src\abstractclass.ts

```
//* Abstract classes provide a way to define common properties and methods that multiple derived classes can share. This promotes code reuse and helps establish a common interface for related classes.
//* abstract class cannot be instantiated

//* abstract classes focus on class inheritance and sharing common functionality,
//* whereas the useContext hook in React focuses on managing global state and allowing components to consume that state.

abstract Class PerObj {
  name: string;
  age: number;
}

class Person: PerObj = {
  name: "vinod",
  age: 29,
};
class Person1: PerObj = {
  name: "thapa",
  age: 29,
};
class Person3: PerObj = {
  name: "thapa",
  age: 29,
};

//? Example: Shape Hierarchy
//? Suppose you're building a graphics application, and you want to create a hierarchy of different shapes. You can use an abstract base class Shape to define common properties and methods that all shapes share.

// circle , rectangle

// SUBSCRIBE TO THAPA TECHNICAL

abstract class Shape {
  constructor(protected color: string) {}
  abstract calculateArea(): number;
  abstract displayArea: () => void;
}

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

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

  displayArea = () => {
    console.log(`This is a ${this.color} circle with radius ${this.radius}.`);
  };
}

const circle = new Circle("red", 5);
console.log(circle.calculateArea());
circle.displayArea;

//* Practice Time
//? You need to do the same for Square and Rectangle and if possible use getter and setter methods or static members

```