OOPs Concepts in TypeScript

1. Class and Object

A class is a blueprint, and an object is an instance of a class.

Example:

```
class Person {
  name: string;
  age: number;

  constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }

  greet(): void {
    console.log(`Hello, my name is ${this.name}`);
  }
}

const person1 = new Person("Alice", 25);
person1.greet(); // Output: Hello, my name is Alice
```

2. Encapsulation

Encapsulation hides the internal state of the object and allows access only through public methods.

Example:

```
class BankAccount {
  private balance: number = 0;

  deposit(amount: number) {
    if (amount > 0) {
      this.balance += amount;
    }
  }

  getBalance(): number {
    return this.balance;
  }
}

const account = new BankAccount();
account.deposit(1000);
console.log(account.getBalance()); // Output: 1000
```

3. Inheritance

A class can inherit properties and methods from another class using the `extends`

keyword.

Example:

```
class Animal {
  makeSound(): void {
    console.log("Animal sound");
  }
}

class Dog extends Animal {
  makeSound(): void {
    console.log("Bark");
  }
}

const dog = new Dog();
dog.makeSound(); // Output: Bark
```

4. Abstraction

Abstract classes define structure without implementation.

Example:

```
abstract class Shape {
  abstract area(): number;

  printArea(): void {
    console.log("Area:", this.area());
  }
}

class Circle extends Shape {
  constructor(private radius: number) {
    super();
  }

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

const circle = new Circle(5);
circle.printArea();
```

5. Polymorphism

Polymorphism allows methods to behave differently based on the object.

Example:

OOPs Concepts in TypeScript

```
class Vehicle {
  move(): void {
    console.log("Vehicle is moving");
  }
}

class Car extends Vehicle {
  move(): void {
    console.log("Car is driving");
  }
}

class Bicycle extends Vehicle {
  move(): void {
    console.log("Bicycle is pedaling");
  }
}

const vehicles: Vehicle[] = [new Car(), new Bicycle()];
vehicles.forEach(v => v.move());
```

6. Access Modifiers

```
public - Accessible from anywhere (default)
private - Accessible only within the class
protected - Accessible within the class and subclasses

Example:
class Test {
  public x = 1;
  private y = 2;
  protected z = 3;
}
```