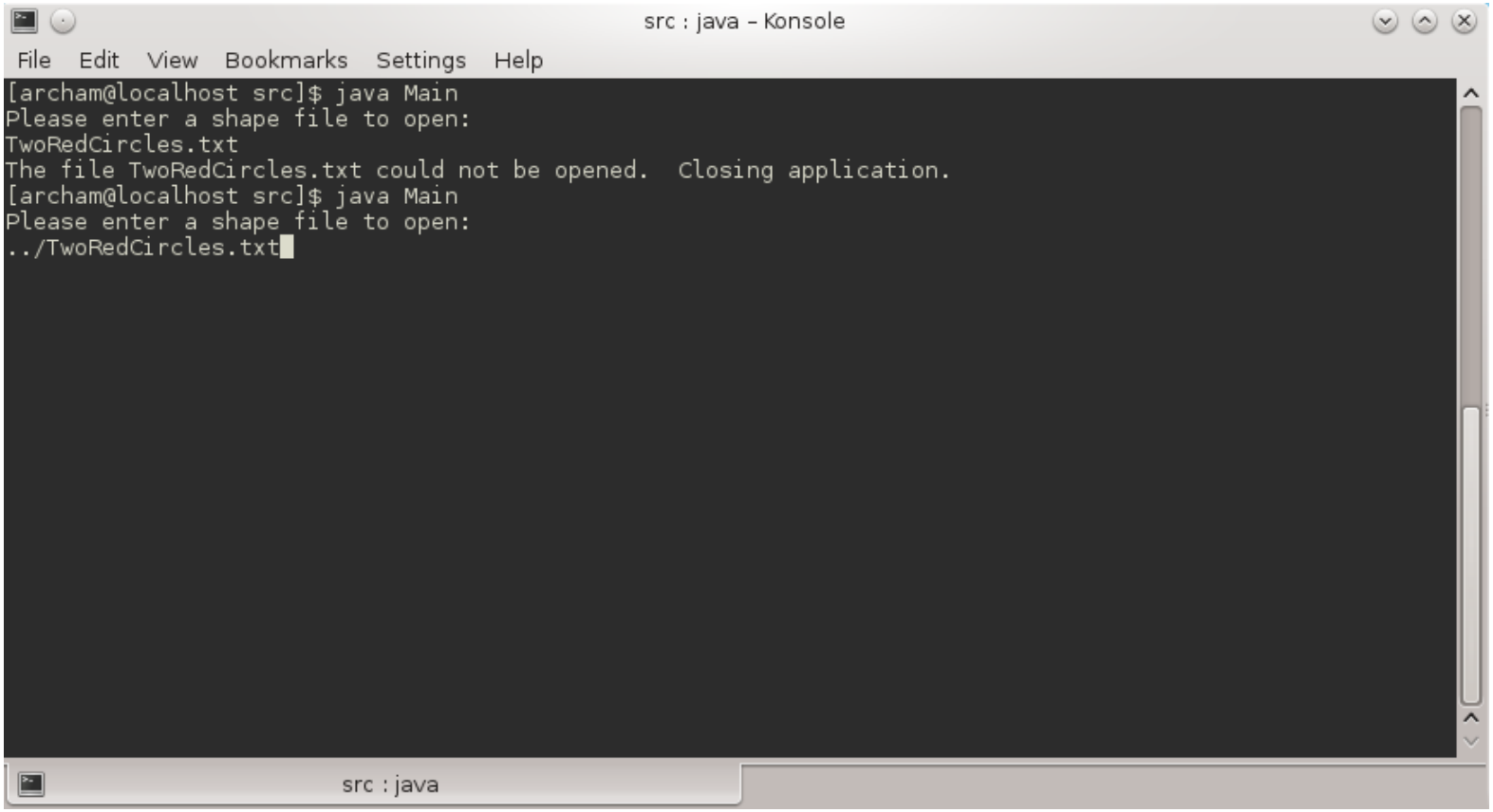
Inheritance

Daniel Archambault

Previously in CS-115



Get the input from files too

Previously in CS-115

- Is opening files similar to `System.in`

Previously in CS-115

- Is opening files similar to `System.in`
- How do we open a file for reading?

Previously in CS-115

- Is opening files similar to `System.in`
- How do we open a file for reading?
- How do we open one for writing?

Previously in CS-115

- Is opening files similar to `System.in`
- How do we open a file for reading?
- How do we open one for writing?
- How do we change the delimiter?

Previously in CS-115

- Is opening files similar to `System.in`
- How do we open a file for reading?
- How do we open one for writing?
- How do we change the delimiter?
- Do we need to close file streams and how do we do it?

Previously in CS-115

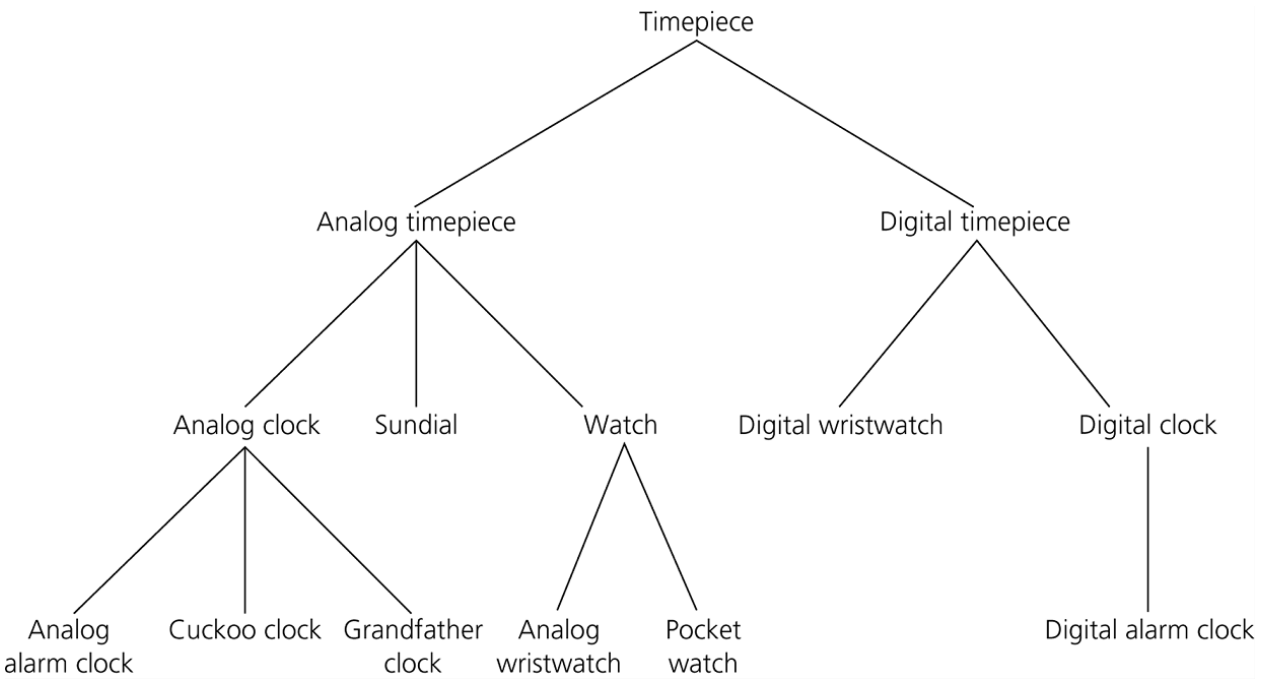
- Objects are good, but sometimes can cause repeated information
- How can we overcome this?

Inheritance

Inheritance

- Is-a relationship
  - a mammal is an animal
  - a bird is a animal
  - a car is a vehicle
  - an aeroplane is a vehicle.
- Animal and vehicle are **superclasses**
  - superclasses are a general class type
- Mammal, bird, car, and aeroplane are **subclasses**
  - subclasses are more specific versions of the superclass.

Example Inheritance Hierarchy



Motivation for Inheritance

- Consider an online store...

class Book	class DVD	class CD	class Teacup
private double price;	private double price;	private double price;	private double price;
private String author;	private String director;	private int numStock;	private int numStock;
private int pages;	private int numStock;	private String artist;	private float vol;
private int numStock;	private double dur;	private double dur;	
private String title;	private String title;	private String title;	

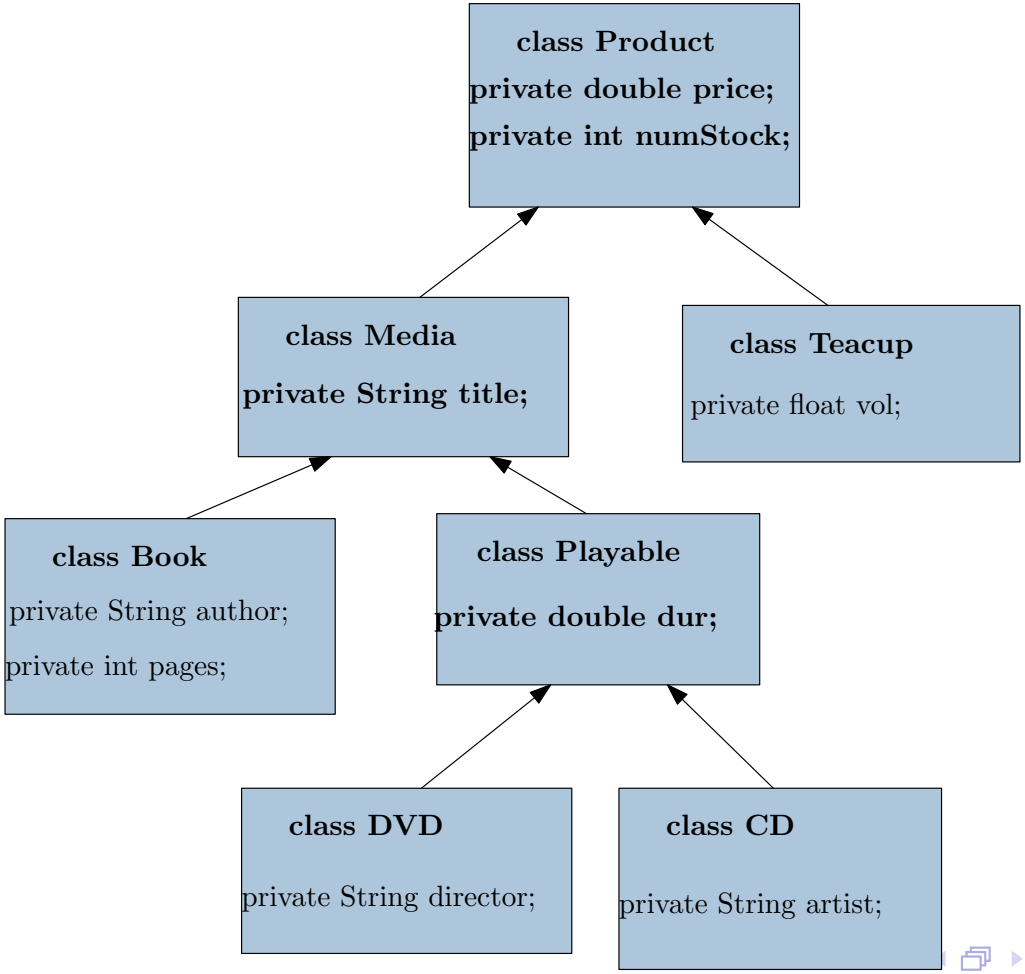
Motivation for Inheritance (2)

- In this design, **lots** of duplication
- Can lead to bugs if we change in one class but not **all**

class Book	class DVD	class CD	class Teacup
private double price;	**private double price;**	**private double price;**	**private double price;**
private String author;	private String director;	**private int numStock;**	**private int numStock;**
private int pages;	**private int numStock;**	private String artist;	private float vol;
private int numStock;	**private double dur;**	**private double dur;**	
private String title;	private String title;	private String title;	

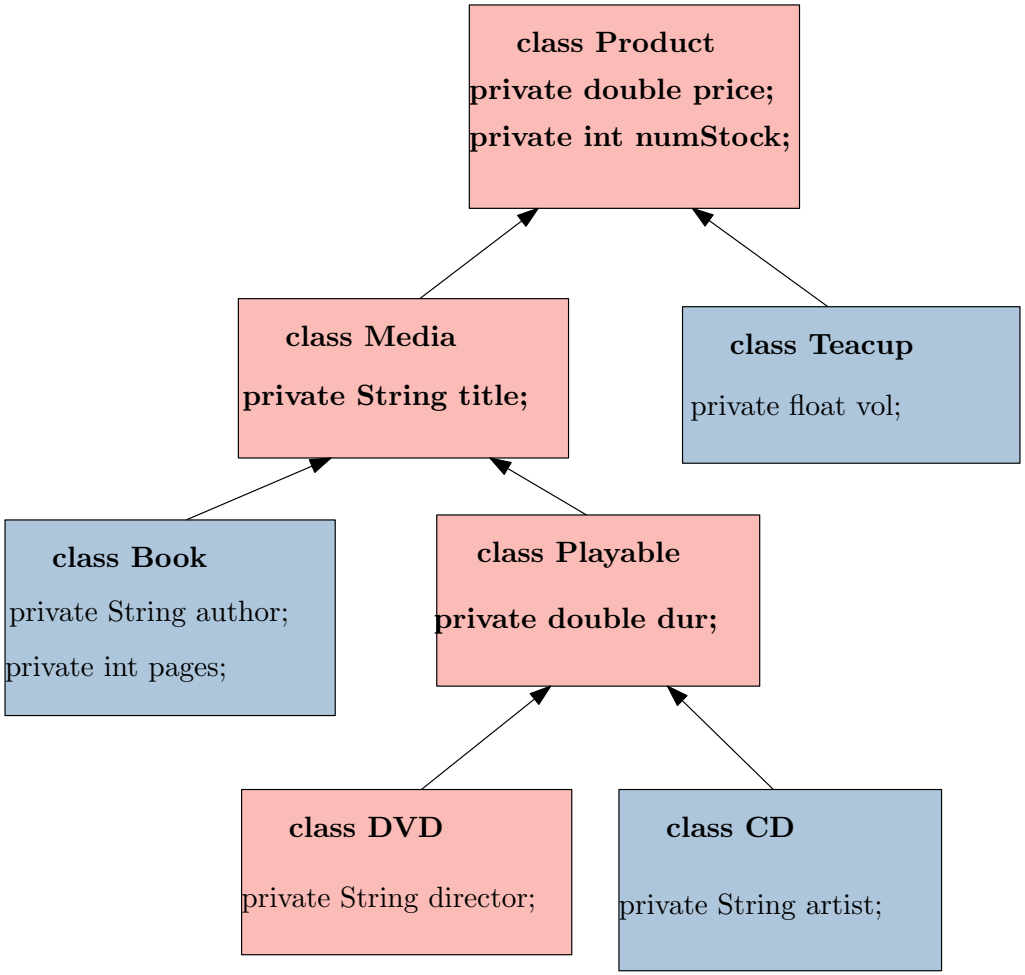
Motivation for Inheritance (3)

- We can factor out common behaviour and attributes, placing in super classes
- All subclasses **inherit** all of the information of the superclass



Motivation for Inheritance (4)

- Information is inherited for all classes above on a path
  - A **DVD** has: price, numStock, title, duration, artist.

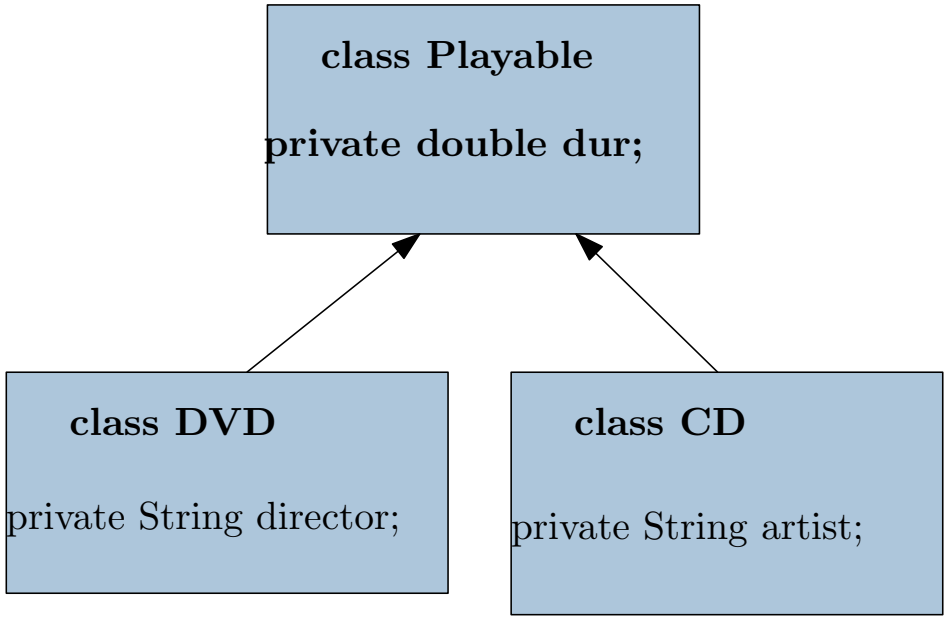


Creating Inheritance Using `extends`

- The `extends` keyword creates inheritance relationships

```
public class DVD extends Playable {

public class CD extends Playable {
```

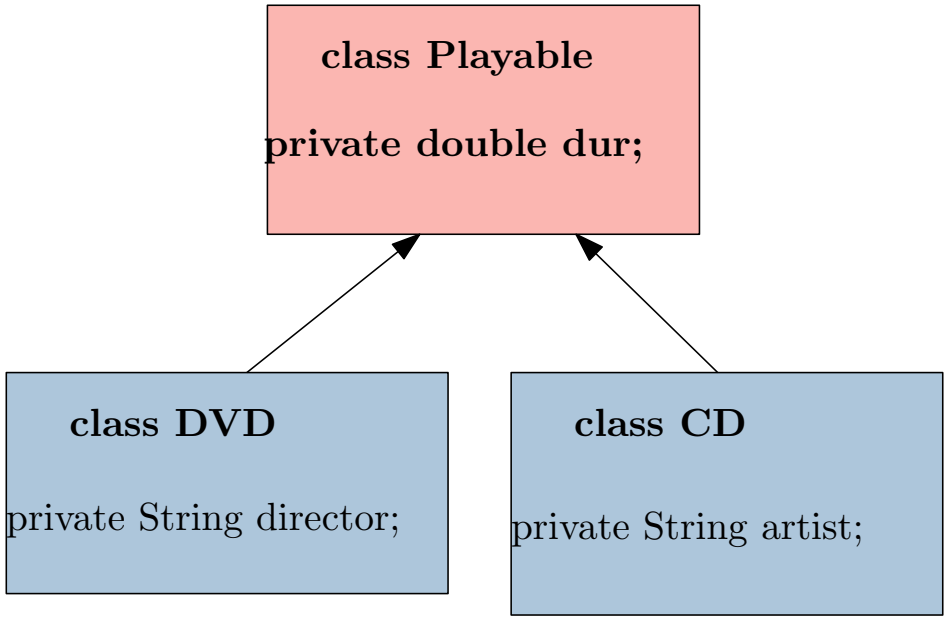


Benefits of Inheritance

- All data and methods in one location
  - makes maintenance easier
  - avoids reimplementation of features
- Data and operations available to all subclasses
- Exceptions in Java form an inheritance hierarchy

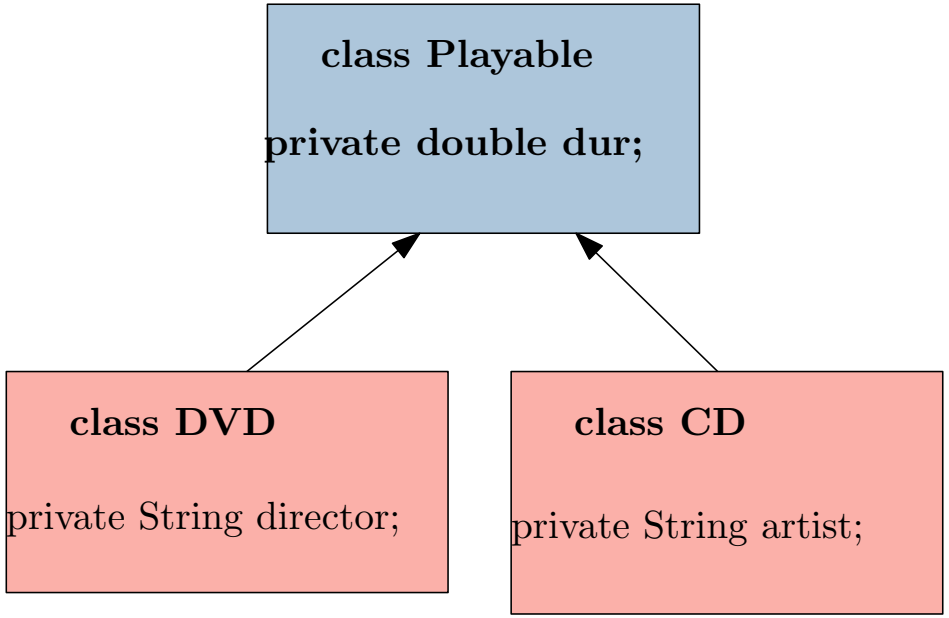
Superclass

- Provides methods and attributes common to all subclasses
- References can store instances of subclasses
- Methods could implement default behaviours unless overridden by subclass



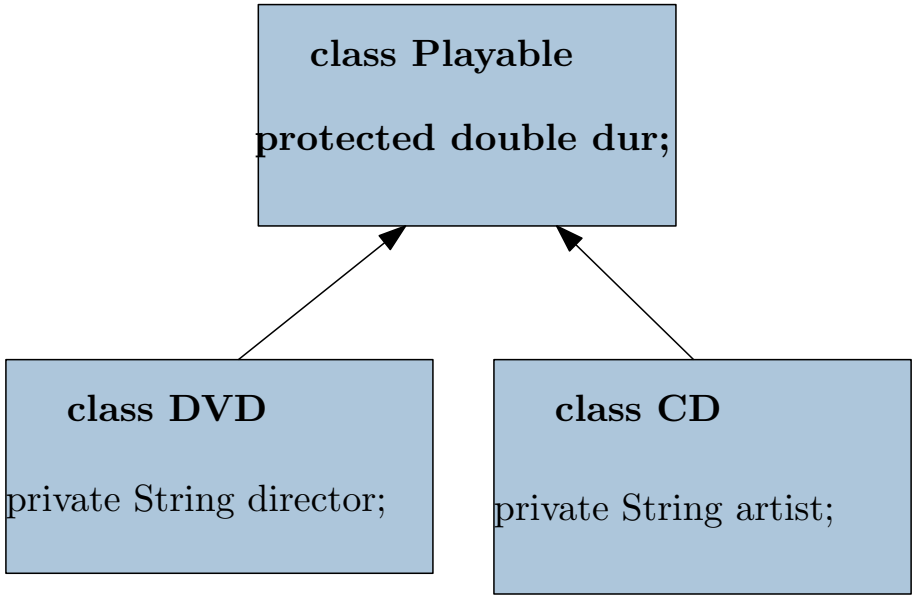
Subclass

- Inherits the methods and attributes of the superclass
- Additional methods and attributes can be added to the subclass
- Methods in the superclass can be overridden
  - done by giving the methods the exact same declarations
- You can call public (and protected) methods of superclass



What is `protected`?

- Private means no calling outside class in which defined (bummer).
- But, the data exists if you are a subclass.
- Is there a way to access this data directly? `protected`
  - protected means than any subclass can access the data directly
  - that is, any class on the `extends` path in the hierarchy



- Now anything that `extends` Playable can access `dur`

What is `super`?

- `super` refers to the superclass of a class
- it is a keyword in Java that you can use when you need to do this
- `super` can be used as a reference
  - `super.myMethod ()` runs myMethod in the superclass
- Usually, you want to run the constructor of the superclass when constructing a subclass so that you can reuse initialisation code
  - `super (...)` calls the superclass constructor, just specify the right parameters
- `https://docs.oracle.com/javase/tutorial/java/IandI/super.html`

Access Quiz

- Ball extends sphere
- Sphere has an attribute `protected int radius;`
- Sphere has a method `public void throwIt ()`
- Ball has a method `protected void bounce ()`
- Ball has a method `public void throwIt ()`
- We are inside the ball class. Is the call legal? What method is called?
  - this.radius;
  - this.throwIt ();
- Outside sphere and ball
  - Ball b = new Ball (.....);
  - Sphere s = new Ball (.....);
  - s.radius;
  - b.radius;
  - s.throwIt ();
  - b.throwIt ();
  - b.bounce ();