**Node.js**

Task

**TypeScript Class**

Classes method is used to create and easier option for reusing items in several objects. A class acts as a container, this container has inputs in a constructor and function logics as methods, both to be used for any variable that follows this class.

class **Pizza** {

constructor(public **name** : *string*, public **components** : *string[]*);

}

Let **pepperoni** = new **Pizza(***“Pepperoni pizza”,[“cheese”, ”pepperoni”***)**

In this example, a new item was created (pepperoni) that follows the class pizza, so automatically the variable receives the inputs as a string for ‘name’ and an array of strings for ‘components’, since the types were set in the class.

**TypeScript Interface**

Interfaces are used to put a structure for the class. This means that for items in a class set with a type of the created interface, the class will automatically expect the same attributes stated in the interface

Interface **PizzaItems** {

**name** :  *string*,

**components** :  *string[]*

class **Pizza** {

constructor(**create :** *PizzaItems*);

}

Let **pepperoni** = new **Pizza(***“Pepperoni pizza”,[“cheese”, ”pepperoni”***)**

In this example, a new item was created (pepperoni) that follows the class pizza, since ‘Pizza’ uses the interface ‘PizzaItems’ it allowed the constructor to receive the structure set in the interface.

**TypeScript Abstract class**

Abstract classes are normal classes that contain specific methods that can be called in other classes. The only difference between an abstract class and an interface is that an abstract class can still have functions on its own.

abstract class **Department** {

constructor ( public **name**: *string*) {

abstract **printMeeting()** : *void* ;

}

class **AccountingDepartment** extends **Department** {

constructor() {

super("Accounting and Auditing") ;

}

**printMeeting()** : *void* {

console.log("The Accounting Department meets each Monday at 10am.");

}

As shown in this example as long as one class extends an abstract class, all abstract functions can be recalled with the same logic.

**References**

*Difference between typescript class and interface - javatpoint (no date) www.javatpoint.com. Available at: https://www.javatpoint.com/typescript-class-vs-interface#:~:text=Classes%20are%20the%20fundamental%20entities%20used%20to%20create%20reusable%20components,a%20contract%20in%20our%20application. (Accessed: February 11, 2023).*

*Handbook - classes (2018) TypeScript. Available at:* "TypeScript: Handbook - Classes." <https://www.typescriptlang.org/docs/handbook/classes.html>.

*(Accessed: February 11, 2023).*

*Motto, T. (no date) Classes VS interfaces in typescript, Ultimate Courses. Available at:* <https://ultimatecourses.com/blog/classes-vs-interfaces-in-typescript>.

*(Accessed: February 11, 2023).*