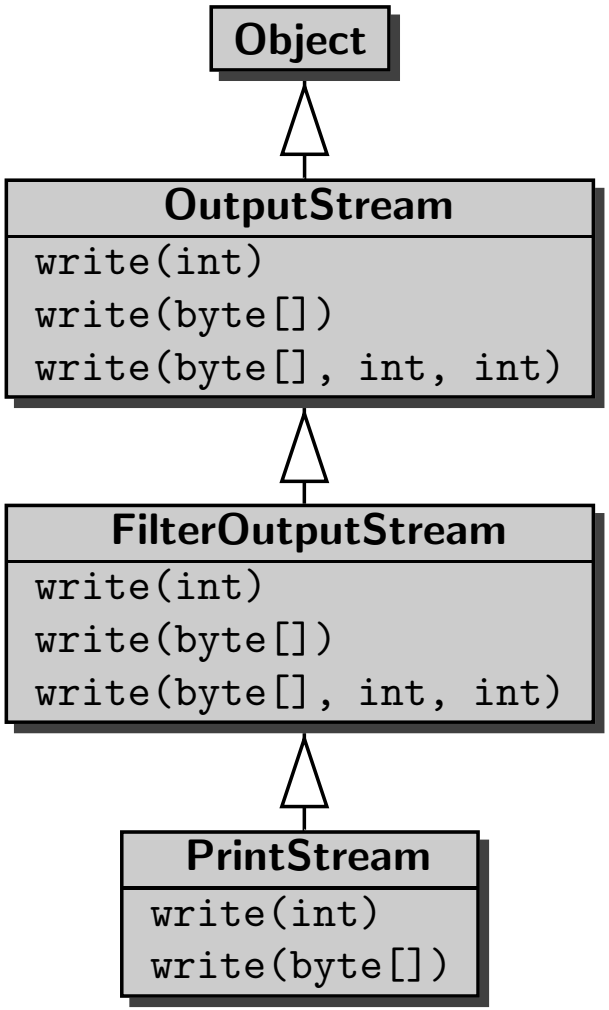
Inheritance



Early binding

Question

Consider the following code snippet.

```
OutputStream stream = System.out;
stream.write(0);
```

To which method of which class is `stream.write(0)` bound early?

Early binding

Question

Consider the following code snippet.

```
OutputStream stream = System.out;
stream.write(0);
```

To which method of which class is `stream.write(0)` bound early?

Answer

`write(int)` of `OutputStream`.

Late binding

Question

Consider the following code snippet.

```
OutputStream stream = System.out;
stream.write(0);
```

To which method of which class is `stream.write(0)` bound late?

Late binding

Question

Consider the following code snippet.

```
OutputStream stream = System.out;
stream.write(0);
```

To which method of which class is `stream.write(0)` bound late?

Answer

`write(int)` of `PrintStream`.

instanceof

The Boolean expression

`r instanceof C`

evaluates to true if `r` is not `null` and its type is `C` or any of its descendants.

Casting: at compile time

Assume that the declared type of the reference `r` is `C`.

- Then `(C')r` gives rise to a compile time error if `C'` is neither a descendant nor an ancestor of `C`.
- If `(C')r` does not give rise to a compile time error, then its declared type is `C'`.

Casting: at compile time

Question

Assume that the declared type of reference `card` is `CreditCard`. Which of the following gives rise to a compile time error?

1. `(RewardCard)card`
2. `(CreditCard)card`
3. `(Object)card`
4. `(Integer)card`

Casting: at compile time

Question

Assume that the declared type of reference `card` is `CreditCard`. Which of the following gives rise to a compile time error?

1. `(RewardCard)card`
2. `(CreditCard)card`
3. `(Object)card`
4. `(Integer)card`

Answer

4.

Casting: at run time

`(C')r` gives rise to a run time error if the actual type of `r` is not a descendant of `C'`.

Casting: at run time

Question

Assume that the actual type of reference `card` is `CreditCard`. Which of the following gives rise to a run time error?

1. `(RewardCard)card`
2. `(CreditCard)card`
3. `(Object)card`

Casting: at run time

Question

Assume that the actual type of reference `card` is `CreditCard`. Which of the following gives rise to a run time error?

1. `(RewardCard)card`
2. `(CreditCard)card`
3. `(Object)card`

Answer

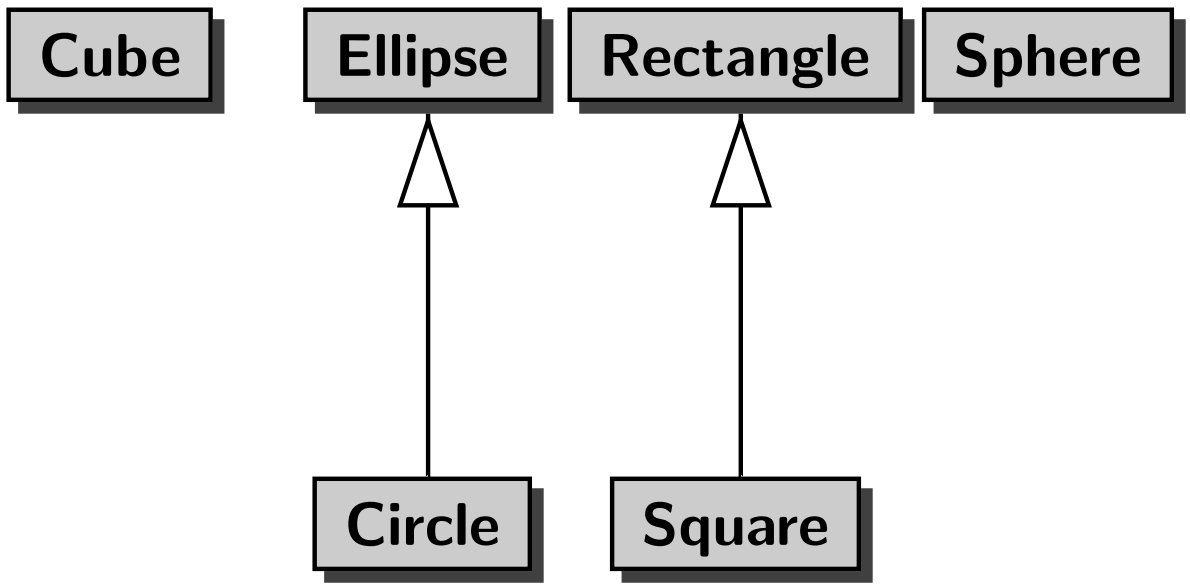
1.

instanceof and casting

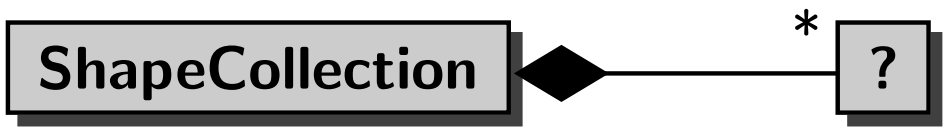
```
if (card instanceof RewardCard) {
    ... (RewardCard) card ...
}
```

- `(RewardCard) card` is needed at compile time
- `card instanceof RewardCard` is needed at run time

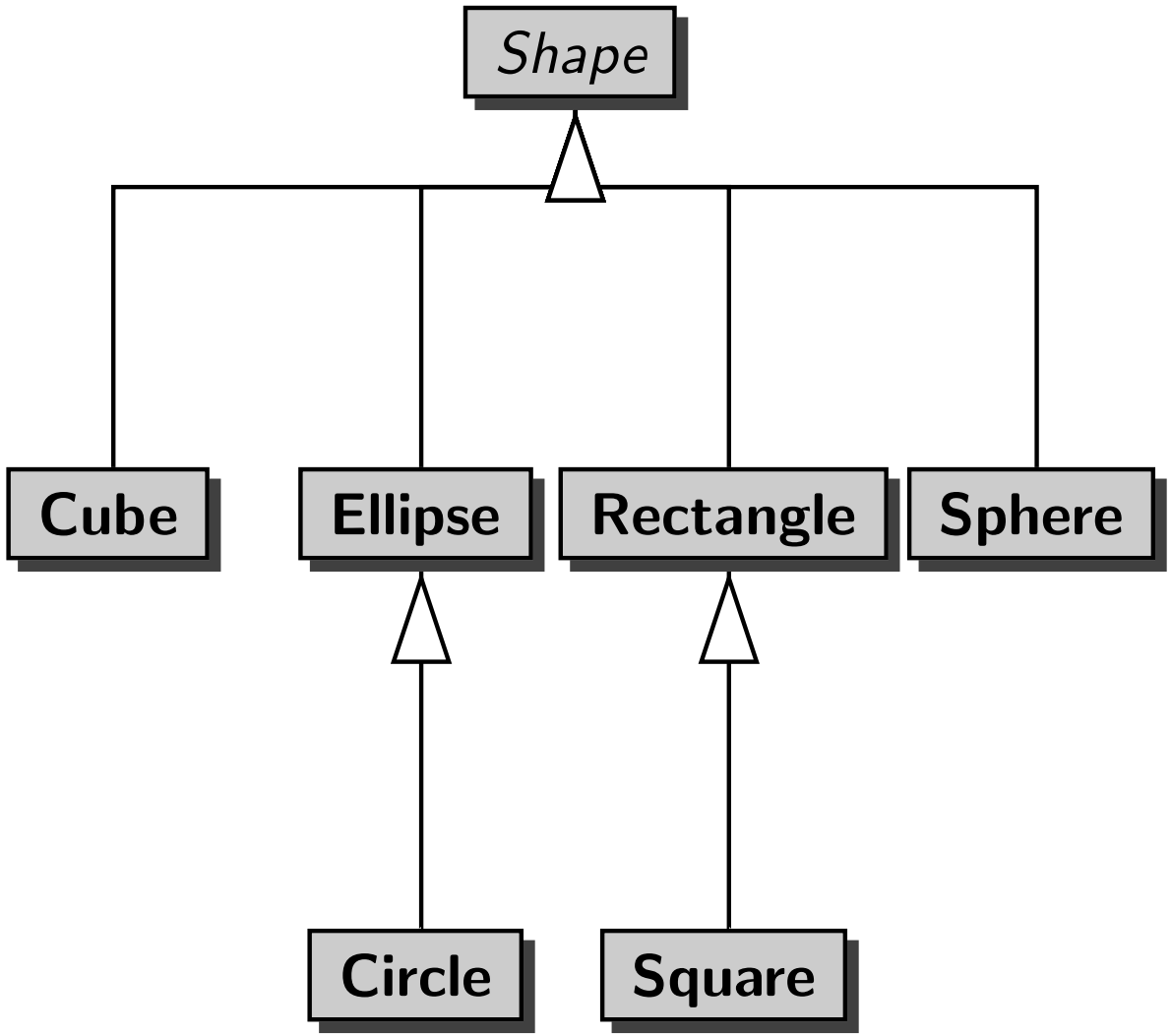
Shapes



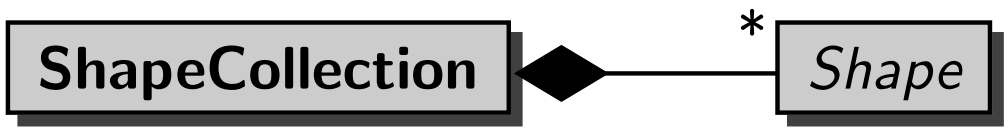
Collection of shapes



Shape



Collection of shapes



Shape

Question

Can you draw a rectangle, ellipse, etc?

Shape

Question

Can you draw a rectangle, ellipse, etc?

Answer

Yes!

Shape

Question

Can you draw a rectangle, ellipse, etc?

Answer

Yes!

Question

Can you draw a shape?

Shape

Question

Can you draw a rectangle, ellipse, etc?

Answer

Yes!

Question

Can you draw a shape?

Answer

No. Shape is an abstract notion.

Question

Can you create a `Rectangle` object, `Ellipse` object, etc?

Shape

Question

Can you create a `Rectangle` object, `Ellipse` object, etc?

Answer

Yes!

Shape

Question

Can you create a `Rectangle` object, `Ellipse` object, etc?

Answer

Yes!

Question

Should one be able to create a `Shape` object?

Shape

Question

Can you create a `Rectangle` object, `Ellipse` object, etc?

Answer

Yes!

Question

Should one be able to create a `Shape` object?

Answer

No.

Abstract class

An abstract class cannot be instantiated, that is, we cannot create instances of the class.

An abstract class may contain methods.

Question

If one cannot create instances of a class, are its methods of any use?

Abstract class

An abstract class cannot be instantiated, that is, we cannot create instances of the class.

An abstract class may contain methods.

Question

If one cannot create instances of a class, are its methods of any use?

Answer

Yes! They can be inherited by subclasses.

Abstract class

- API: `public abstract class Shape`
- UML: class name in *italics*

Shape collection

Problem

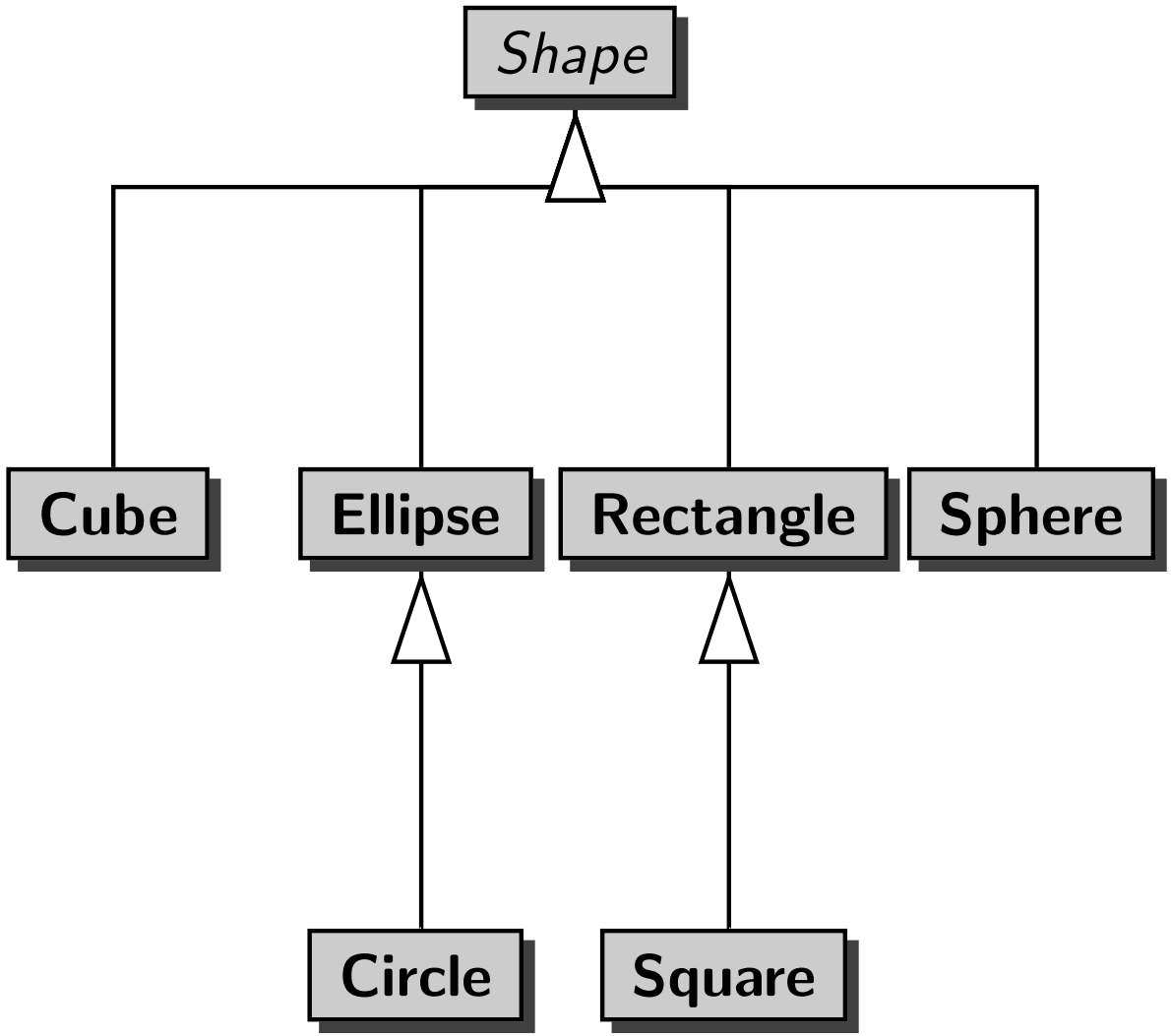
Create a random collection of shapes and print the total area of all shapes combined.

Shape collection

Problem

Create a random collection of shapes and print the total volume of all shapes combined.

Shape



Shape collection

Only `Cube` and `Sphere` have a volume.

Question

Can we introduce an abstract class `HasVolume` with method `getVolume()` as a superclass for `Cube` and `Sphere`?

Shape collection

Only `Cube` and `Sphere` have a volume.

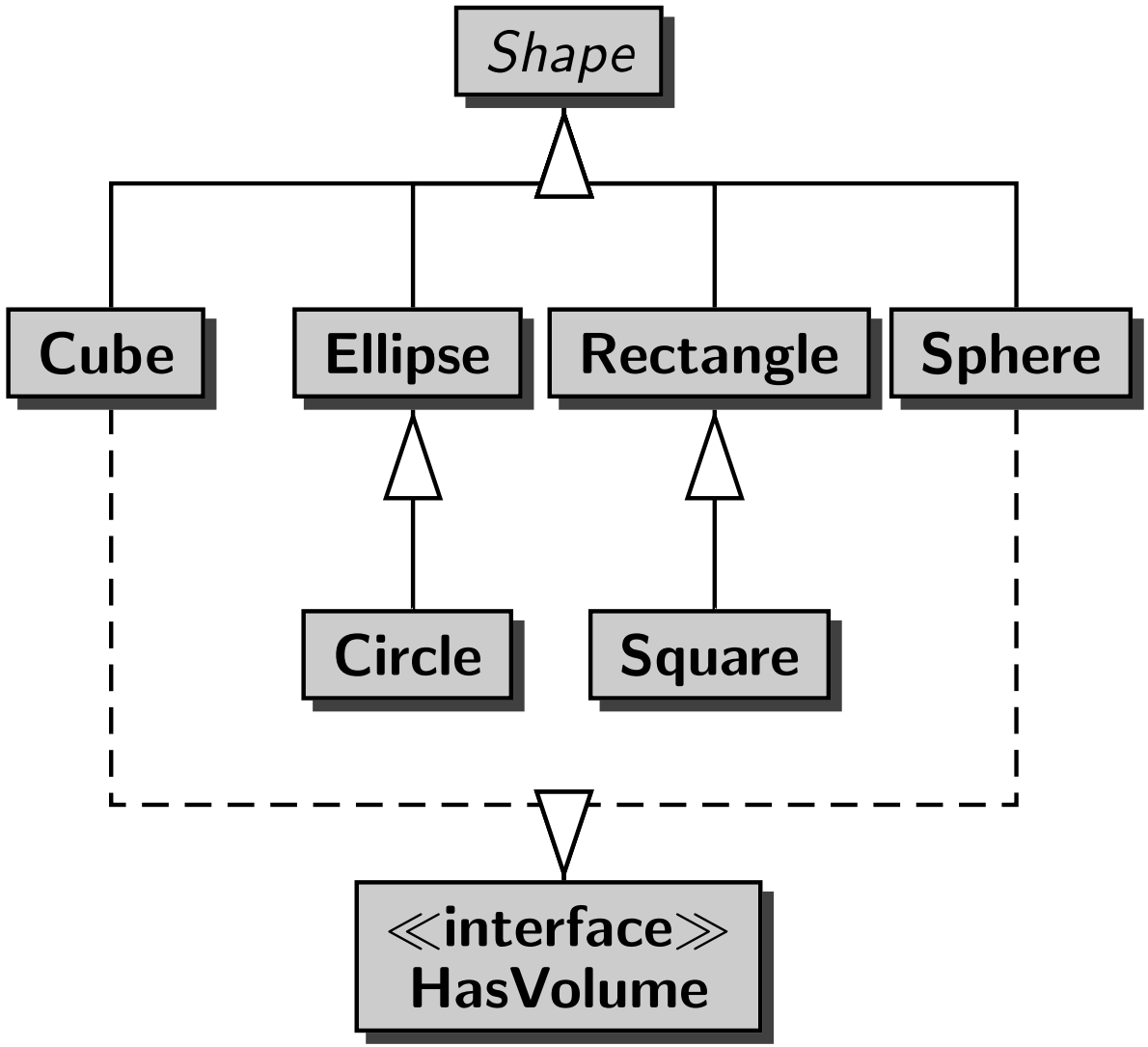
Question

Can we introduce an abstract class `HasVolume` with method `getVolume()` as a superclass for `Cube` and `Sphere`?

Answer

No, because `Cube` and `Sphere` already have a superclass and Java does not support multiple inheritance.

Shape


Shape
Cube
Ellipse
Rectangle
Sphere
Circle
Square
≪interface≫
HasVolume

Interface

An interface only contains method headers (specifications).

Interface

- API: `public interface HasVolume`
- UML: interface name preceded by ≪interface≫

Interface

An interface specifies methods, it does not provide an implementation for them.

A class C implements an interface I if C contains an implementation of each method specified in I.

A class can implement multiple interfaces.

Class implements interface

- API: `public class Cube implements HasVolume`
- UML: dashed arrow

Another interface: Iterator

`Interface Iterator<E>`

`E` is a type parameter.

To use the `Iterator` interface, you need to provide a type as argument.

```
Iterator<Shape> iterator = collection.iterator();
```

To do

- Study the remainder of Chapter 9 of the textbook.