Dynamic Polymorphism

In this lesson, you will learn about the concepts of dynamic polymorphism.

WE'LL COVER THE FOLLOWING

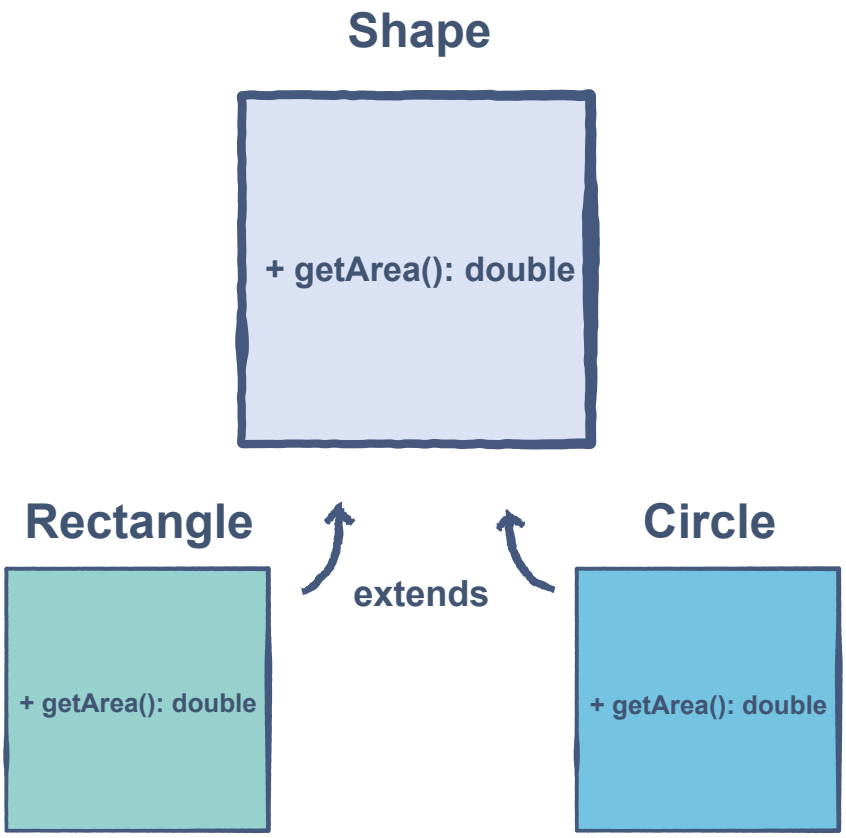
What is Dynamic Polymorphism?

> **Dynamic polymorphism** is the mechanism by which methods can be defined with the same name, return type, and parameters in the base class and derived classes.

The call to an overridden method is decided at the runtime.

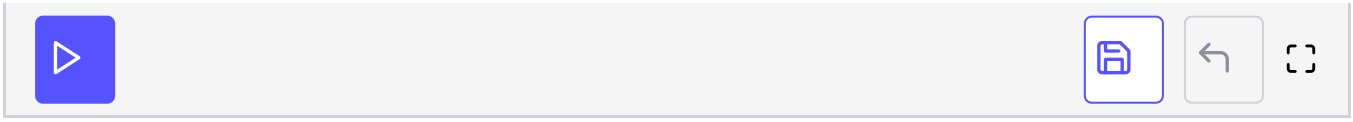
Dynamic Polymorphism Example

Let's consider the example of the `Shape` class:




```
// Shape Class
class Shape {

  public double getArea() {
    return 0;
  }

}

// A Rectangle is a Shape
class Rectangle extends Shape {  // extended form the Shape class

  private double length;
  private double width;

  public Rectangle(double length, double width) {
    this.length = length;
    this.width = width;
  }

  public double getArea() {
    return this.width * this.length;
  }

}

// A Circle is a Shape
class Circle extends Shape {

  private double radius;
```

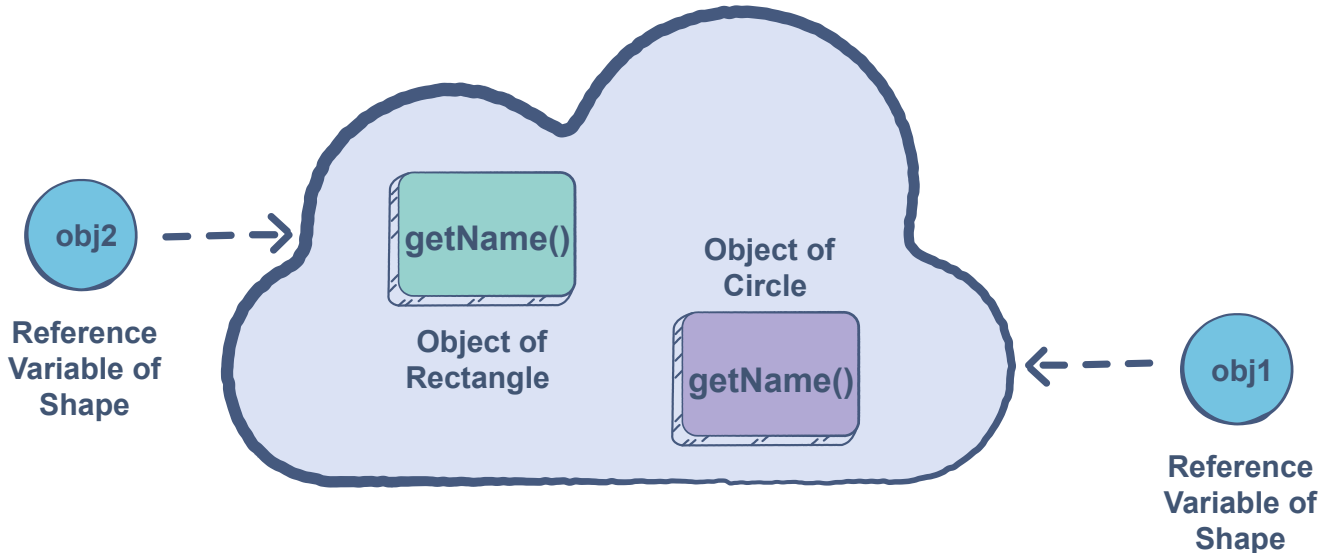
```
  public Circle(double radius) {
    this.radius = radius;
  }

  public double getArea() {
    return 3.13 * this.radius * this.radius;
  }


  public static void main(String args[]) {
    Shape[] shape = new Shape[2]; // Creating the shape array of size 2

    shape[0] = new Circle(3); // creating the circle object at index 0
    shape[1] = new Rectangle(2, 3); // creating the rectangle object at index 1

    System.out.println("Area of Circle: " + shape[0].getArea());
    System.out.println("Area of Rectangle: " + shape[1].getArea());
  }

}
```



A reference variable of the base class can be referred to the derived classes objects:

```
Shape obj1 = new Circle(3);
Shape obj2 = new Rectangle(2, 3);

//.
//.
//.

obj1.getName();
obj2.getName();
```

Here, the reference variables **obj1** and **obj2** are of the `Shape` class, but they are pointing to the `Circle` and `Rectangle` respectively.



Explanation

- `obj1.getName()` will execute `getName()` method of the subclass `Circle` class.
- `obj2.getName()` will execute `getName()` method of the subclass `Rectangle` class.
- `obj1` is a reference to the `Circle` class, it calls the method of `Circle` class, as it points to a *Circle* object.
- `obj2` is a reference to the `Rectangle` class, it calls the method of `Rectangle` class, as it points to a *Rectangle* object.

This is decided during runtime and is, therefore, called **dynamic** or **runtime** polymorphism.

Now that we are familiar with the concept of Dynamic Polymorphism let's understand the difference between dynamic and static polymorphism in the next lesson.