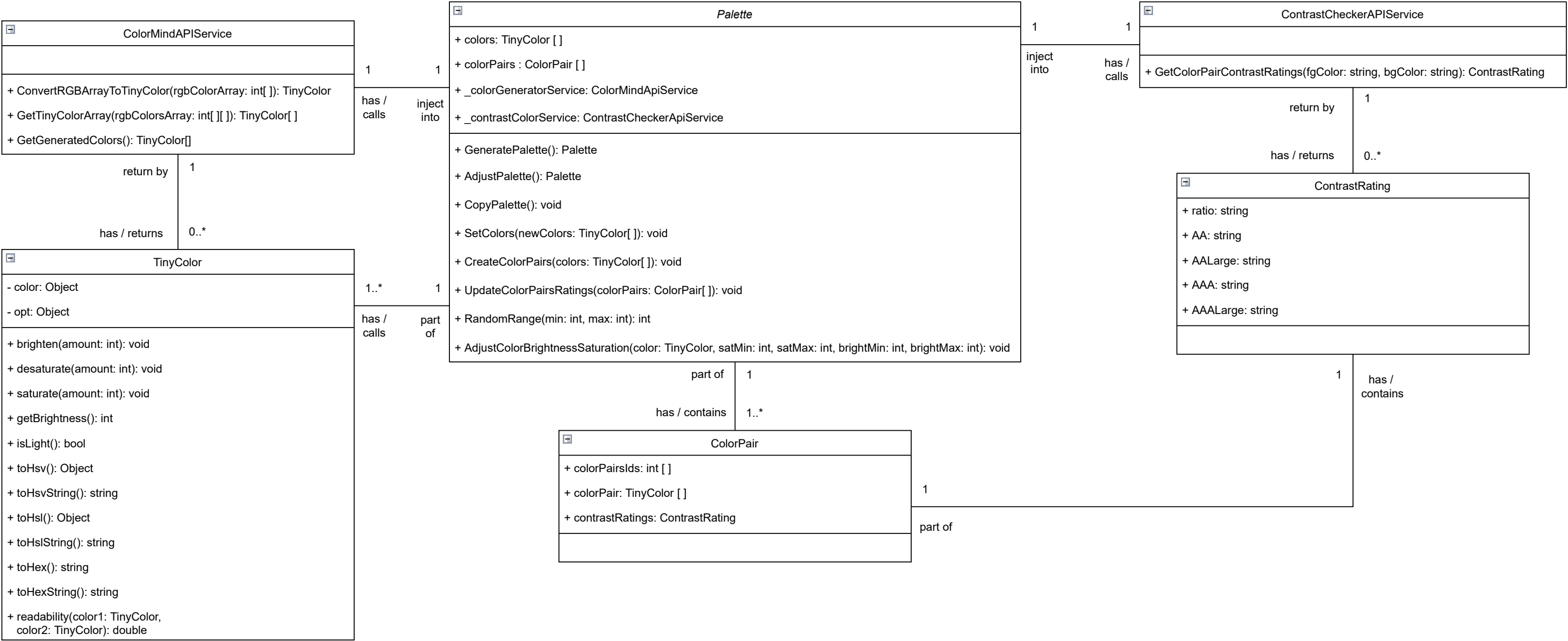


Class Diagram

Palette Class & Associated Classes

CD-1

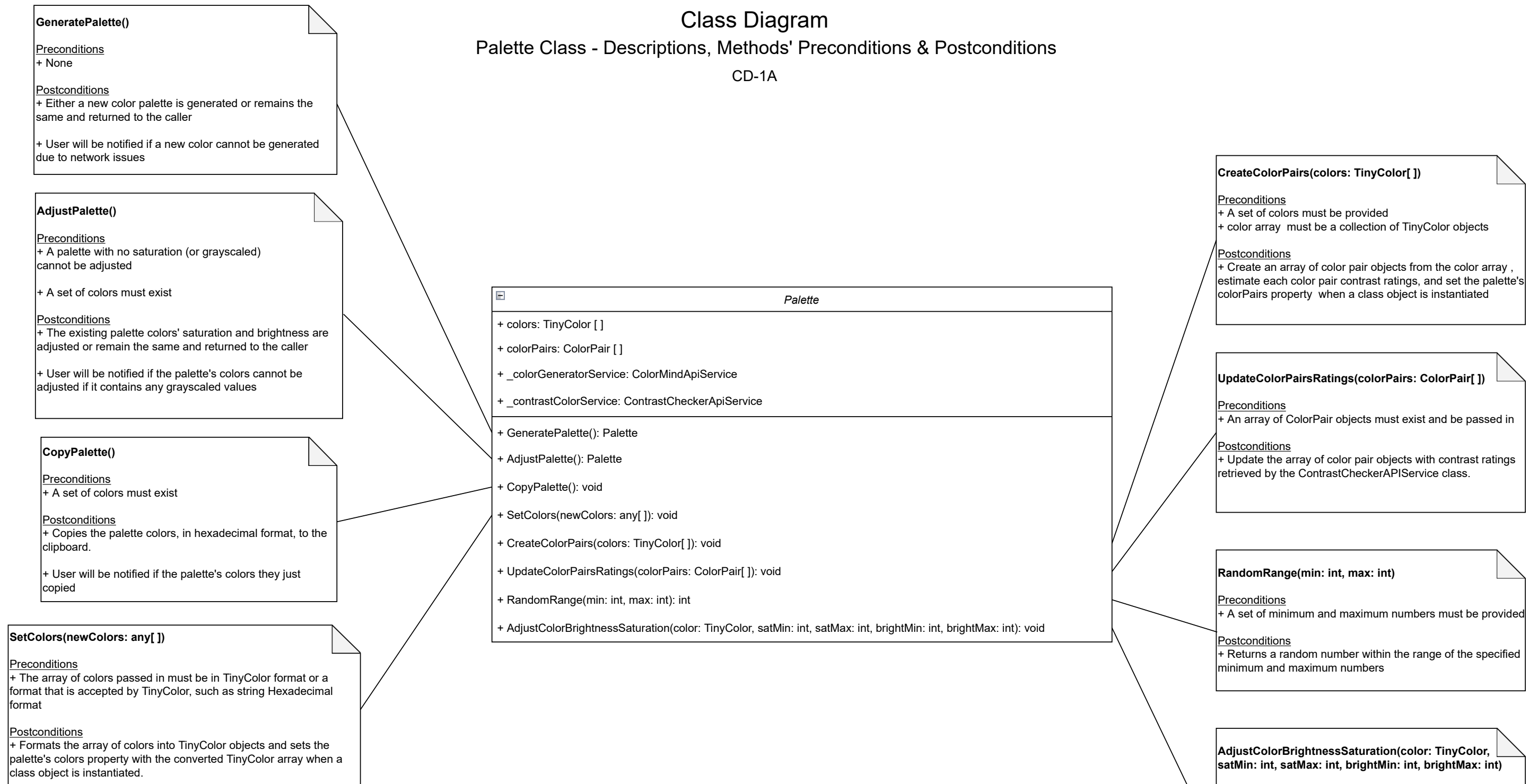


- Note:**
- 1) All JavaScript Class' properties and methods are public by default. The team did treat and use some as private within the class while some were accessed and called publicly once instantiated
- 2) TinyColor is a dependency library and class installed for this project
- 2a) For this diagram, only the TinyColor properties and methods used in this application are shown.
- 3) Due to the class diagram's size, each class method's preconditions and postconditions are shown on a separate page.
- 4) The object diagram is shown on a separate sheet due to the size of the class diagram.

Class Diagram

Palette Class - Descriptions, Methods' Preconditions & Postconditions

CD-1A



Additional Description: Palette Class / Object

The global Palette Class/Object is a critical piece of the application. It provides all the color and paired color data needed to hydrate the content and modify the HTML element styles of various components and pages within the application. Its properties are used as arguments for multiple methods and functions in the application, and its methods are passed to close child components as props.

A global Palette object is instantiated at the start of the application and stored in as a state using React's useState hook. The global Palette state is located in the top parent component and accessible to other child components through React's Context hook.

A palette context and context provider is created for the application at the top level, and the palette state tuple, [palette, SetPalette], is assigned as the context value property.

The Palette object in the state can be set/replaced anywhere in the application using the palette context and `setPalette()` method it holds, so long as the child component consumes the context. And any child components that consume the context will subscribe to its changes.

The Palette Class/Object itself cannot update the global state from within because calling hooks are forbidden in classes, so various Palette methods return a new Palette object to the caller so that a new palette state can be set globally.

brighten(amount: int): void

Preconditions
+ None

Postconditions
+ Adjusts the TinyColor object's brightness by a given amount. Brightness level will range from 0 to 100

desaturate(amount: int): void

Preconditions
+ None

Postconditions
+ Reduces the TinyColor object's saturation by a given amount. The saturation level will range from 0 to 100

saturate(amount: int): void

Preconditions
+ None

Postconditions
+ Increases the TinyColor object's saturation by a given amount. The saturation level will range from 0 to 100

getBrightness (): int

Preconditions
+ None

Postconditions
+ Returns the brightness value of the color from a range of 0 to 255.

isLight(): bool

Preconditions
+ None

Postconditions
+ Returns a boolean whether the brightness value of the color is considered "light."

toHsv(): Object

Preconditions
+ None

Postconditions
+ Returns an object with a h, s, and v property of float type.

toHsvString(): string

Preconditions
+ None

Postconditions
+ Returns the h, s, and v values as a string.

TinyColor

- color: Object

- opt: Object

+ brighten(amount: int): void

+ desaturate(amount: int): void

+ saturate(amount: int): void

+ getBrightness(): int

+ isLight(): bool

+ toHsv(): Object

+ toHsvString(): string

+ toHsl(): Object

+ toHslString(): string

+ toHex(): string

+ toHexString(): string

+ readability(color1: TinyColor, color2: TinyColor): double

Class Diagram

TinyColor Class - Descriptions, Methods' Preconditions & Postconditions

CD-1B

toHsl(): Object

Preconditions
+ None

Postconditions
+ Returns an object with a h, s, and l property of float type.

toHslString(): string

Preconditions
+ None

Postconditions
+ Returns the h, s, and l values as a string.

toHex(): string

Preconditions
+ None

Postconditions
+ Returns hexadecimal value as a string, without the "#" character

toHexString(): string

Preconditions
+ None

Postconditions
+ Returns hexadecimal value as a string, with the "#" character

readability(color1: TinyColor, color2: TinyColor): string

Preconditions
+ It requires the input of two colors of TinyColor types

Postconditions
+ Returns contrast ratio value as a double type

Additional Description: TinyColor Class / Object

The TinyColor class and object are from a library and dependency installed for the application.

The TinyColor class has a set of "setter" and "getter" methods that allow color conversation to different color models such as RGB, HSV, HSL, and HEX. It also allows various color adjustments, such as saturation, brightness, and hue. It can even determine the contrast ratio between two colors.

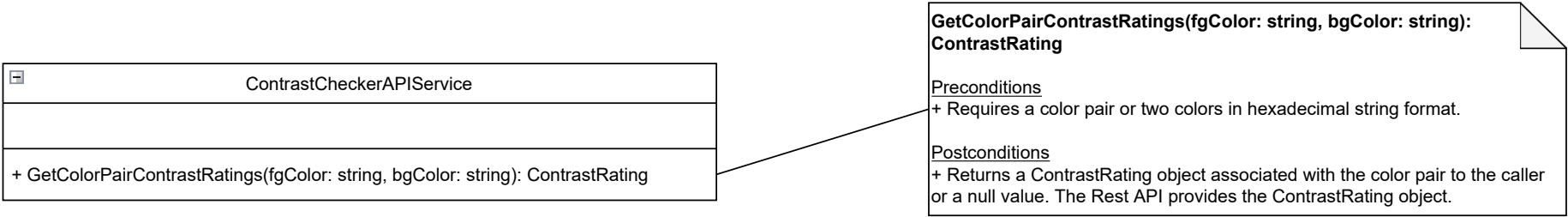
The TinyColor class/object is used as this project's color class/object and allowed the team to save some development time building a class with similar functionalities.

It is the property type of many classes in the application. It is also the parameter type and return type of many class methods.

Class Diagram

ContrastCheckerAPIService, ColorMindAPIService, ContrastRating, and ColorPair Classes - Descriptions, Methods' Preconditions & Postconditions

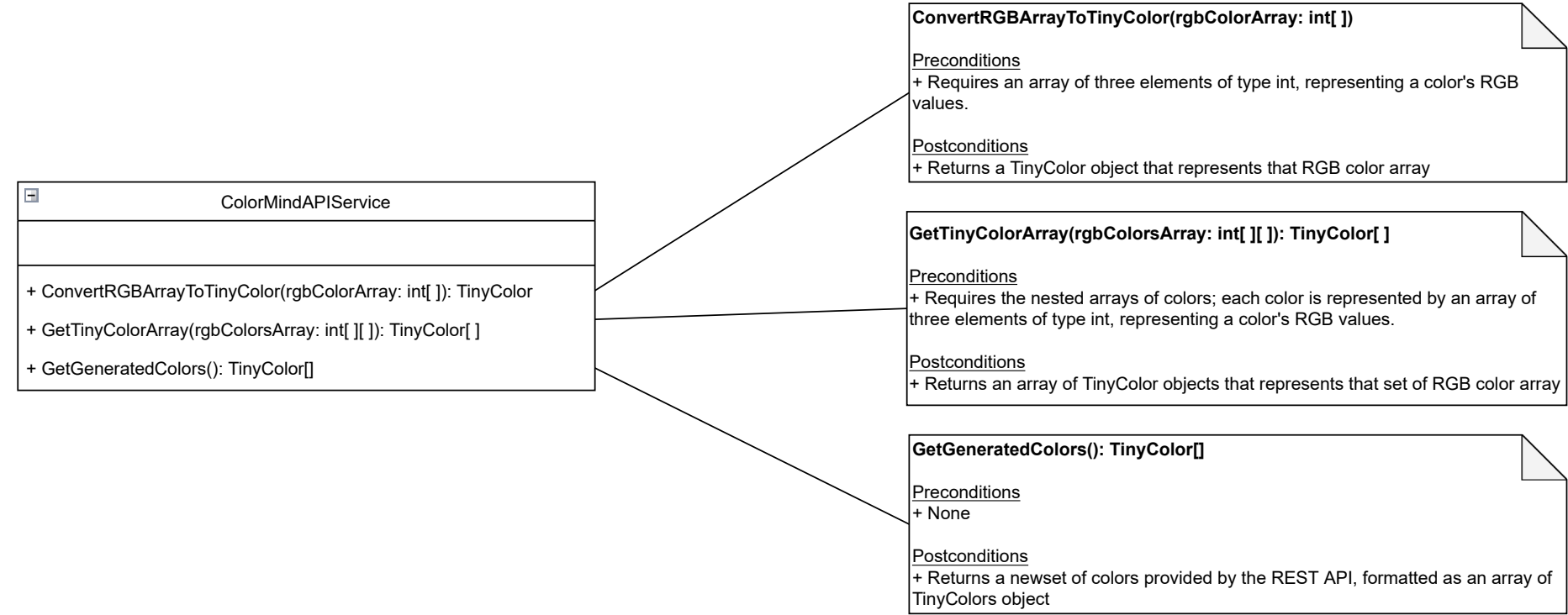
CD-1C



Additional Description: ContrastCheckerAPIService Class / Object

The ContrastCheckerAPIService class serves as an API Service layer. The ContrastCheckerAPIServices takes two colors in Hexidecimal strings, makes a GET HTTP Request with Axios to the Web Aim Contrast Checker REST API, and returns either a ContrastRating object if the request was successful or a null object if the request failed due to network issues. Depending on the object return, the caller will handle the outcome. The service logs any error on the console.

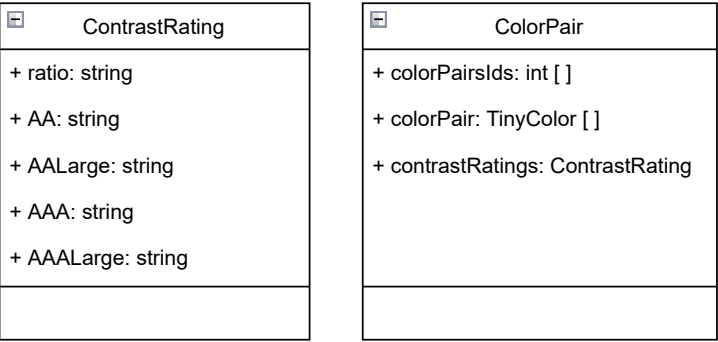
API calls performed and data retrieved in a separate layer ensure separation of concerns and code reusability by ensuring accessibility in other components or classes in the application that may require it.



Additional Description: ContrastCheckerAPIService Class / Object

The ColorMindAPIService class serves as an API Service layer. The ColorMindAPIService takes a predefined object and places it in a request body, making a POST HTTP Request with Axios to the Color Mind REST API. If successful, it converts the results into an array of TinyColor objects returned to the caller. If the request fails due to network issues, the service logs the error to the console and returns a null value to the caller. Depending on the object return, the caller will handle the outcome.

API calls performed and data retrieved in a separate layer ensure separation of concerns and code reusability by ensuring accessibility in other components or classes in the application that may require it.



Note: These classes contains no methods, and will not have any preconditions or postconditions

Additional Description: ContrastRating Class / Object

The ContrastRating class is a model class used to define a set of properties to model the WCAG contrast rating data sent by the Web Aim Contrast Checker API.

The ContrastRating class is a model class used to define the object's properties type.

This class defines the types of various class properties or method returns.

Additional Description: ColorPair Class / Object

The ColorPair class is a DTO class used to define a set of properties that holds information about two palette colors and their WCAG contrast ratings.

The ColorPair class defines the type for one of Palette's class property..

The properties in the ColorPair objects are used to hydrate and populate content within components and the Contrast Checker page.

Object Diagram

Palette Class & Associated Classes

OD-1

