luv2code

Abstract Classes

Abstract Class

Abstract Class

- An abstract class represents a general concept

Abstract Class

- An abstract class represents a general concept
  - For example: Shape, Vehicle, Computer, etc …

Abstract Class

- An abstract class represents a general concept
  - For example: Shape, Vehicle, Computer, etc …

- Can't create an instance of an abstract class

Abstract Class

- An abstract class represents a general concept
  - For example: Shape, Vehicle, Computer, etc …

- Can't create an instance of an abstract class
- Abstract class can also have abstract method(s)

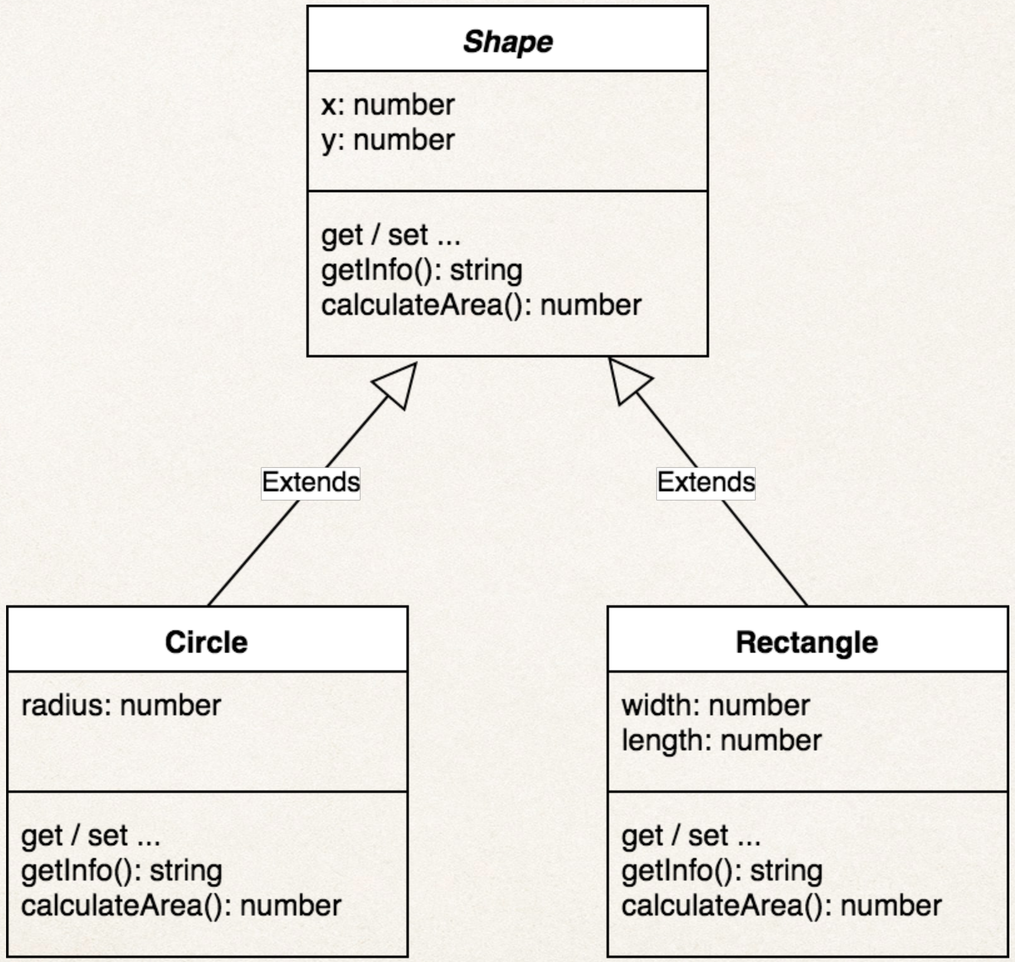
Abstract Class

- An abstract class represents a general concept
  - For example: Shape, Vehicle, Computer, etc …

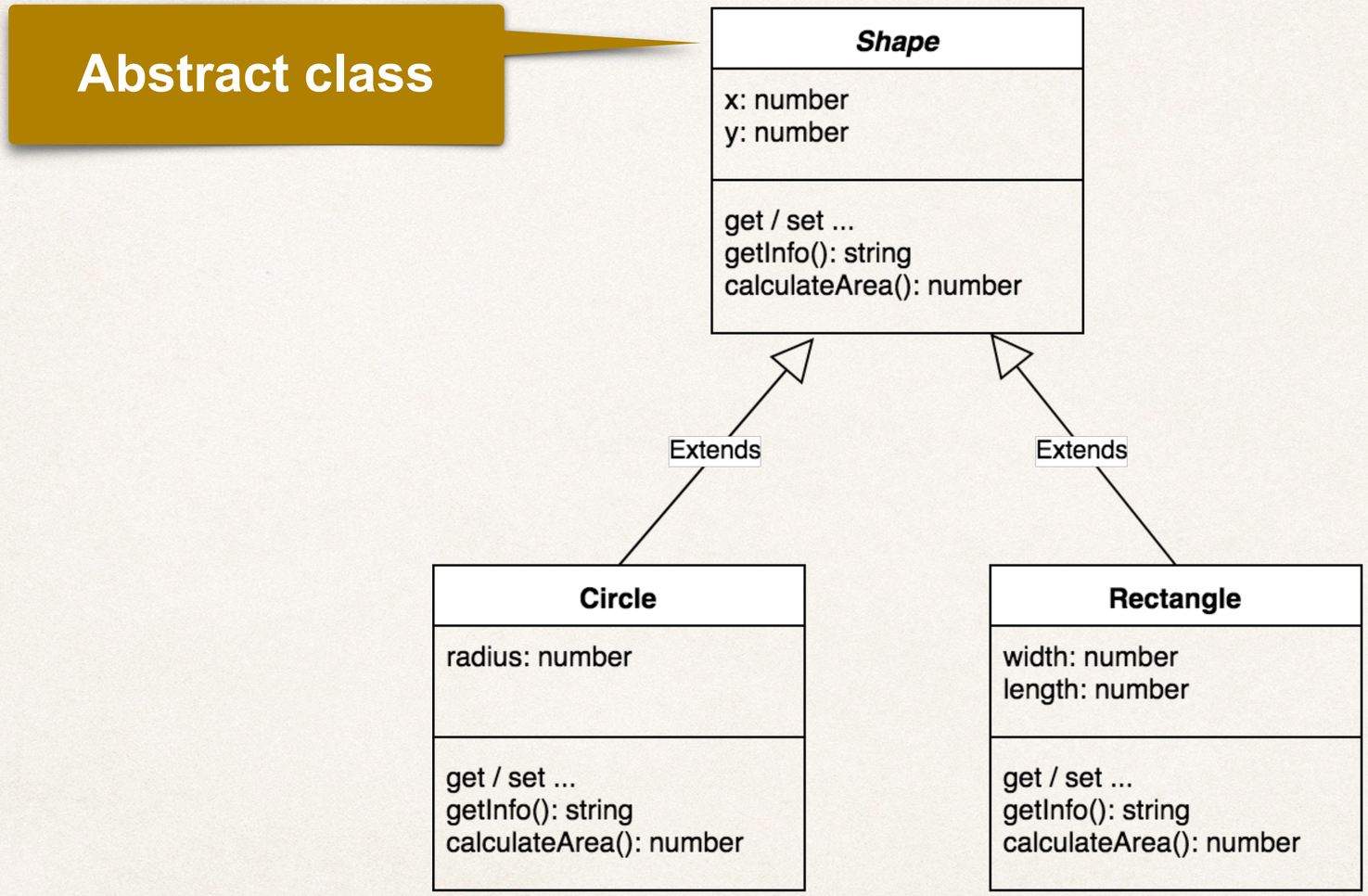
- Can't create an instance of an abstract class
- Abstract class can also have abstract method(s)
- Abstract method must be implemented by concrete subclasses

Abstract Class Example

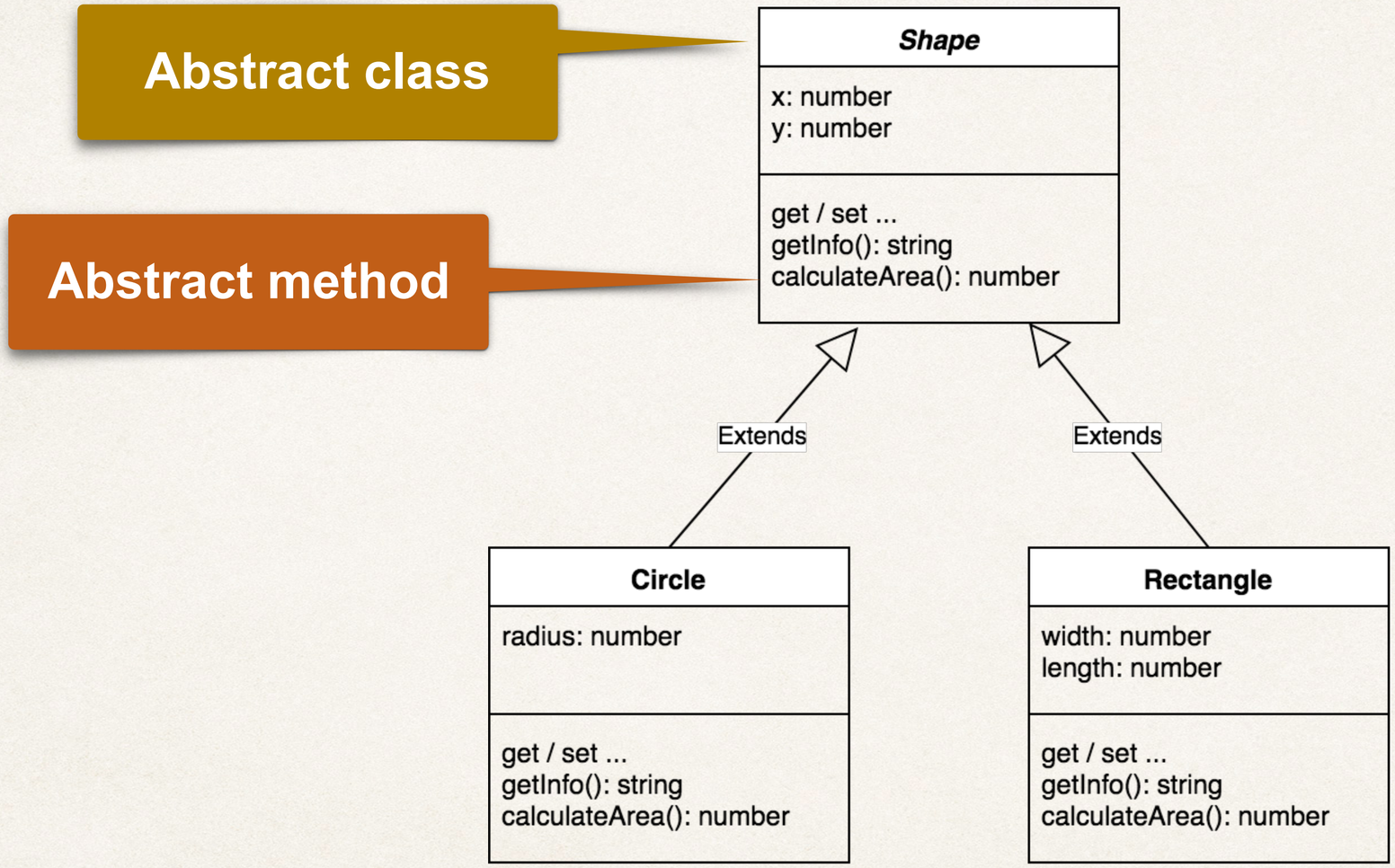
Abstract Class Example



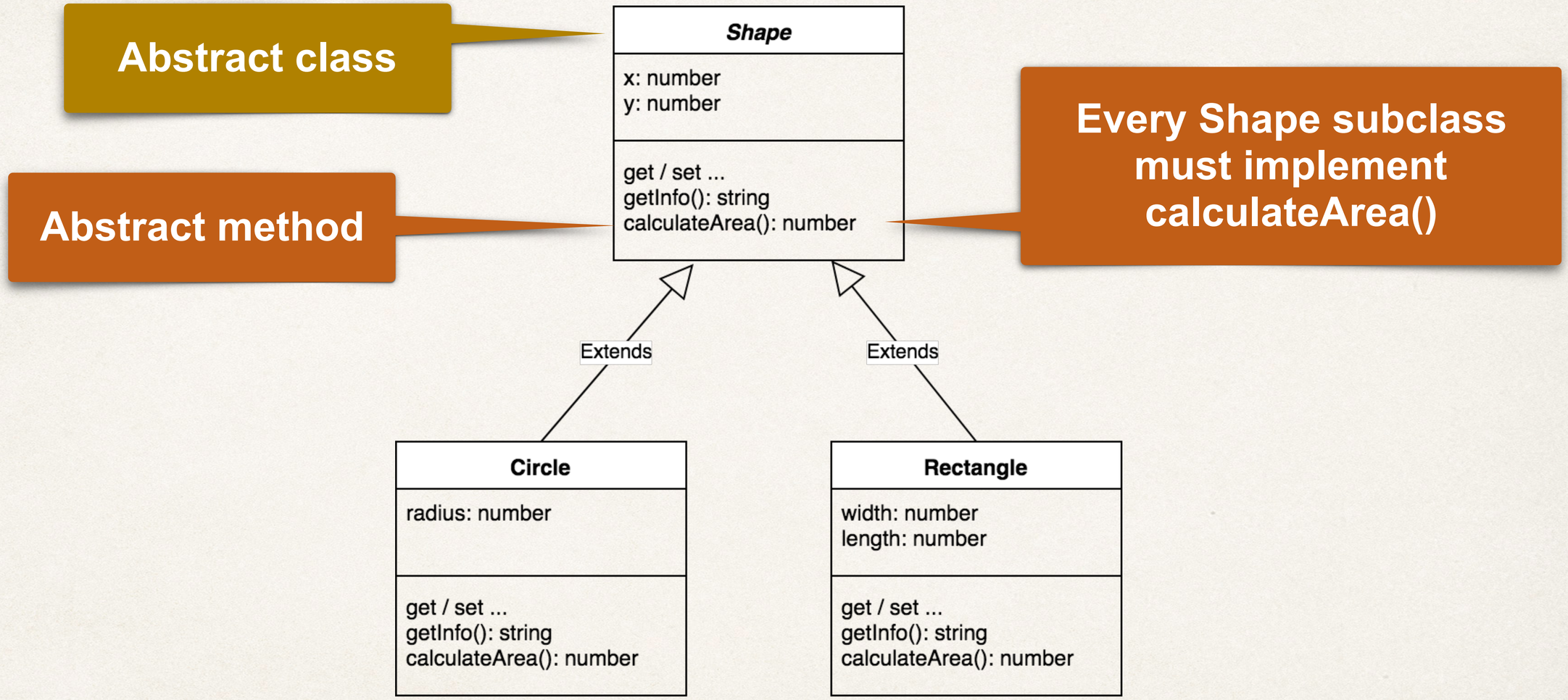
Abstract Class Example



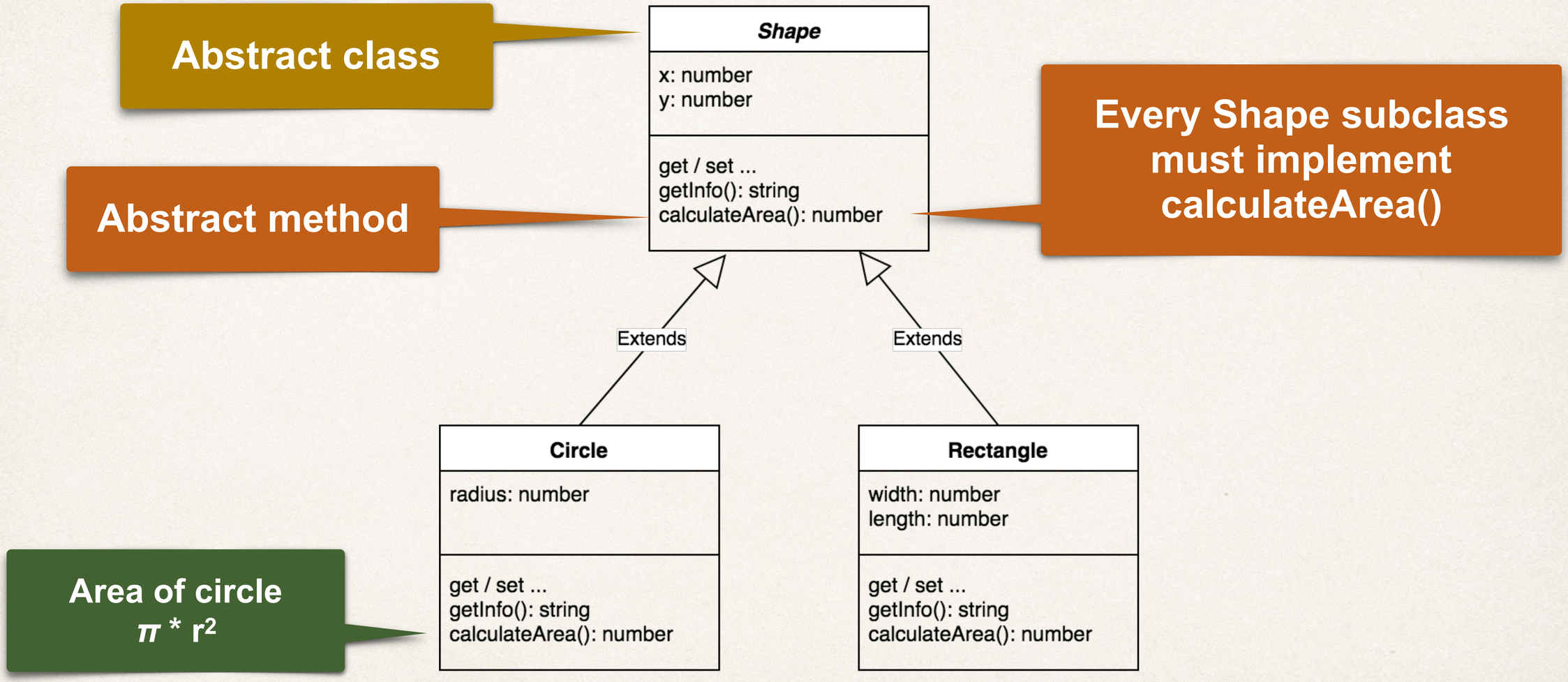
Abstract Class Example



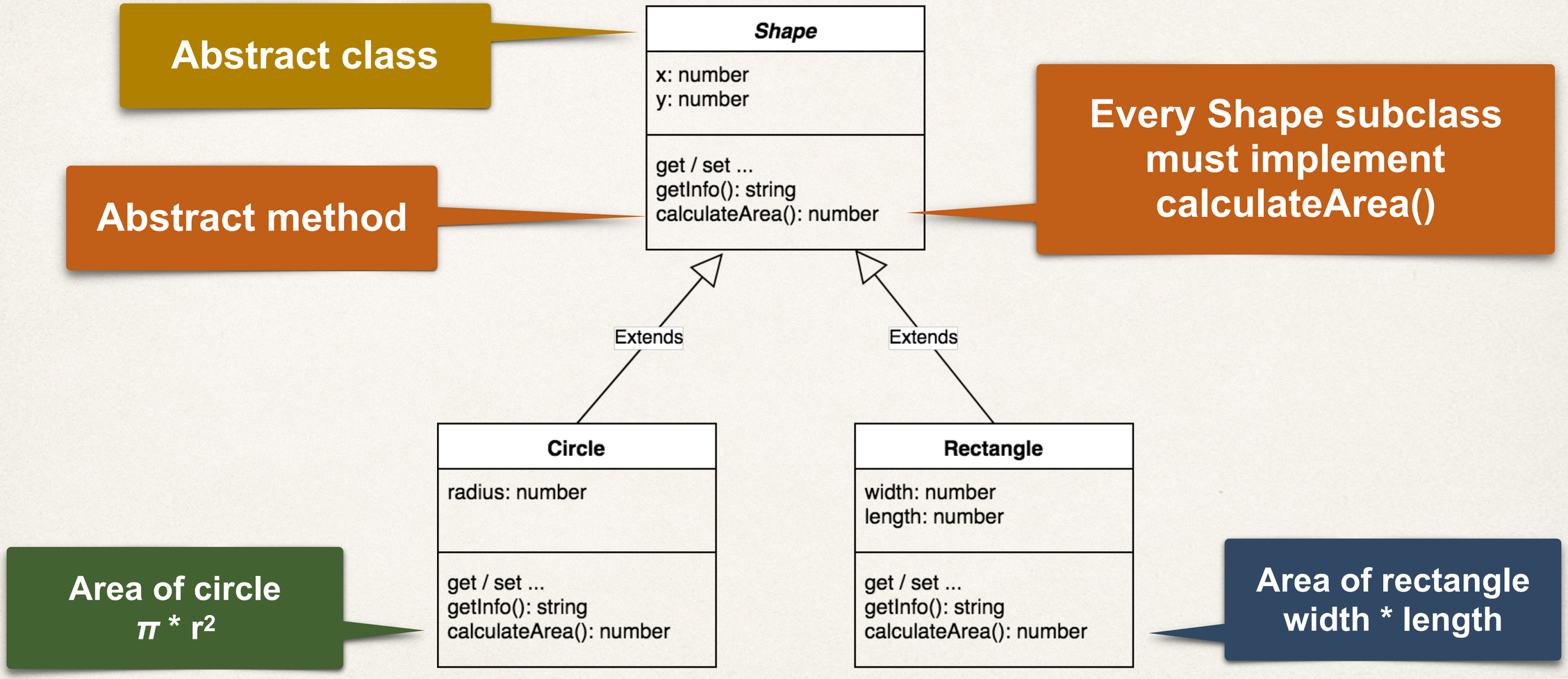
Abstract Class Example



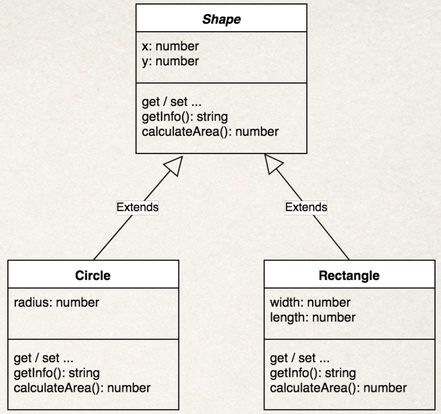
Abstract Class Example



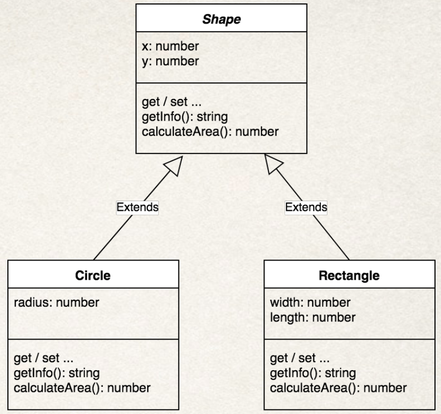
Abstract Class Example



Abstract Class Example



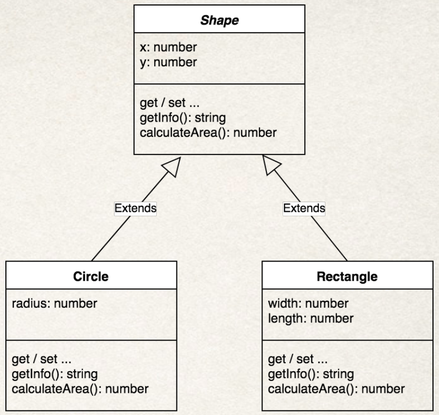
Abstract Class Example



File: Shape.ts

```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

Abstract Class Example

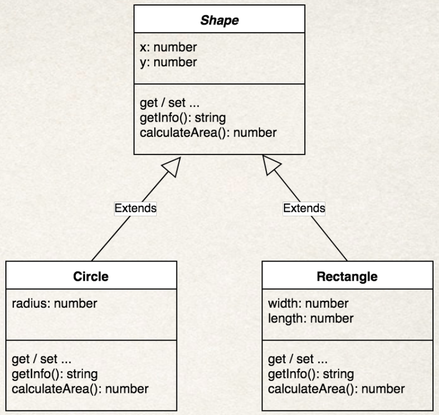


Mark the class as abstract

File: Shape.ts

```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

Abstract Class Example



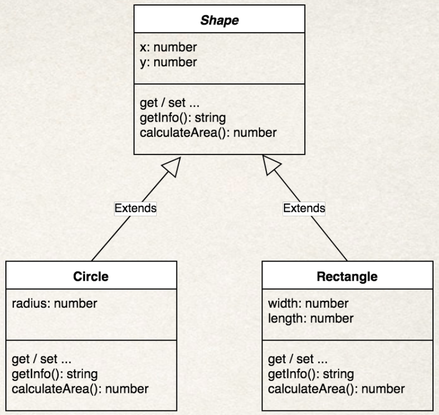
Mark the class as abstract

File: Shape.ts

```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

Abstract method

Abstract Class Example



Mark the class as abstract

File: Shape.ts

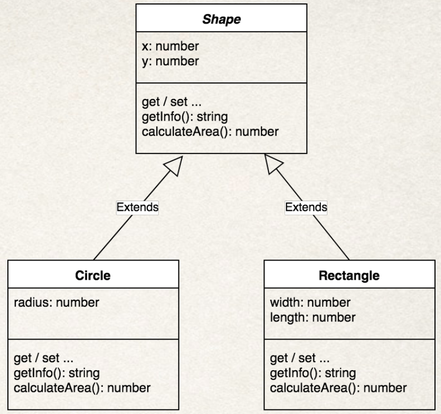
```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

Abstract method

File: Rectangle.ts

```
import { Shape } from './Shape';

export class Rectangle extends Shape {

    // previous code ...

    calculateArea(): number {
        return this._width * this._length;
    }
}
```

Abstract Class Example



Mark the class as abstract

File: Shape.ts

```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

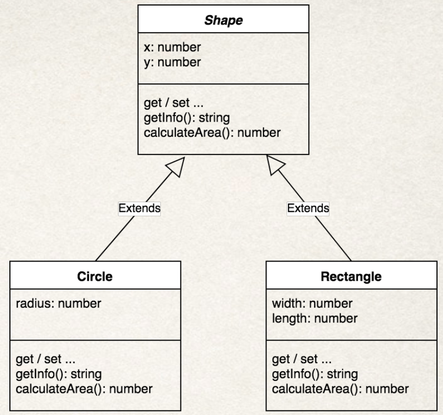
Abstract method

Override the calculateArea() method

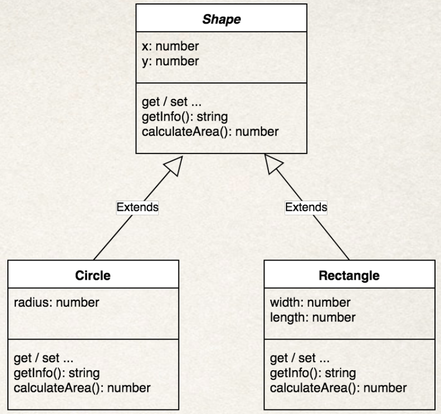
File: Rectangle.ts

```
import { Shape } from './Shape';

export class Rectangle extends Shape {

    // previous code ...

    calculateArea(): number {
        return this._width * this._length;
    }
}
```

Circle



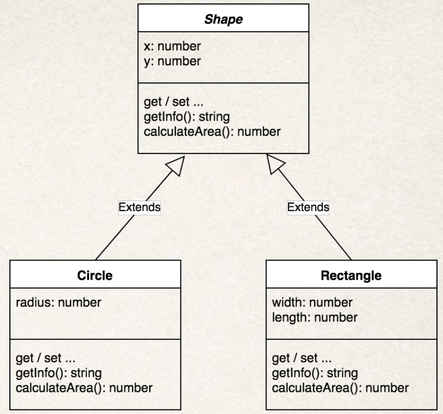
Circle



File: Shape.ts

```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

Circle



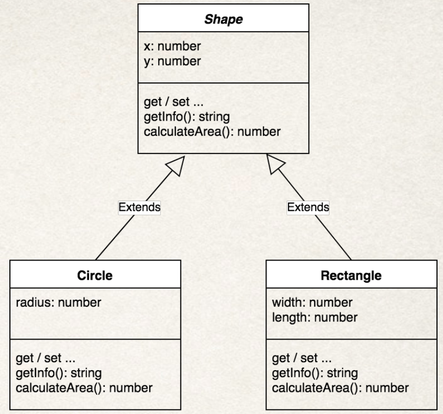
File: Shape.ts

```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

File: Circle.ts

```
import { Shape } from './Shape';

export class Circle extends Shape {

    // previous code ...

    calculateArea(): number {
        return Math.PI * Math.pow(this._radius, 2);
    }
}
```

Circle



File: Shape.ts

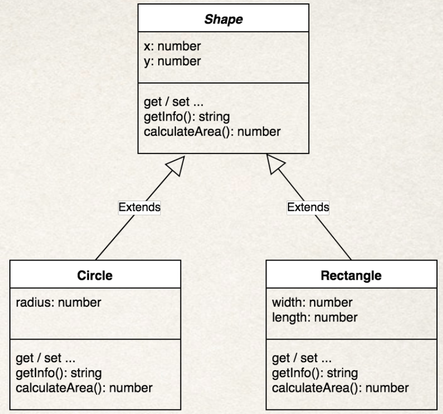
```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

Override the calculateArea() method

File: Circle.ts

```
import { Shape } from './Shape';

export class Circle extends Shape {

    // previous code ...

    calculateArea(): number {
        return Math.PI * Math.pow(this._radius, 2);
    }
}
```

Circle



File: Shape.ts

```ts
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

File: Circle.ts

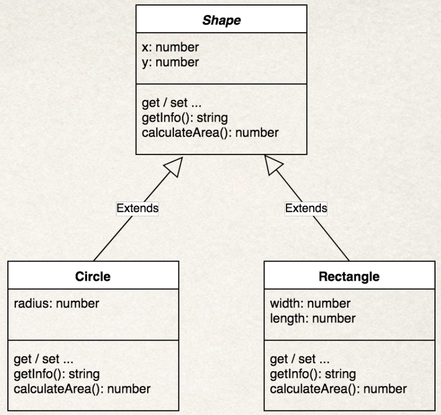
```ts
import { Shape } from './Shape';

export class Circle extends Shape {

    // previous code ...

    calculateArea(): number {
        return Math.PI * Math.pow(this._radius, 2);
    }
}
```

Override the calculateArea() method

Area of circle
$\pi * r^2$

Circle

`Math` is a built-in object that has properties and methods for mathematical constants and functions



File: Shape.ts

```
export abstract class Shape {

    // previous code ...

    abstract calculateArea(): number;
}
```

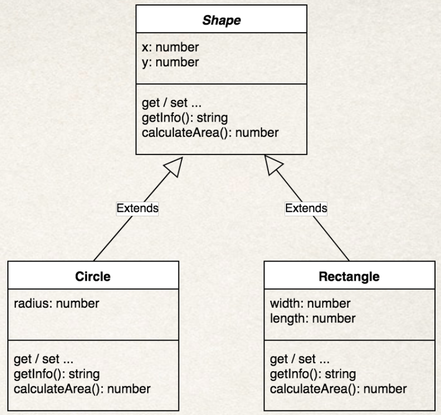
File: Circle.ts

```
import { Shape } from './Shape';

export class Circle extends Shape {

    // previous code ...

    calculateArea(): number {
        return Math.PI * Math.pow(this._radius, 2);
    }
}
```

Override the calculateArea() method

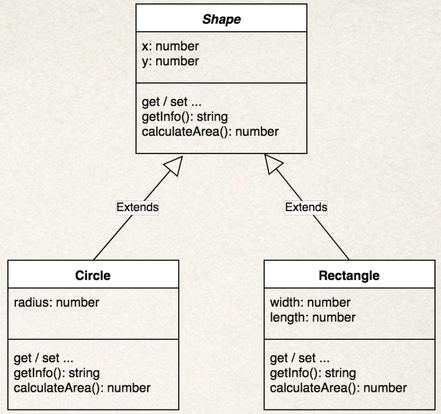
Area of circle
$\pi * r^2$

Creating an Instance

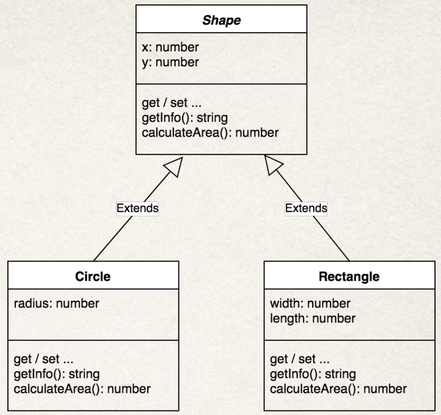


Creating an Instance

```
let myShape = new Shape(10, 15);
console.log(myShape.getInfo());
```



Creating an Instance



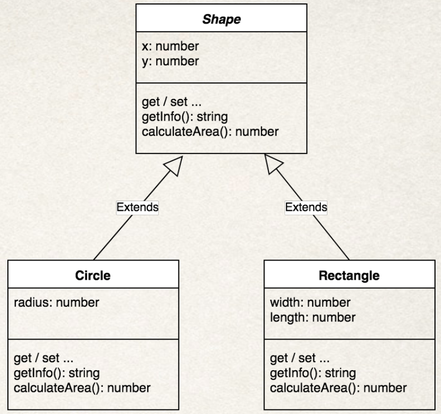

```
let myShape = new Shape(10, 15);
console.log(myShape.getInfo());
```

This will NOT compile since Shape is an abstract class

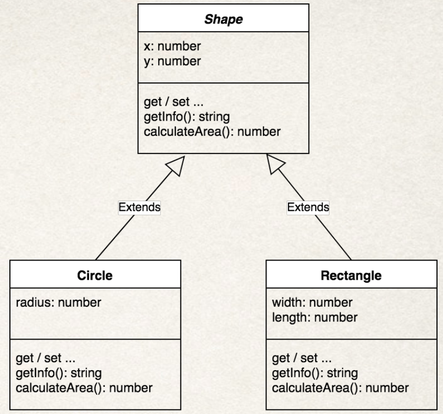
Can't create instance of abstract class directly

Only concrete subclasses: Circle, Rectangle, …

Creating an Array of Shapes



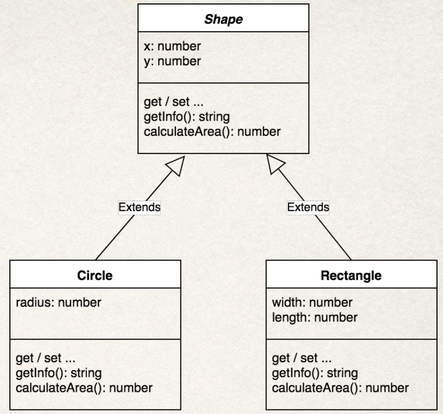
Creating an Array of Shapes



File: ArrayDriver.ts

```
… …

let myCircle = new Circle(5, 10, 20);
let myRectangle = new Rectangle(0, 0, 3, 7);

// declare an array for shapes … initially empty
let theShapes: Shape[] = [];

// add the shapes to the array
theShapes.push(myCircle);
theShapes.push(myRectangle);
```

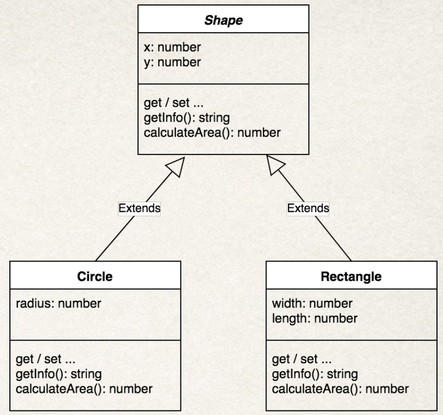
Creating an Array of Shapes



File: ArrayDriver.ts

```
… …

let myCircle = new Circle(5, 10, 20);
let myRectangle = new Rectangle(0, 0, 3, 7);

// declare an array for shapes … initially empty
let theShapes: Shape[] = [];

// add the shapes to the array
theShapes.push(myCircle);
theShapes.push(myRectangle);

for (let tempShape of theShapes) {
    console.log(tempShape.getInfo());
    console.log("Area=" + tempShape.calculateArea());
    console.log();
}
```

Creating an Array of Shapes

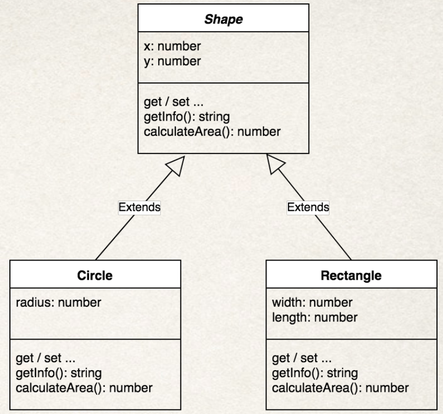


File: ArrayDriver.ts

```
… …

let myCircle = new Circle(5, 10, 20);
let myRectangle = new Rectangle(0, 0, 3, 7);

// declare an array for shapes … initially empty
let theShapes: Shape[] = [];

// add the shapes to the array
theShapes.push(myCircle);
theShapes.push(myRectangle);

for (let tempShape of theShapes) {
    console.log(tempShape.getInfo());
    console.log("Area=" + tempShape.calculateArea());
    console.log();
}
```

```
x=5, y=10, radius=20
Area=1256.6370614359173

x=0, y=0, width=3, length=7
Area=21
```

Creating an Array of Shapes



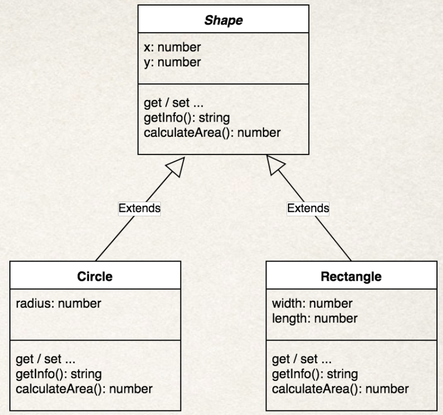
File: ArrayDriver.ts

```
… …

let myCircle = new Circle(5, 10, 20);
let myRectangle = new Rectangle(0, 0, 3, 7);

// declare an array for shapes … initially empty
let theShapes: Shape[] = [];

// add the shapes to the array
theShapes.push(myCircle);
theShapes.push(myRectangle);

for (let tempShape of theShapes) {
    console.log(tempShape.getInfo());
    console.log("Area=" + tempShape.calculateArea());
    console.log();
}
```

Area of circle
$\pi * r^2$

```
x=5, y=10, radius=20
Area=1256.6370614359173

x=0, y=0, width=3, length=7
Area=21
```

Creating an Array of Shapes



File: ArrayDriver.ts

```
… …

let myCircle = new Circle(5, 10, 20);
let myRectangle = new Rectangle(0, 0, 3, 7);

// declare an array for shapes … initially empty
let theShapes: Shape[] = [];

// add the shapes to the array
theShapes.push(myCircle);
theShapes.push(myRectangle);

for (let tempShape of theShapes) {
    console.log(tempShape.getInfo());
    console.log("Area=" + tempShape.calculateArea());
    console.log();
}
```

```
x=5, y=10, radius=20
Area=1256.6370614359173

x=0, y=0, width=3, length=7
Area=21
```

Area of circle
$\pi * r^2$

Area of rectangle
width * length