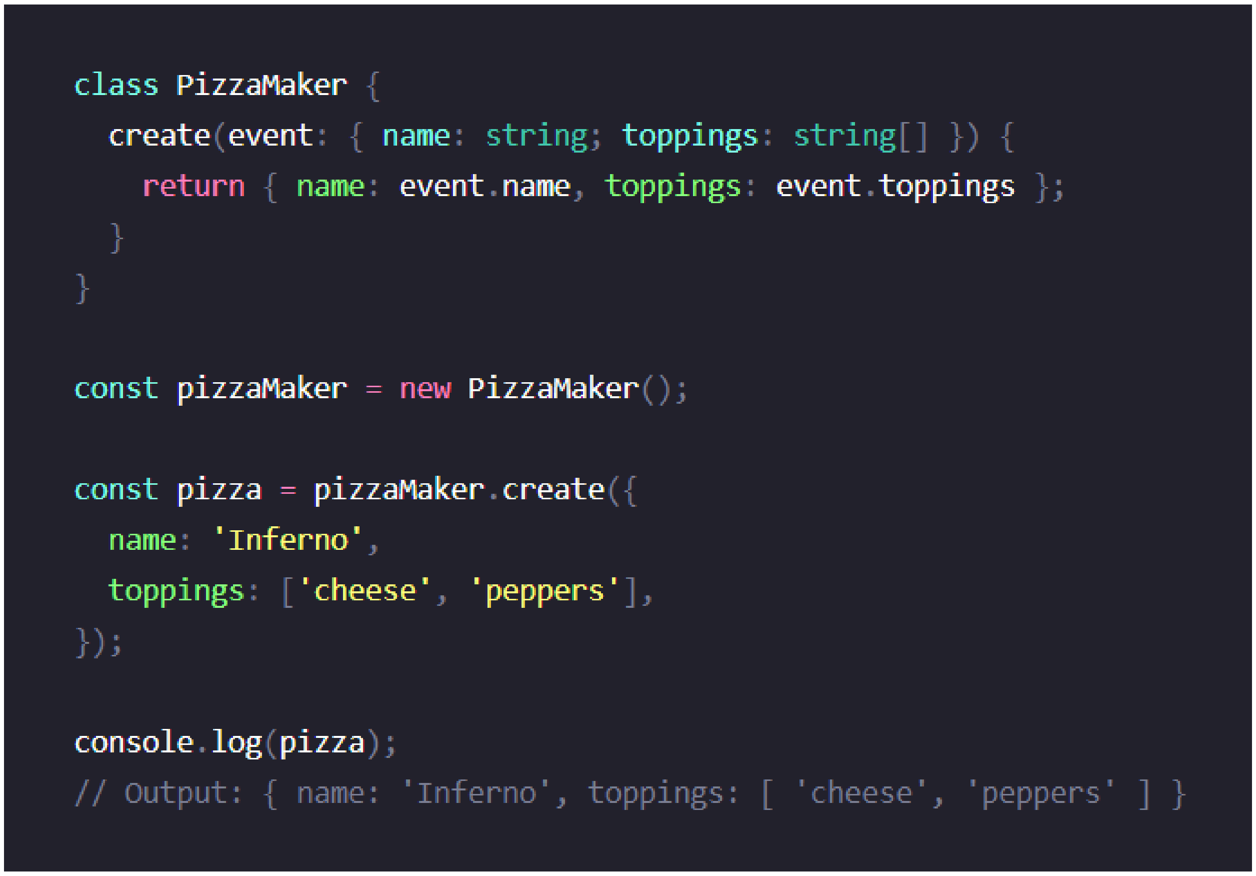
Classes Vs Abstract Classes Interfaces

Classes:

Typescript only started to include classes from ECMAScript 2015 despite the importance of object oriented programming, but it was finally introduced giving the benefits of OOP including encapsulation and abstraction. Classes have the ability to use reusable components giving a template with constructors, methods, interfaces and modifiers.

Example:



```
class PizzaMaker {
  create(event: { name: string; toppings: string[] }) {
    return { name: event.name, toppings: event.toppings };
  }
}

const pizzaMaker = new PizzaMaker();

const pizza = pizzaMaker.create({
  name: 'Inferno',
  toppings: ['cheese', 'peppers'],
});

console.log(pizza);
// Output: { name: 'Inferno', toppings: [ 'cheese', 'peppers' ] }
```

Abstract Classes:

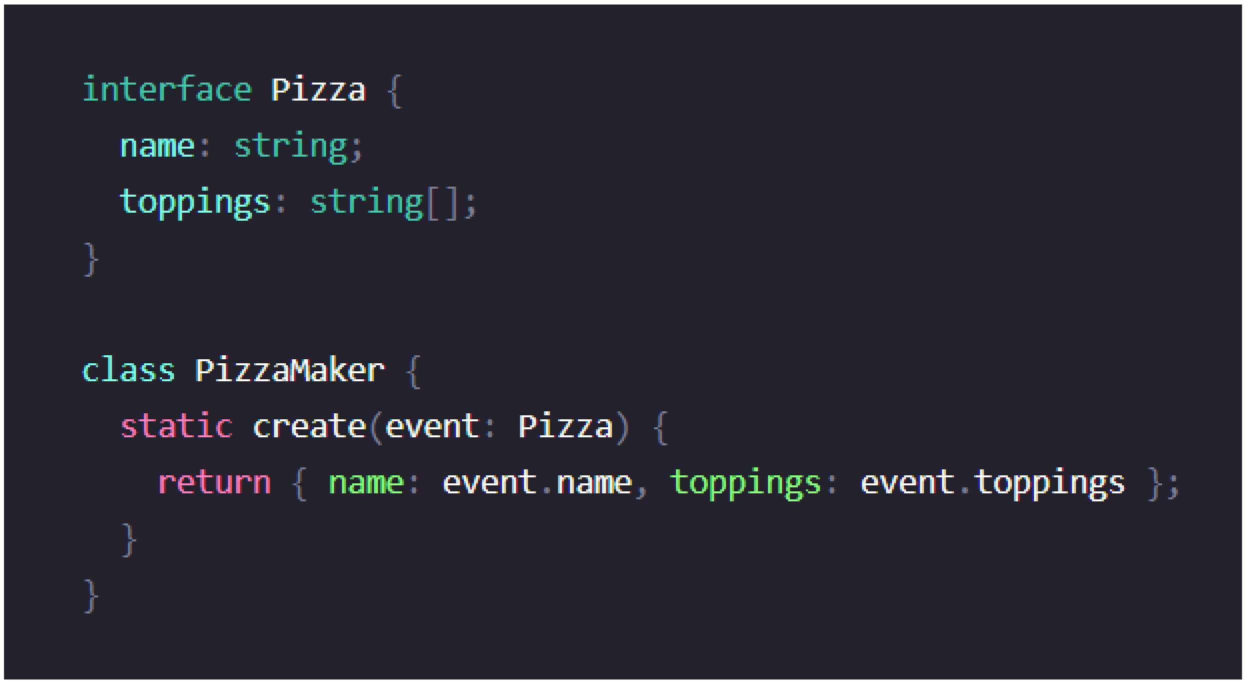
Abstract classes are the same as classes but are only used for inheritance as a blueprint for other classes to use without having to implement everything. This is done by using the “abstract” keyword when defining the class or method.

Example:

```typescript
abstract class Person {
    name: string;

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

    display(): void{
        console.log(this.name);
    }

    abstract find(string): Person;
}

class Employee extends Person {
    empCode: number;

    constructor(name: string, code: number) {
        super(name); // must call super()
        this.empCode = code;
    }

    find(name:string): Person {
        // execute AJAX request to find an employee from a db
        return new Employee(name, 1);
    }
}

let emp: Person = new Employee("James", 100);
emp.display(); //James

let emp2: Person = emp.find('Steve');
```

Interfaces:

Interfaces are structures that are similar to classes but they act as a contract that the other classes must follow and implement the structure given in the interface using it as a type checker. It's defined by using the word “interface” just like with using classes.



```
interface Pizza {
  name: string;
  toppings: string[];
}

class PizzaMaker {
  static create(event: Pizza) {
    return { name: event.name, toppings: event.toppings };
  }
}
```

Comparison:

There aren’t a lot of differences between classes and abstract classes apart from the word “abstract being used” when defining an abstract class and you can not create instances of the abstract class only extend from it.

While between classes and interfaces there are several differences including:

- Classes are used to create reusable components while interfaces just define a structure only containing declarations with nothing initialized and no body.
- You can’t instantiate an interface unlike the class.
- Members of the class can be private, public, or protected while members of interfaces are always public.
- A class can only extend one class while implementing any number of interfaces while interfaces can extend more than one interface.

References:

- https://www.javatpoint.com/typescript-classes
- https://www.javatpoint.com/typescript-class-vs-interface#:~:text=TypeScript%20class%20vs.,-TypeScript%20Interface&text=Classes%20are%20the%20fundamental%20entities%20used%20to%20create%20reusable%20components,a%20contract%20in%20our%20application.
- https://ultimatecourses.com/blog/classes-vs-interfaces-in-typescript
- https://www.tutorialsteacher.com/typescript/typescript-class
- https://www.tutorialsteacher.com/typescript/abstract-class