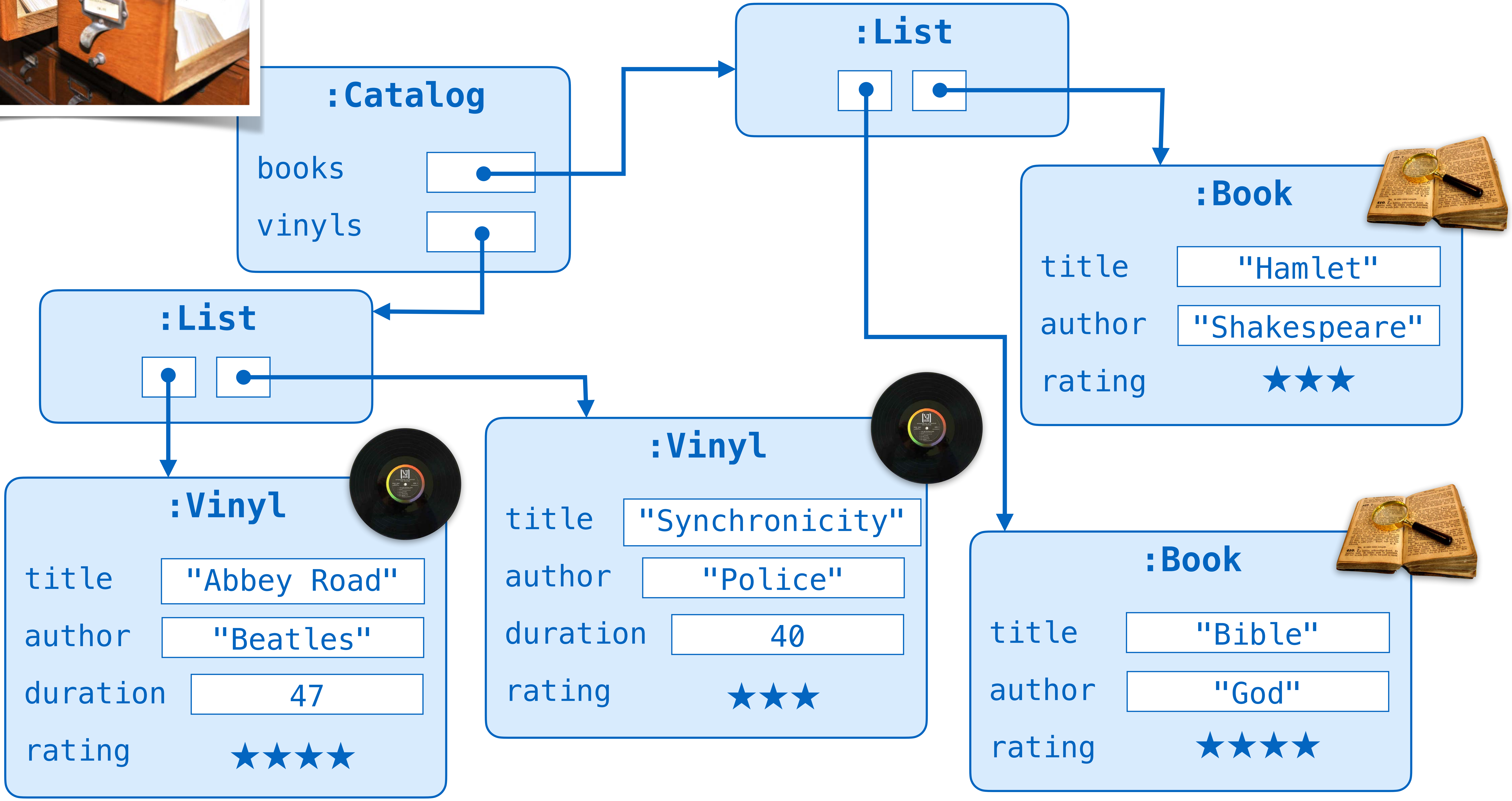
Book
title author rating
init rate printInfo

Vinyl
title artist duration rating
init rate printInfo

Catalog
books vinyls
addBook addVinyl listInfo

a vintage example

catalog of books & vinyls



catalog of books & vinyls

```java
public class Book {

    private final String title;
    private final String author;
    private int rating;

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
        this.rating = -1;
    }
    public void rate(int rating) {
        this.rating = rating;
    }
    public void printInto() {
        String rating = "?";

        if (this.rating >= 0) {
            rating = "";
            for (int i = 0; i < this.rating; i++) {
                rating = rating + "*";
            }
        }
        System.out.println(
              "Book[title: " + this.title + " | author: " + this.author + " | rating: " + rating + "]");
    }
}
```

catalog of books & vinyls

```
public class Vinyl {

    private final String title;
    private final String artist;
    private final int duration;
    private int rating;

    public Vinyl(String title, String artist, int duration) {
        this.title = title;
        this.artist = artist;
        this.duration = duration;
        this.rating = -1;
    }
    public void rate(int rating) {
        this.rating = rating;
    }
    public void printInto() {
        String rating = "?";

        if (this.rating >= 0) {
            rating = "";
            for (int i = 0; i < this.rating; i++) {
                rating = rating + "*";
            }
        }
        System.out.println("Vinyl[title: " + this.title + " | artist: " + this.artist +
                           " | duration: " + this.duration + " minutes | rating: " + rating + "]");
    }
}
```

catalog of books & vinyls

```java
public class Catalog {

    private final List<Book> books = new ArrayList<Book>();
    private final List<Vinyl> vinyls = new ArrayList<Vinyl>();

    public void addBook(Book book) {
        this.books.add(book);
    }
    public void addVinyl(Vinyl vinyl) {
        this.vinyls.add(vinyl);
    }
    public void listInfo() {
        System.out.println("BOOKS");

        for (Book book : books) {
            book.printInfo();
        }

        System.out.println("------");
        System.out.println("VINYLS");
        for (Vinyl vinyl : vinyls) {
            vinyl.printInfo();
        }
    }
}
```

```java
Catalog c = new Catalog();

Book b1 = new Book("Bible", "God");
b1.rate(3);
Book b2 = new Book("Hamlet", "Shakespeare");

c.addBook(b1);
c.addBook(b2);

Vinyl v1 = new Vinyl("Abbey Road", "Beatles", 47);
Vinyl v2 = new Vinyl("Synchronicity", "Police", 40);

c.addVinyl(v1);
c.addVinyl(v2);

c.listInfo();
```

```
BOOKS
Book[title: Bible | author: God | rating: ***]
Book[title: Hamlet | author: Shakespeare | rating: ?]
------
VINYLS
Vinyl[title: Abbey Road | artist: Beatles | duration: 47 minutes | rating: ?]
Vinyl[title: Synchronicity | artist: Police | duration: 40 minutes | rating: ?]
```

catalog of books & vinyls

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.rating = -1

    def rate(self, rating):
        self.rating = rating

    def printInfo(self):
        rating = "?"
        if self.rating >= 0:
            rating = ""
            for i in range(0, self.rating):
                rating = rating + "*"

        print("Book[title: {0} | author: {1} | rating: {2}]"
              .format(self.title, self.author, rating))
```

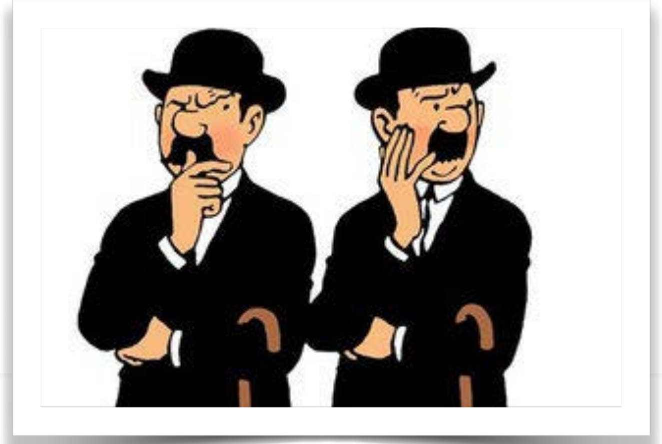
```
class Vinyl:
    def __init__(self, title, artist, duration):
        self.title = title
        self.artist = artist
        self.duration = duration
        self.rating = -1

    def rate(self, rating):
        self.rating = rating

    def printInfo(self):
        rating = "?"
        if self.rating >= 0:
            rating = ""
            for i in range(0, self.rating):
                rating = rating + "*"

        print("Vinyl[title: {0} | artist: {1} | duration: {2} minutes | rating: {3}]"
              .format(self.title, self.artist, self.duration, rating))
```

catalog of books & vinyls

```
class Catalog:
    def __init__(self):
        self.books = []
        self.vinyls = []

    def addBook(self, book):
        self.books.append(book)

    def addVinyl(self, vinyl):
        self.vinyls.append(vinyl)

    def listInfo(self):
        print("BOOKS")
        for b in self.books:
            b.printInfo()
        print("------")
        print("VINYLS")
        for v in self.vinyls:
            v.printInfo()
```

```
c = Catalog()

b1 = Book("Bible", "God")
b1.rate(3)
b2 = Book("Hamlet", "Shakespeare")

c.addBook(b1)
c.addBook(b2)

v1 = Vinyl("Abbey Road", "Beatles", 47)
v2 = Vinyl("Synchronicity", "Police", 40)

c.addVinyl(v1)
c.addVinyl(v2)

c.listInfo()
```

```
BOOKS
Book[title: Bible | author: God | rating: ***]
Book[title: Hamlet | author: Shakespeare | rating: ?]
------
VINYLS
Vinyl[title: Abbey Road | artist: Beatles | duration: 47 minutes | rating: ?]
Vinyl[title: Synchronicity | artist: Police | duration: 40 minutes | rating: ?]
```

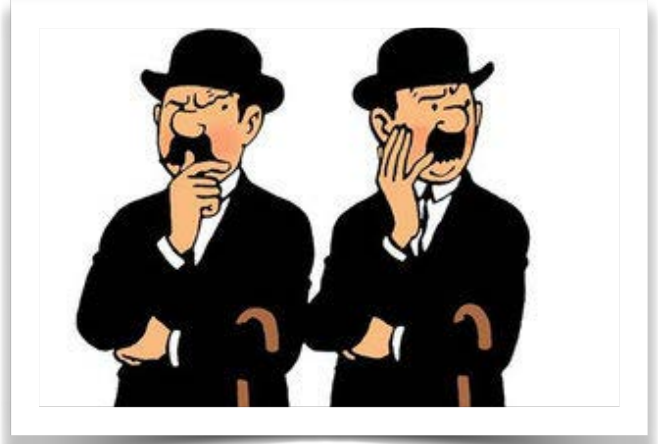


what's wrong?

```java
public class Vinyl {
    private final String title;
    private final String artist;
    private final int duration;
    private int rating;

    public Vinyl(String title, String artist, int duration) {
        this.title = title;
        this.artist = artist;
        this.duration = duration;
        this.rating = -1;
    }

    public void rate(int rating) {
        this.rating = rating;
    }

    public void printInto() {
        String rating = "?";

        if (this.rating >= 0) {
            rating = "";
            for (int i = 0; i < this.rating; i++) {
                rating = rating + "*";
            }
        }
        System.out.println("Vinyl[title: " + this.title + " | arti
    }
}
```

```java
public class Book {
    private final String title;
    private final String author;
    private int rating;

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
        this.rating = -1;
    }

    public void rate(int rating) {
        this.rating = rating;
    }

    public void printInto() {
        String rating = "?";

        if (this.rating >= 0) {
            rating = "";
            for (int i = 0; i < this.rating; i++) {
                rating = rating + "*";
            }
        }
        System.out.println("Book[title: " + this.title + " | aut
    }
}
```

the Vinyl and Book classes are very similar → code duplication



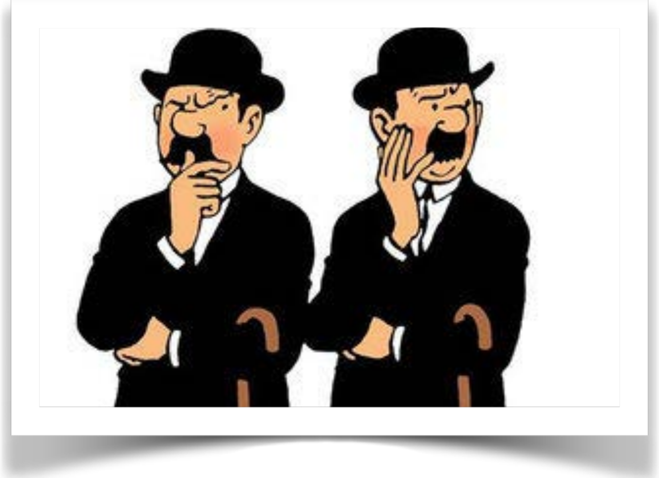
code duplication

...also in the Catalog class

```java
public class Catalog {

    private final List<Book> books = new ArrayList<Book>();
    private final List<Vinyl> vinyls = new ArrayList<Vinyl>();

    public void addBook(Book book) {
        this.books.add(book);
    }

    public void addVinyl(Vinyl vinyl) {
        this.vinyls.add(vinyl);
    }

    public void listInfo() {
        System.out.println("BOOKS");

        for (Book book : books) {
            book.printInfo();
        }

        System.out.println("------");
        System.out.println("VINYLS");

        for (Vinyl vinyl : vinyls) {
            vinyl.printInfo();
        }
    }
}
```

maintenance is made difficult

great danger of introducing bugs when evolving



great danger of introducing bugs when evolving

assume we want to support games in v2.0

create a Game class

with many similarities to the Book and Vinyl classes

add a games field in Catalog

initialize it in the constructor in Catalog

add an addGame() method in Catalog

add a loop in the listInfo() method of Catalog

solution: inheritance

inheritance

allows us to define one class, the subclass, as an extension of another, the superclass

inheritance

superclass

defines common attributes

subclasses

inherit all fields and

methods from its superclass

define specific attributes

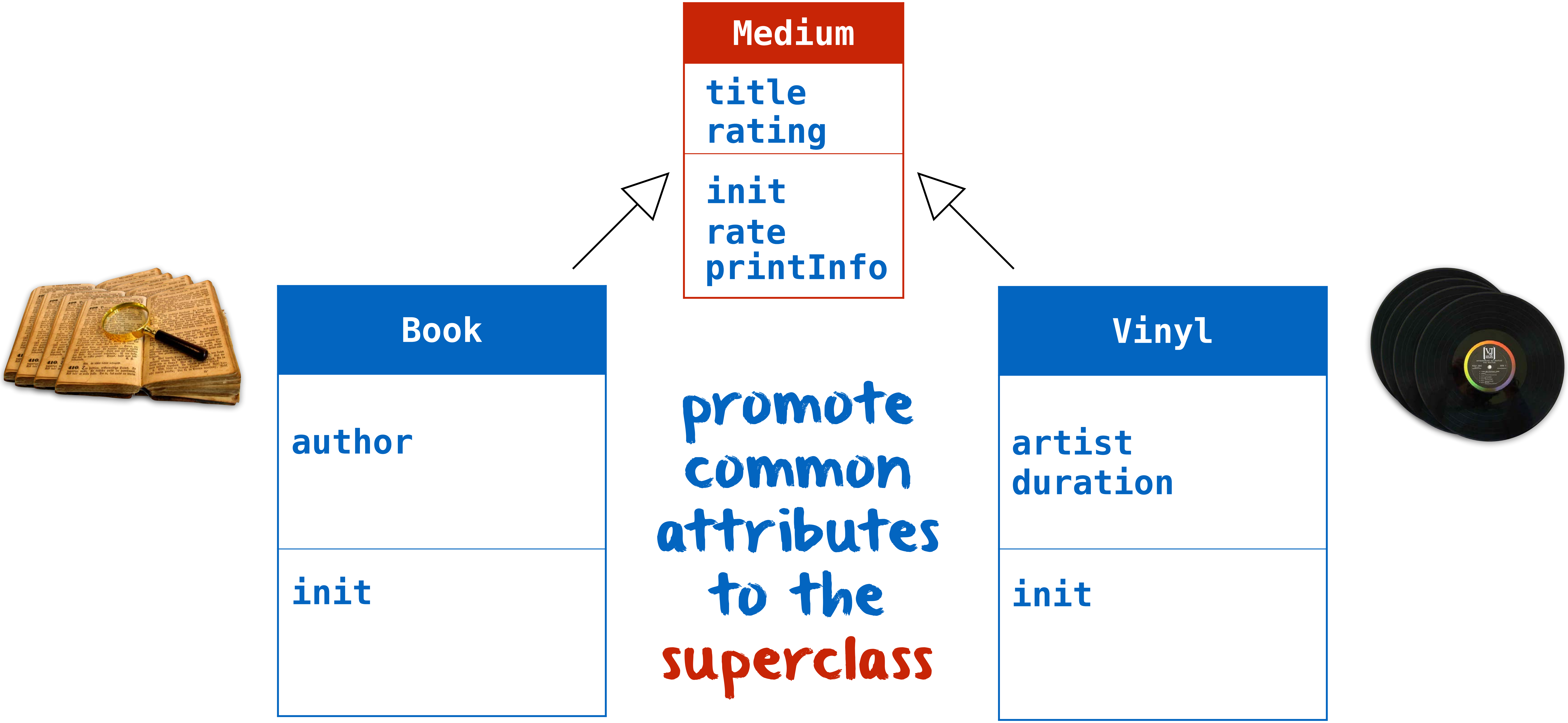
inheritance

common attributes

Book
title author rating
init rate printInfo

Vinyl
title artist duration rating
init rate printInfo

inheritance



inheritance

Book
- author
- init

Medium
- title
- rating
- init
- rate
- printInfo

Vinyl
- artist
- duration
- init

promote common attributes to the superclass

subclasses **inherit common attributes** from the superclass and **define their specific attributes**

inheritance

```java
public class Medium {

    private final String title;
    private int rating;

    public Medium(String title) {
        this.title = title;
        this.rating = -1;
    }

    public void rate(int rating) {
        this.rating = rating;
    }

    public void printInfo() {
        String rating = "?";

        if (this.rating >= 0) {
            rating = "";
            for (int i = 0; i < this.rating; i++) {
                rating = rating + "*";
            }
        }
        System.out.println("Medium[title: " + this.title + " | rating: " + rating + "]");
    }
}
```

```java
public class Book extends Medium {

    private final String author;

    public Book(String title, String author) {
        super(title);
        this.author = author;
    }
}
```

```java
public class Vinyl extends Medium {

    private final String artist;
    private final int duration;

    public Vinyl(String title, String artist, int duration) {
        super(title);
        this.artist = artist;
        this.duration = duration;
    }
}
```

```java
public class Catalog {

    private final List<Medium> media = new ArrayList<Medium>();

    public void addMedium(Medium medium) {
        this.media.add(medium);
    }
    public void listInfo() {
        System.out.println("MEDIA");

        for (Medium medium : media) {
            medium.printInfo();
        }
    }
}
```

```java
Catalog c = new Catalog();
Book b = new Book("Bible", "God");
Vinyl v = new Vinyl("Abbey Road", "Beatles", 47);
b.rate(3);

c.addMedium(b);
c.addMedium(v);

c.listInfo();
```

inheritance

```
class Medium:
    def __init__(self, title):
        self.title = title
        self.rating = -1

    def rate(self, rating):
        self.rating = rating

    def printInfo(self):
        rating = "?"
        if self.rating >= 0:
            rating = ""
            for i in range(0, self.rating):
                rating = rating + "*"

        print("Medium[title: {0} | rating: {1}]".format(self.title, rating))
```

```
class Book(Medium):
    def __init__(self, title, author):
        self.author = author
        super().__init__(title)
```

```
class Vinyl(Medium):
    def __init__(self, title, artist, duration):
        self.artist = artist
        self.duration = duration
        super().__init__(title)
```

```
class Catalog:
    def __init__(self):
        self.media = []

    def addMedium(self, medium):
        self.media.append(medium)

    def listInfo(self):
        print("MEDIA")
        for m in self.media:
            m.printInfo()
```

```
c = Catalog()
b = Book("Hamlet", "Shakespeare")
v = Vinyl("Abbey Road", "Beatles", 47)
b.rate(3)

c.addMedium(b)
c.addMedium(v)

c.listInfo()
```

calling the superclass constructor

this is necessary to initialize the fields inherited from the superclass

```java
public class Book extends Medium {

    private final String author;

    public Book(String title, String author) {
        super(title);
        this.author = author;
    }
}
```

```python
class Vinyl(Medium):
    def __init__(self, title, artist, duration):
        self.artist = artist
        self.duration = duration
        super().__init__(title)
```

subtyping

before

```
public class Catalog {
  ...
  public void addBook(Book book) { ... }
  public void addVinyl(Vinyl vinyl) { ... }
  ...
}
```

after

```
public class Catalog {
  ...
  public void addMedium(Medium medium) { ... }

  ...
}
```

```
Catalog c = new Catalog();
Book b = new Book("Bible", "God");
Vinyl v = new Vinyl("Abbey Road", "Beatles", 47);
b.rate(3);

c.addMedium(b);
c.addMedium(v);
```

classes define types

a geek

a person with a devotion to something in a way that places him or her outside the mainstream*

*wikipedia

subclasses define subtypes

substitution principle

objects of subtypes can be used where objects of supertypes are required

LINUX GEEK

CODE GEEK

KISS GEEK

polymorphic variables

object variables are polymorphic because they can hold objects of more than one type

they can hold objects of the declared type (Medium), or of subtypes of the declared type (Book, Vinyl)

type casting

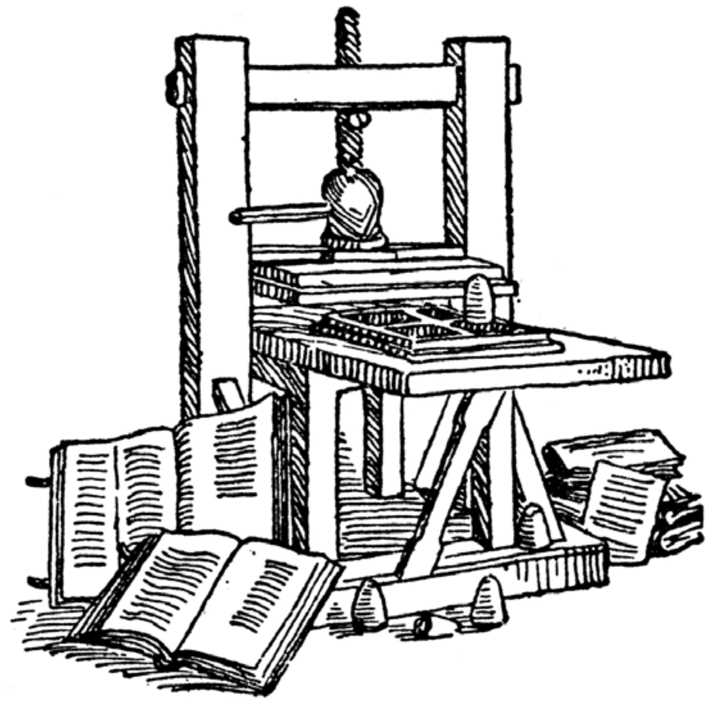
```
Book b = new Book("Hamlet", "Shakespeare");
Vinyl v = new Vinyl("Abbey Road", "Beatles", 47);
Medium m;
v = b;

b = v;

m = v;

m = b;

b = m;
```

the substitution principle

only works…

```
b = (Book) m;
```

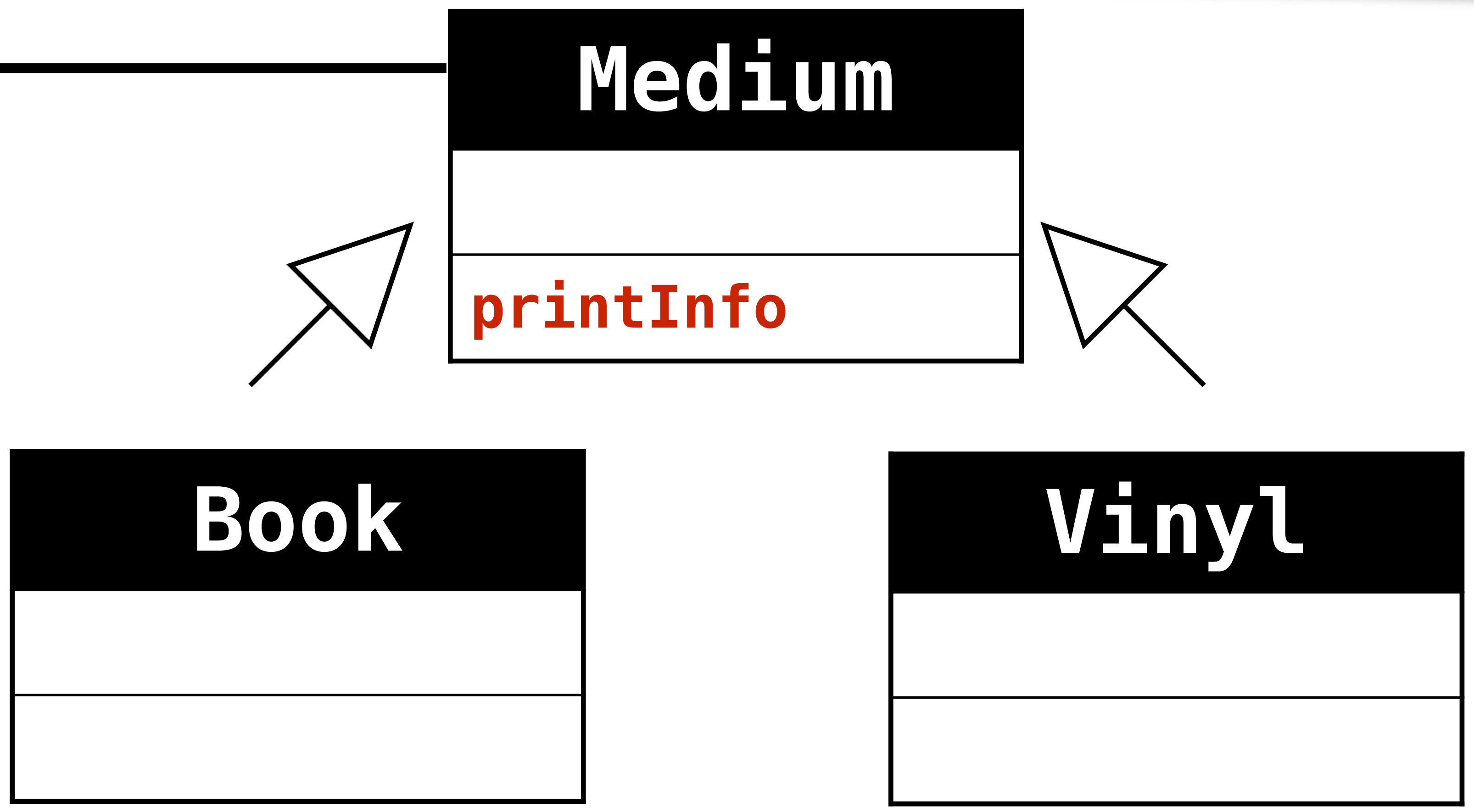
back to printing books and vinyls



```
public void listInfo() {
  System.out.println("MEDIA");
  for (Medium medium : media) {
    medium.printInfo();
  }
}
```

Catalog

```
public void printInfo() {
  String rating = "?";

  if (this.rating >= 0) {
    rating = "";
    for (int i = 0; i < this.rating; i++) {
      rating = rating + "*";
    }
  }
  System.out.println("Medium[title: " + this.title + " | rating: " + rating + "]");
}
```




```
Medium[title: Bible | rating: ***]
Medium[title: Synchronicity | rating: ?]
```

the printInfo method in Medium only prints the common fields

the problem

inheritance is a one-way street: a subclass inherits the superclass fields

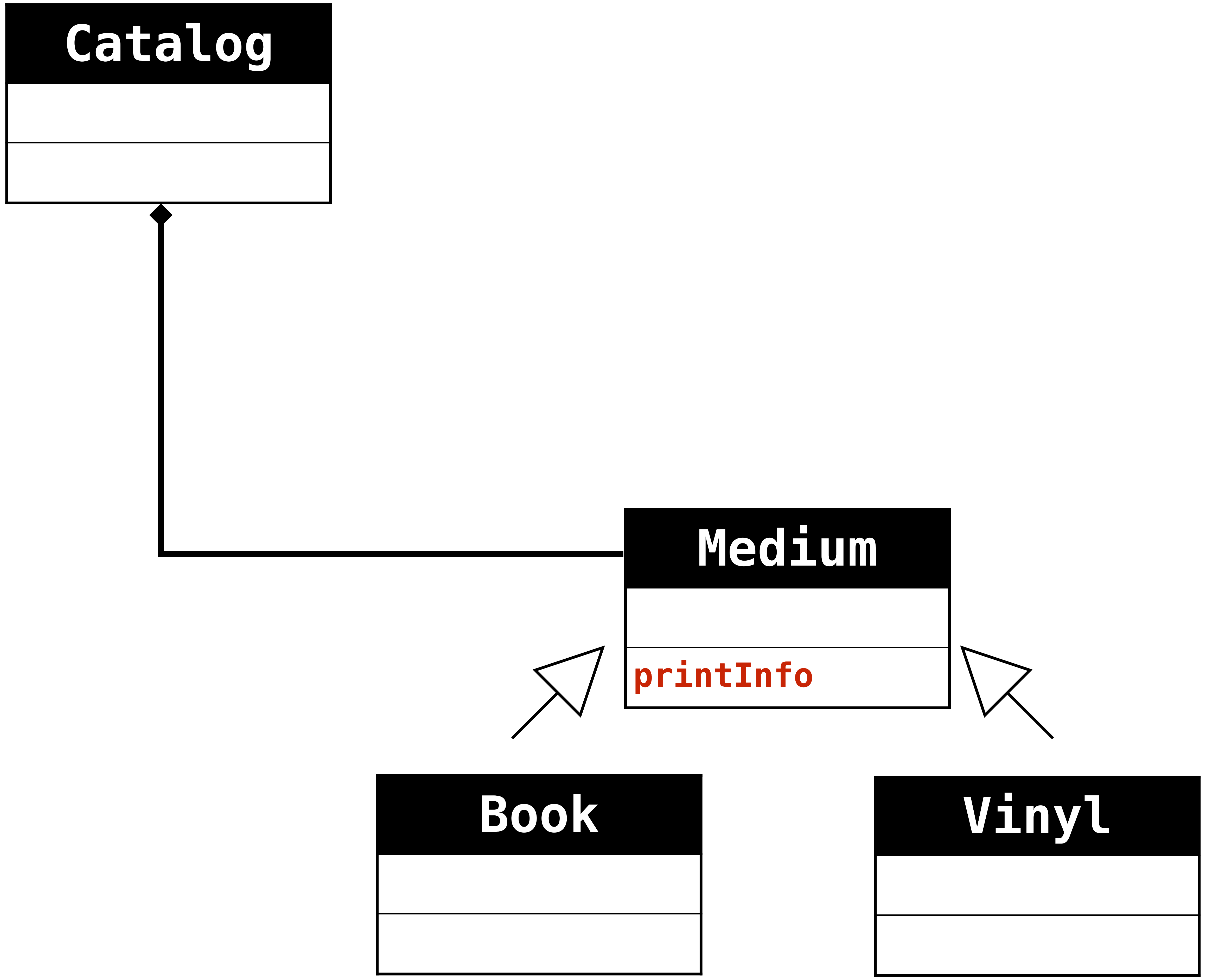
the superclass knows nothing about the fields of its subclasses

try to

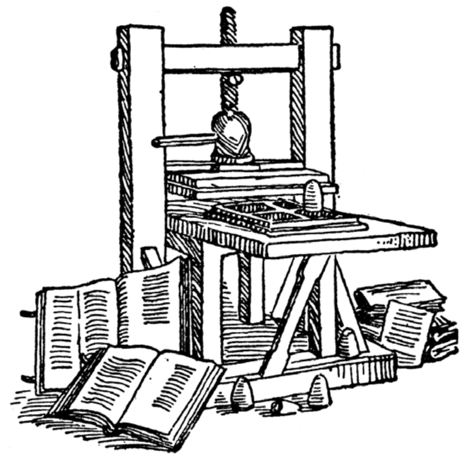
move `printInfo` down to `Book` and `Vinyl`

printing books & vinyls

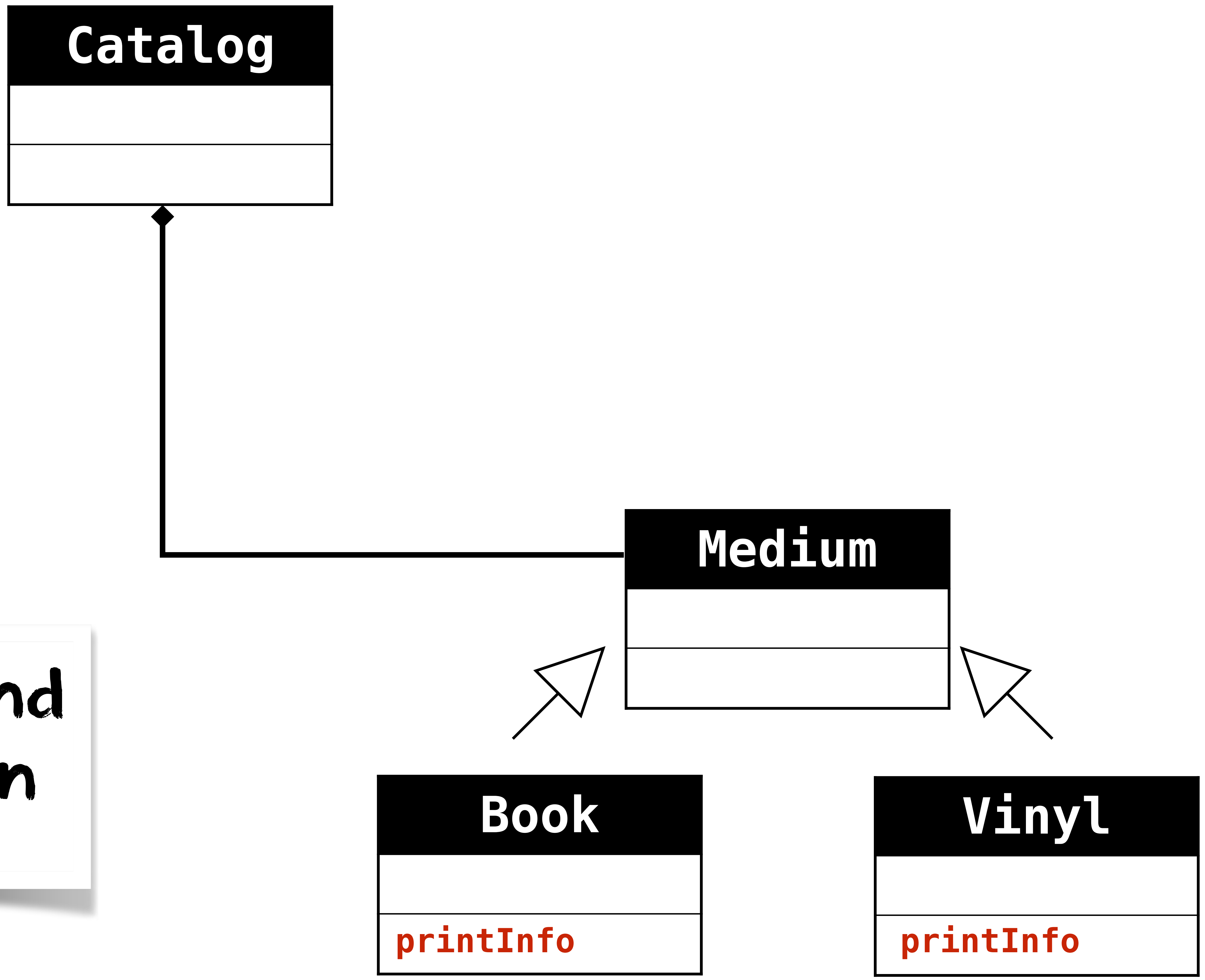
```
public void listInfo() {
  System.out.println("MEDIA");
  for (Medium medium : media) {
    medium.printInfo();
  }
}
```



printing books & vinyls



```
public void listInfo() {
  System.out.println("MEDIA");
  for (Medium medium : media) {
    medium.printInfo();
  }
}
```



the Catalog cannot find a printInfo method in Medium

back to polymorphism

static type vs dynamic type

what's the type of variable b ?

```
Book b = new Book("Hamlet", "Shakespeare");
```

what's the type of variable m ?

```
Medium m = new Book("Hamlet", "Shakespeare");
```

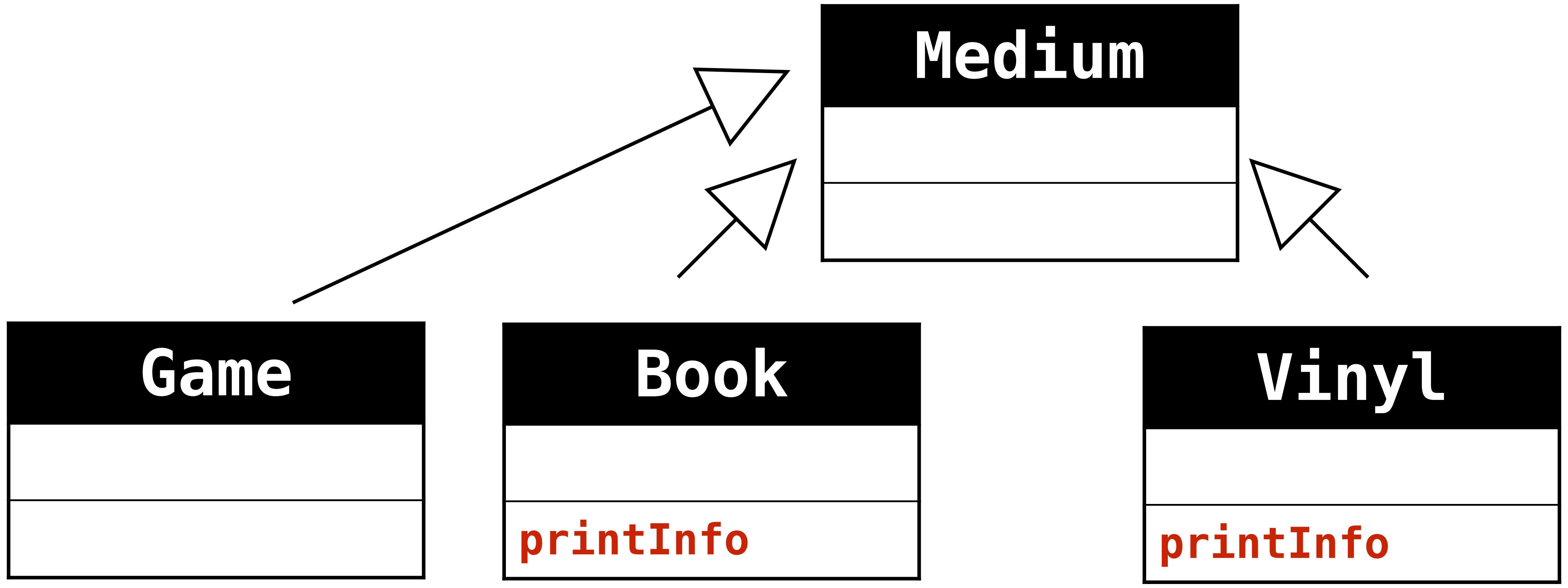
static type

dynamic type

back to polymorphism

what's the type of variable m ?

```
Medium m = new Book("Hamlet", "Shakespeare");
```




```
public void listInfo() {
  System.out.println("MEDIA");
  for (Medium medium : media) {
    medium.printInfo();
  }
}
```

compile-time error!


back to work

try to

add printInfo to

Medium, Book and

Vinyl

printing books & vinyls

```
public void listInfo() {
  System.out.println("MEDIA");
  for (Medium medium : media) {
    medium.printInfo();
  }
}
```

```
public void printInfo() {
  String rating = "?";

  if (this.rating >= 0) {
    rating = "";
    for (int i = 0; i < this.rating; i++) {
      rating = rating + "*";
    }
  }
  System.out.println("Medium[title: " + this.title + " | rating: " + rating + "]");
}
```

Catalog

Medium

printInfo

Vinyl

printInfo

Book

printInfo

```
@Override
public void printInfo() {
  System.out.println("Vinyl[artist: " + this.artist + " | Duration: " + this.duration + "]");
}
```

```
@Override
public void printInfo() {
  System.out.println("Book[author: " + this.author  + "]");
}
```

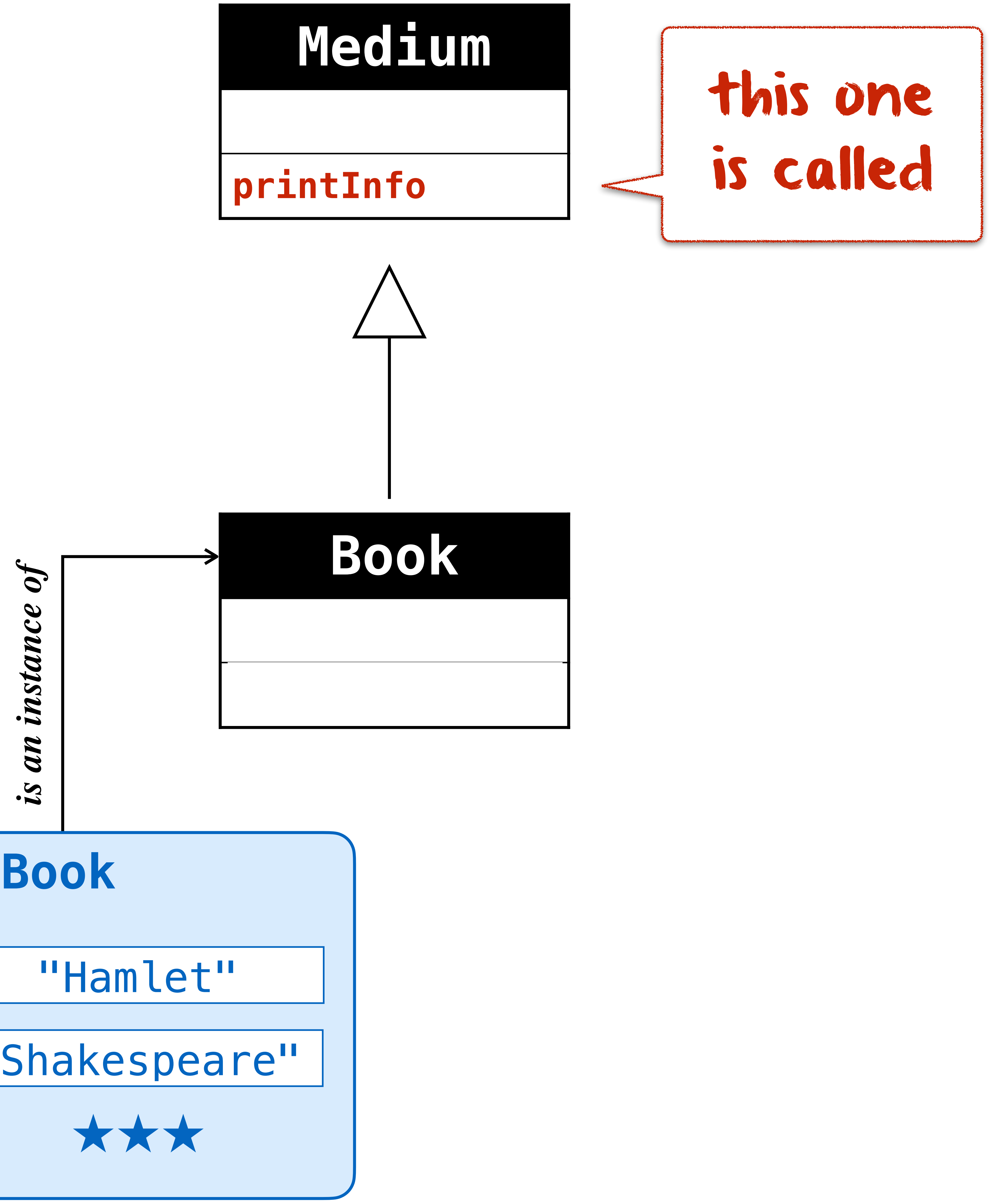
Vinyl[artist: Beatles | Duration: 47]
Book[author: Shakespeare]

no longer printing common fields

the problem

when **overriding** a method in a subclass (Book or Vinyl), the **inherited method** from the super class is **hidden**

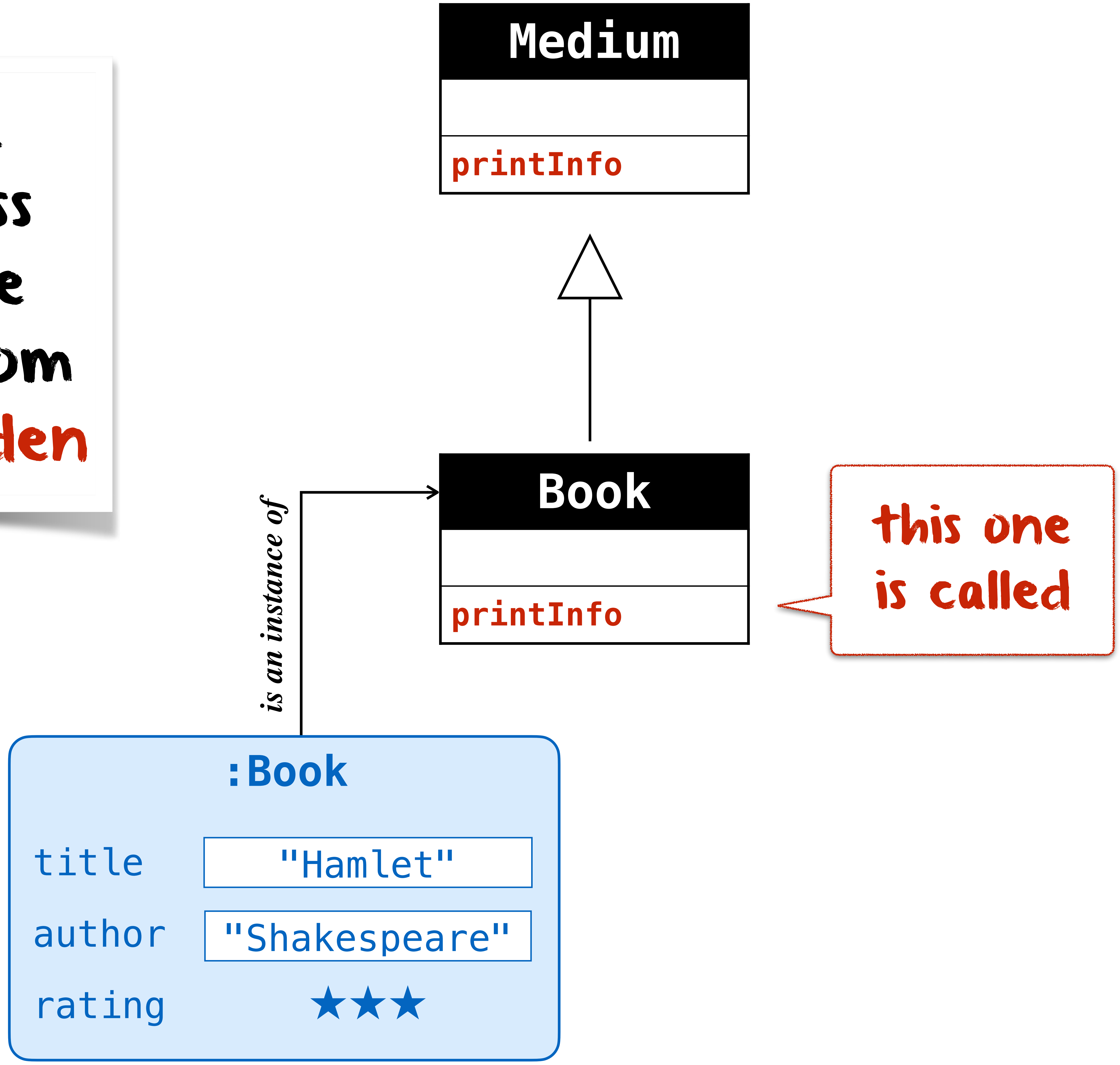
```
public void listInfo() {
  System.out.println("MEDIA");
  for (Medium medium : media) {
    medium.printInfo();
  }
}
```



the problem

when **overriding** a method in a subclass (Book or Vinyl), the **inherited method** from the super class is **hidden**

```
public void listInfo() {
  System.out.println("MEDIA");
  for (Medium medium : media) {
    medium.printInfo();
  }
}
```




almost there

solution

call super

printing books & vinyls

```java
public void listInfo() {
  System.out.println("MEDIA");
  for (Medium medium : media) {
    medium.printInfo();
  }
}
```

Catalog

```java
public void printInfo() {
  String rating = "?";

  if (this.rating >= 0) {
    rating = "";
    for (int i = 0; i < this.rating; i++) {
      rating = rating + "*";
    }
  }
  System.out.println("Medium[title: " + this.title + " | rating: " + rating + "]");
}
```

Medium

printInfo

Vinyl

printInfo

Book

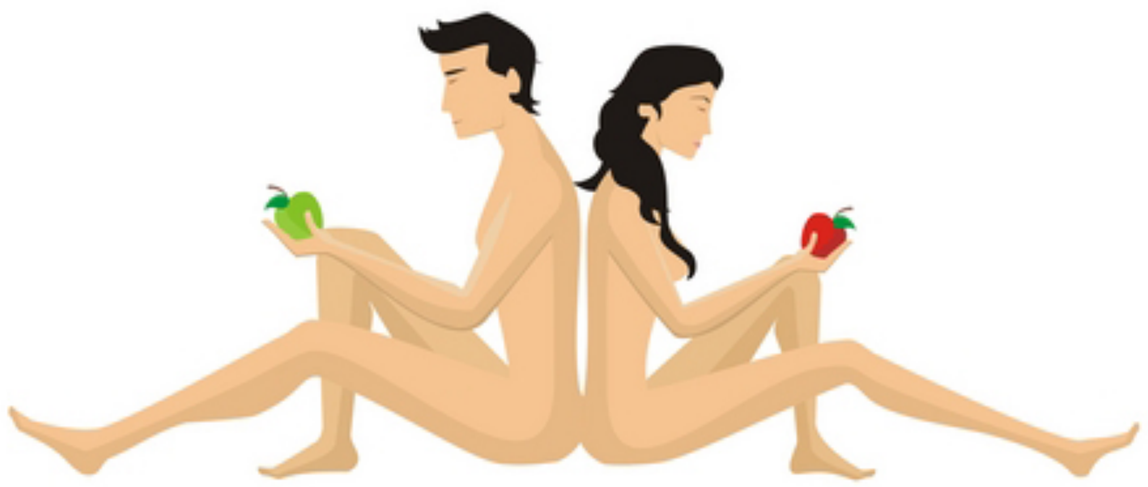
printInfo

```java
@Override
public void printInfo() {
  super.printInfo();
  System.out.println("Vinyl[artist: " + this.artist + " | Duration: " + this.duration + "]");
}
```

```java
@Override
public void printInfo() {
  super.printInfo();
  System.out.println("Book[author: " + this.author  + "]");
}
```

```
Medium[title: Hamlet | rating: ?]
Book[author: Shakespeare]

Medium[title: Abbey Road | rating: ?]
Vinyl[artist: Beatles | Duration: 47]
```

the object superclass



in most object-oriented languages, all classes have an implicit superclass, the `Object` class

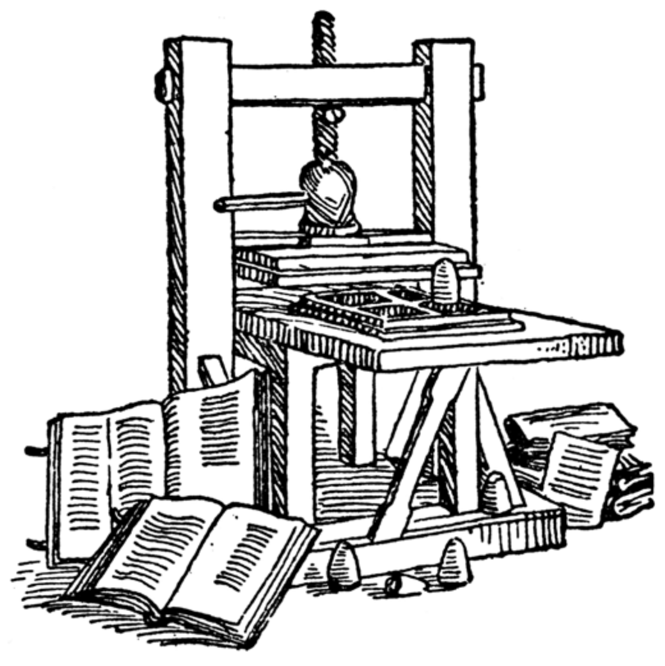
so methods in `Object` are inherited by all classes and can be overridden by any subclasses

for example in Python and in Java, all objects, inherit from a method to represent themselves as strings and can override it

```
class Medium {
    @Override
    public String toString() {
        return "I am a medium";
    }
}
```

```
class Medium:
  def __str__(self):
    return "I am a Medium"
```

back to printing books and vinyls



I promise

one last time

```java
public class Medium {
  ...
  @Override
  public String toString() {
    String rating = "?";

    if (this.rating >= 0) {
      rating = "";
      for (int i = 0; i < this.rating; i++) {
        rating = rating + "*";
      }
    }
    return "title: " + this.title + " | rating: " + rating;
  }

  public void printInfo() {
    System.out.println(this.getClass().getSimpleName() + "[" + this.toString() + "]");
  }
}
```

```java
class Book extends Medium {
  ...
  @Override
  public String toString() {
    return "author:" + this.author + " | " + super.toString();
  }
}
```

```java
class Vinyl extends Medium {
  ...
  @Override
  public String toString() {
    return "artist:" + this.artist + " | duration:" + this.duration + " minutes | " + super.toString();
  }
}
```

```
Book[author: God | title: Bible | rating: ***]
Book[author: Shakespeare | title: Hamlet | rating: ?]
Vinyl[artist: Beatles | duration: 47 minutes | title: Abbey Road | rating: ?]
Vinyl[artist: Police | duration: 40 minutes | title: Synchronicity | rating: ?]
```