Abstraction in OOP

Abstraction simplifies complex systems by focusing on essential properties and behaviors while ignoring unnecessary details. It allows developers to create models that represent real-world entities or concepts without getting bogged down by every intricate aspect. Abstraction involves creating abstract classes or interfaces that define a blueprint for concrete classes, providing a high-level overview without specifying implementation details.

Example

Consider the concept of a `Shape` in a drawing application. Abstracting the idea of a shape allows us to define common properties and behaviors shared by various shapes without worrying about specific details for each shape. In this example, we'll create an abstract `Shape` class in TypeScript:

```
abstract class Shape {
  protected color: string;

  constructor(color: string) {
    this.color = color;
  }

  // Abstract method for calculating area
  abstract calculateArea(): number;

  // Common method to display information
  displayInfo(): void {
    console.log(`This ${this.color} shape has an area of
    ${this.calculateArea()} square units.`);
  }
}
```

In this abstract `Shape` class, we define a property `color` and an abstract method `calculateArea()`. The `displayInfo()`method provides a common way to display information about any shape. Now, let's create concrete classes for specific shapes:

```
class Circle extends Shape {
  constructor(color: string, private radius: number) {
    super(color);
  }

  calculateArea(): number {
    return Math.PI * this.radius ** 2;
  }
}

class Rectangle extends Shape {
  constructor(color: string, private width: number,
              private height: number) {
    super(color);
  }

  calculateArea(): number {
    return this.width * this.height;
  }
}
```

In these concrete classes (`Circle` and `Rectangle`), we implement the `calculateArea()` method specific to each shape. By abstracting the common properties and methods into the `Shape` class, we've simplified the representation of shapes, focusing on their essential characteristics. Abstraction facilitates code organization, reuse, and the creation of easily extendable systems.
