Abstract class

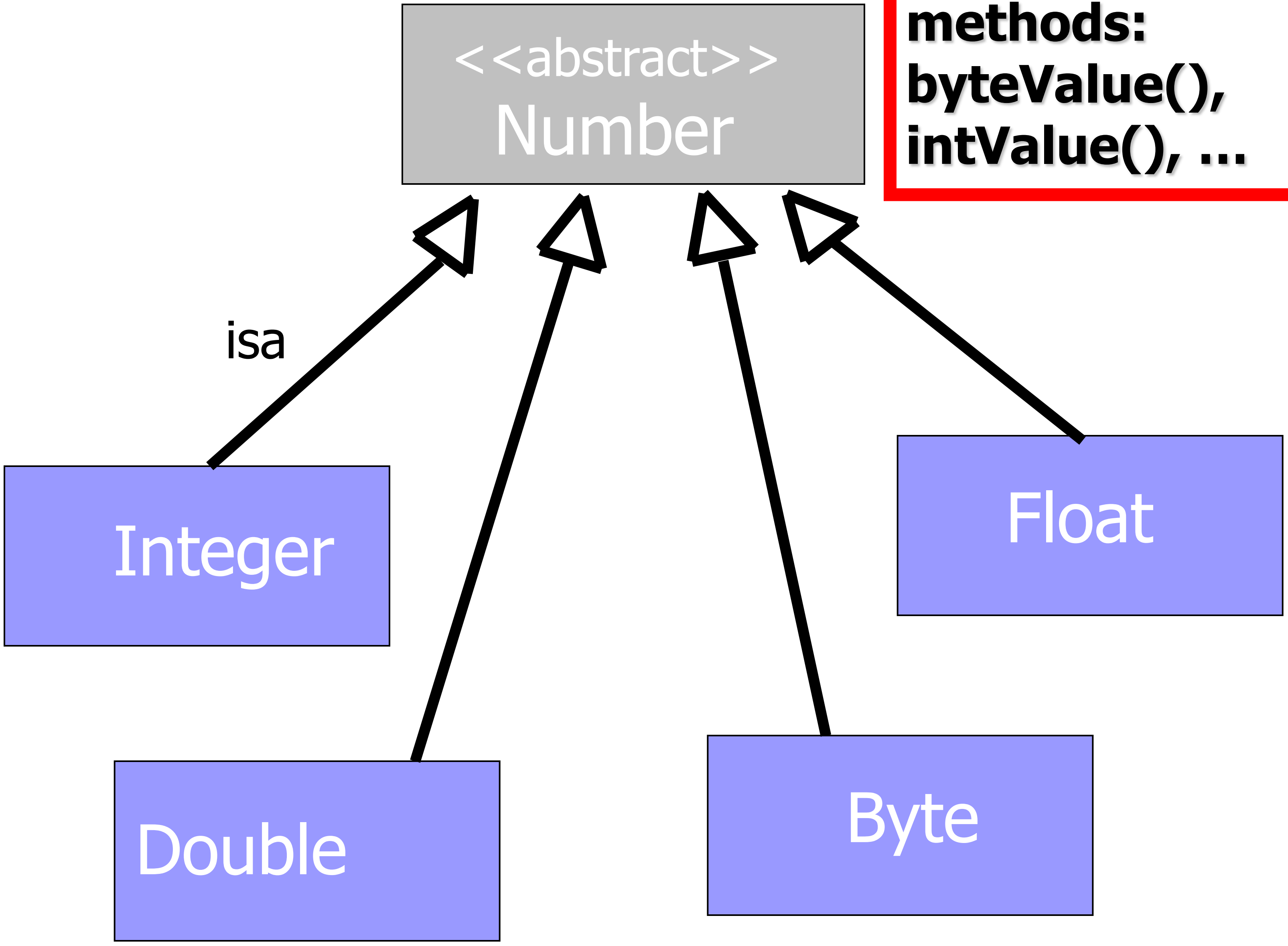
Abstract Class

Some classes exist only so their methods can be inherited (guarantees that all descendants share a common set of operations on their public interface)

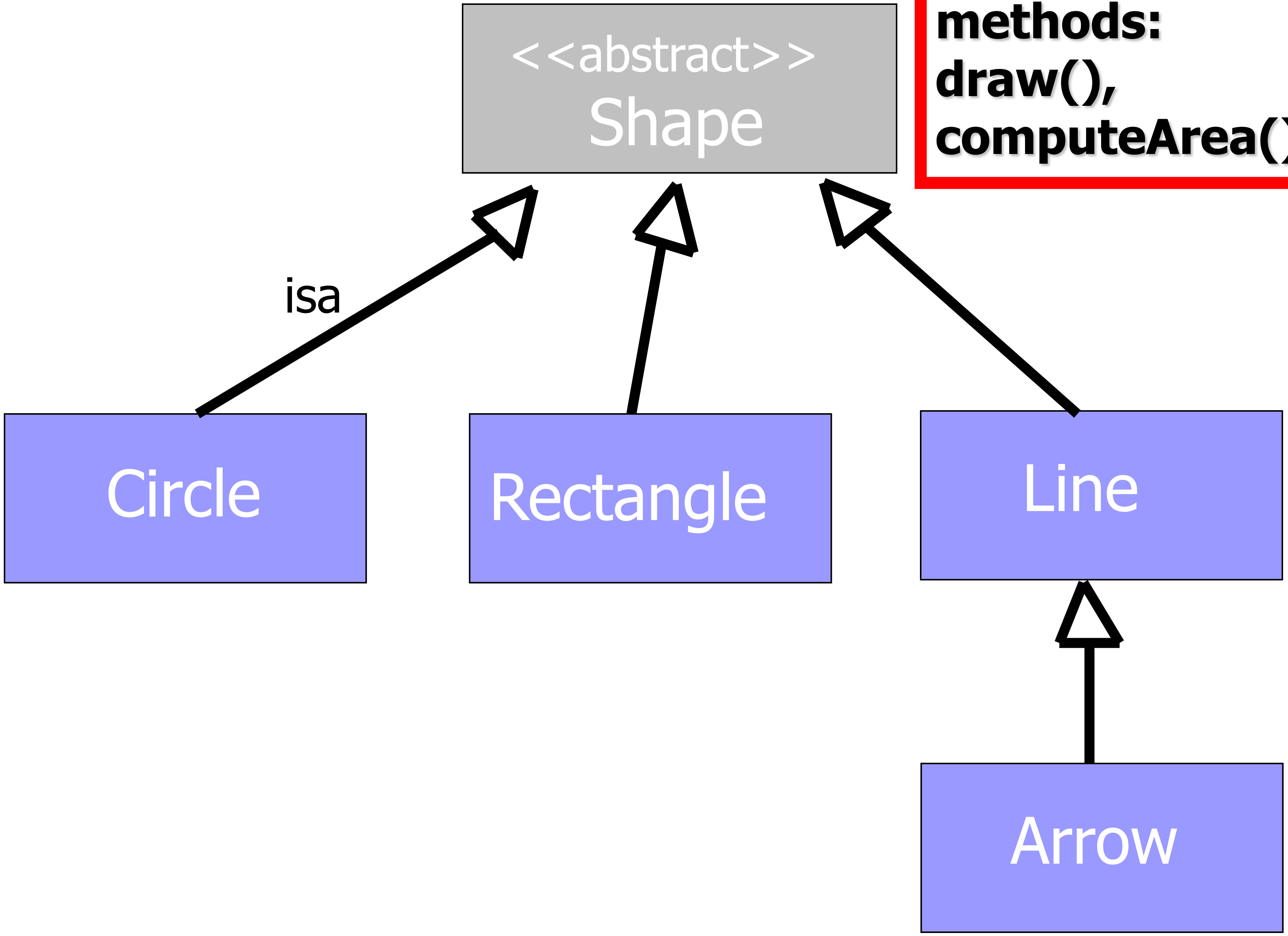
Cannot instantiate an abstract class

examples:

- Number
  - Integer, Float, Double, …
- Shape
  - Circle, Line, Rectangle, …



methods: byteValue(), intValue(), ...



**methods:
draw(),
computeArea()**

```
public abstract class Shape
{
  // can define constants
  public static final double TWO_PI = 2*Math.PI;

  // can declare abstract methods
  public abstract double computeArea();

  public abstract void draw();

  // can implement methods
  public String aka() { return “euclidean”; }
}
```

```
public class Circle extends Shape
    {
    // override draw() & computeArea()
}


public class Rectangle extends Shape

{
    // override draw() & computeArea()

}
```

Abstract class

“template for a collection of related subclasses”

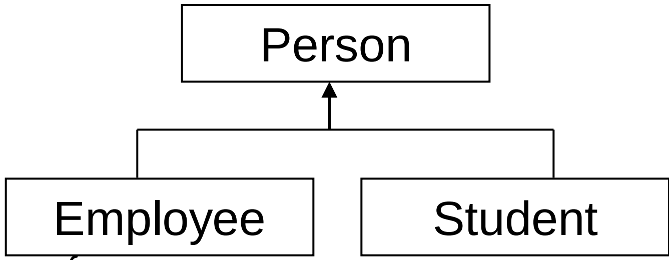
- may contain instance variables, constants, concretely implemented methods
- when a class extends an abstract class, it may implement all or some of the methods; if it does not implement all abstract methods, then it must be declared to be abstract itself
- cannot instantiate an abstract class
- can declare a reference to one

Abstract classes and methods

```
abstract class Person {
   protected String name;
   public abstract String getDescription();
   . . . }

Class Student extends Person {
  private String major;
  public String getDescription() {
    return "a student of " + major;
  } . . . }

Class Employee extends Person {
  private float salary;
  pulic String getDescription() {
    return "an employee with a salary of $
  " + salary;
  } . . . }
```



Abstract classes and methods

- each method which has no implementation in the `abstract` class must be declared `abstract`
- any class with any `abstract` methods must be declared `abstract`
- when you extend an `abstract` class, two situations
  1. leave some or all of the abstract methods be still undefined. Then the subclass must be declared as `abstract` as well
  2. define concrete implementation of all the inherited abstract methods. Then the subclass can be abstract or concrete.

Abstract classes and methods

- an object of an `abstract` class can **NOT** be created
- note that declaring object variables of an abstract class is still allowed, but such a variable can only refer to an object of a non-abstract subclass

```
        Person p = new Student( );
```