luv2code

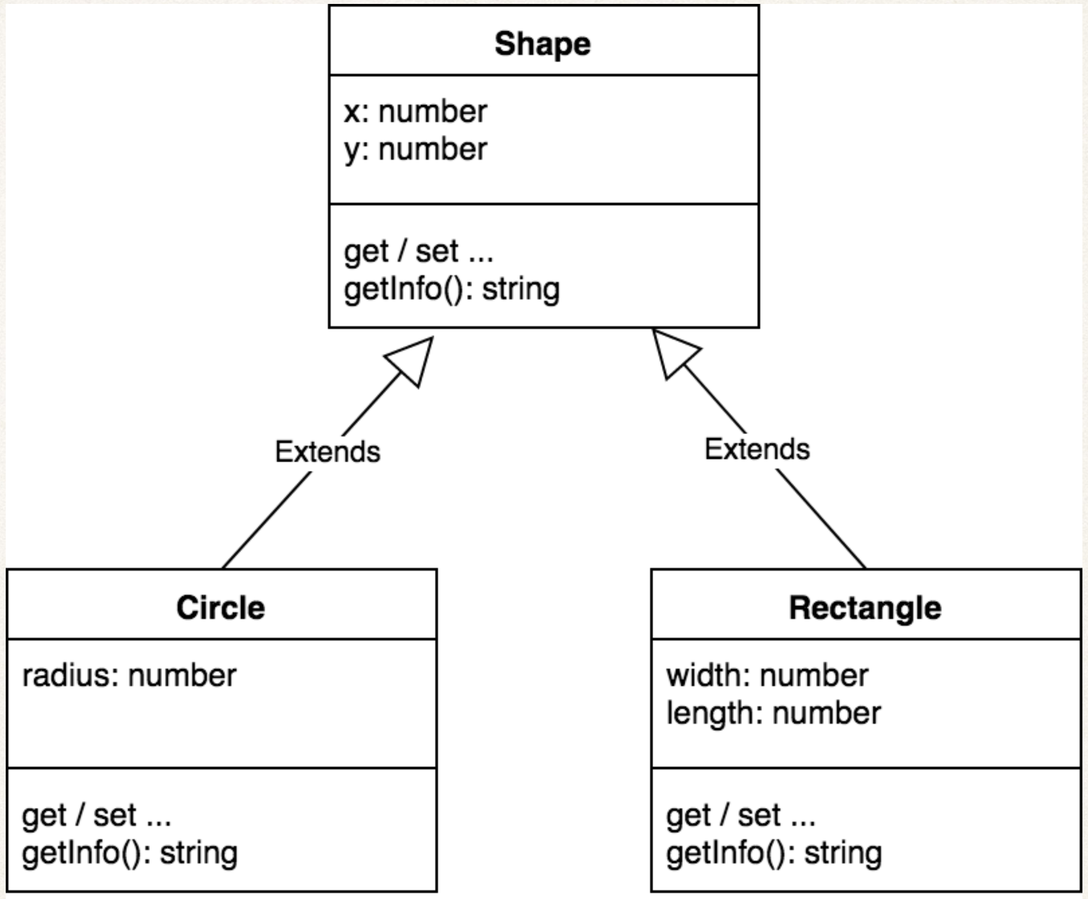
Inheritance

Inheritance

- TypeScript supports the object-oriented concept of inheritance
  - Define common properties and methods in the superclass
  - Subclasses can extend superclasses and add properties and methods
  - Support for abstract classes and overriding

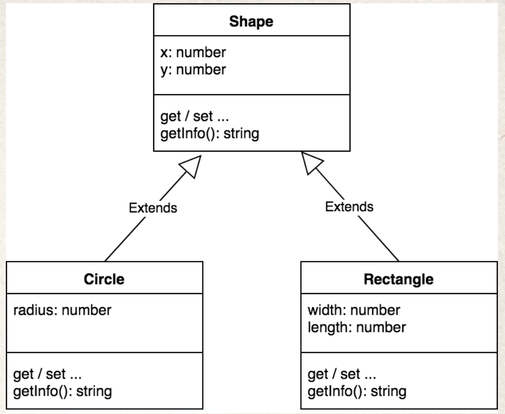
- TypeScript only supports single inheritance
  - However, you can implement multiple interfaces

Inheritance Example



Can override the getInfo() method

Inheritance Example



File: Shape.ts

```
export class Shape {

    constructor(private _x: number, private _y: number) {
    }

    // get/set accessors ...

    getInfo(): string {
        return `x=${this._x}, y=${this._y}`;
    }
}
```

Parameter Properties

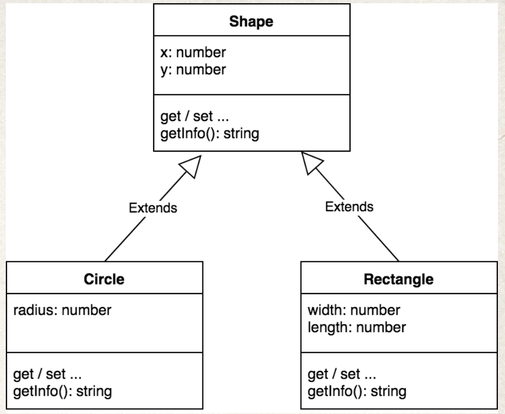
Inheritance

File: Circle.ts

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

export class Circle extends Shape {

    constructor(theX: number, theY: number,
                private _radius: number) {
        super(theX, theY);
    }

    // get/set accessors ...

    getInfo(): string {
        return super.getInfo() + `, radius=${this._radius}`;
    }
}
```

Regular parameters `theX` and `theY`

Inheritance Example



File: Shape.ts

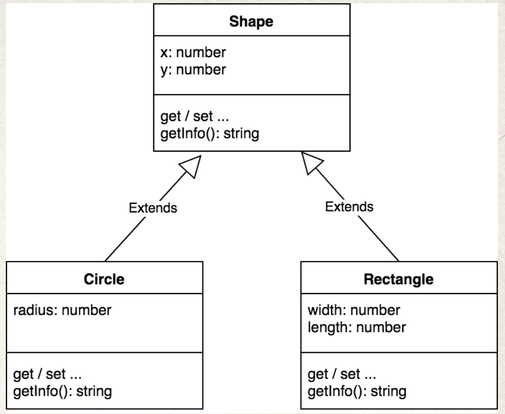
```
export class Shape {

    constructor(private _x: number, private _y: number) {
    }

    // get/set accessors ...

    getInfo(): string {
        return `x=${this._x}, y=${this._y}`;
    }
}
```

File: Circle.ts

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

export class Circle extends Shape {

    constructor(theX: number, theY: number,
                private _radius: number) {
        super(theX, theY);
    }

    // get/set accessors ...

    getInfo(): string {
        return super.getInfo() + `, radius=${this._radius}`;
    }
}
```

Call superclass constructor

Inheritance Example



File: Shape.ts

```
export class Shape {

    constructor(private _x: number, private _y: number) {
    }

    // get/set accessors ...

    getInfo(): string {
        return `x=${this._x}, y=${this._y}`;
    }
}
```

File: Circle.ts

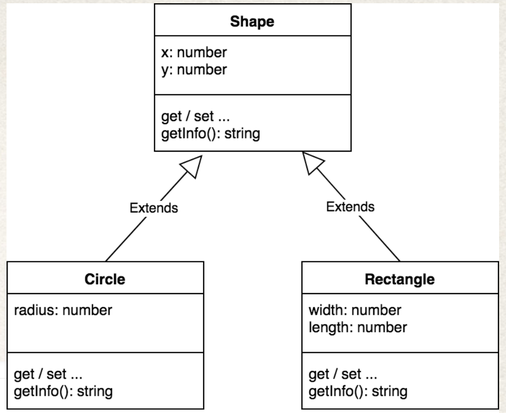
```
import { Shape } from './Shape';

export class Circle extends Shape {

    constructor(theX: number, theY: number,
                private _radius: number) {
        super(theX, theY);
    }

    // get/set accessors ...

    getInfo(): string {
        return super.getInfo() + `, radius=${this._radius}`;
    }
}
```

Override the getInfo() method

Call superclass method

Parameter Property _radius

Creating a main app



File: Shape.ts

```
export class Shape {
    …
    getInfo() {
        return `x=${this._x}, y=${this._y}`;
    }
}
```

File: Circle.ts

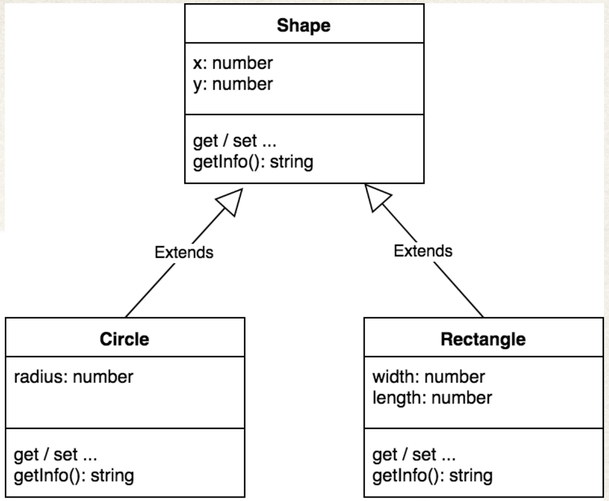
```
import { Shape } from './Shape';

export class Circle extends Shape {
    …
    getInfo() {
        return super.getInfo() + `, radius=${this._radius}`;
    }
}
```

File: Driver.ts

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

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

let myCircle = new Circle(5, 10, 20);
console.log(myCircle.getInfo());
```

```
x=10, y=15
x=5, y=10, radius=20
```

Rectangle



File: Rectangle.ts

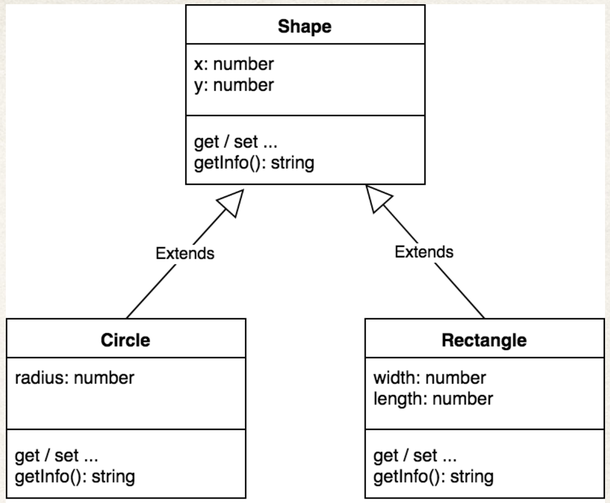
```
import { Shape } from './Shape';

export class Rectangle extends Shape {

    constructor(theX: number, theY: number,
                private _width: number, private _length: number) {

        super(theX, theY);
    }

    // get/set accessors ...

    getInfo(): string {
        return super.getInfo() + `, width=${this._width}, length=${this._length}`;
    }
}
```

Regular parameters `theX` and `theY`

Call superclass constructor

Parameter Properties `_width` and `_length`

Creating a main app



File: Driver.ts

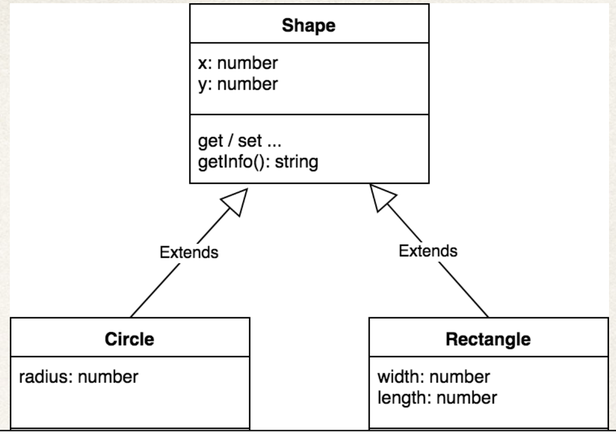
```
import { Shape } from './Shape';
import { Circle } from './Circle';
import { Rectangle } from './Rectangle';

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

let myCircle = new Circle(5, 10, 20);
console.log(myCircle.getInfo());
```

```
x=10, y=15
x=5, y=10, radius=20
x=0, y=0, width=3, length=7
```

Creating an Array of Shapes

File: ArrayDriver.ts

```
… …

let myShape = new Shape(10, 15);
let myCircle = new Circle(5, 10, 20);
let myRectangle = new Rectangle(0, 0, 3, 7);

// declar  array  shapes … initially empty
let theSha   Sh   = [];

// add the s    to the array
theShapes.    ape);
theShape     (    le);
theShape     ush(my ectangle);

for (let tempShape of theShapes) {
    console.log(tempShape.getInfo());
}
```




```
// declare an array for shapes
let theShapes: Shape[] = [myShape, myCircle, myRectangle];
```

```
x=10, y=15
x=5, y=10, radius=20
x=0, y=0, width=3, length=7
```