To adding new implementation of new color system.

I have created the class that using that factory pattern called ColorSpaceFactory.
It can create the object that using the IColorSpace interface.

```
class ColorSpaceFactory {
  private static colorPool = [HSLColor, RGBColor];

  public static getColorSpace(type: ColorSpaceType): IColorSpace {
    switch (type) {
      case ColorSpaceType.HSL:
        return new HSLColor();
      case ColorSpaceType.RGB:
        return new RGBColor();
    }
  }

  public static randomColorSpace(): IColorSpace {
    const rand = randomInteger(this.colorPool.length);
    const color: IColorSpace = new this.colorPool[rand]();
    color.random();
    return color;
  }
}
```

Existed HSLColor and RGBColor classes had implements this interface.

```
export interface IColorSpace {
  random(): void;
  toResponse(): ColorSpaceResponse;
}
```

It also has some type configuration.

```
export interface HSLColorResponse {
  type: ColorSpaceType;
  hue: number;
  saturation: number;
  lightness: number;
}

export interface RGBColorResponse {
  type: ColorSpaceType;
  red: number;
  green: number;
  blue: number;
}

export type ColorSpaceResponse = HSLColorResponse | RGBColorResponse

export enum ColorSpaceType {
  HSL = 'hsl',
  RGB = 'rgb',
}
```

If other teams want to adding new color system, they can add new one by follow these steps:

1. Create new class (ex. BRGBColor) and implements IColorSpace interface.
2. Add new class into ColorSpaceFactory.
3. Add new type configuration.
4. Done.

Example.

I want to add new BRGB color system.

First, create new BRGBColor that implements IColorSpace.
(It also can extends RGBColor but I think it will tightly coupling)

```
class BRGBColor implements IColorSpace {
  private type = ColorSpaceType.BRGB;

  private MAX_RED = 10000;
  private MAX_GREEN = 10000;
  private MAX_BLUE = 10000;

  private red: number;
  private green: number;
  private blue: number;

  constructor(r = 0, g = 0, b = 0) {
    this.red = r;
    this.green = g;
    this.blue = b;
  }

  public random(): void {
    this.red = randomInteger(this.MAX_RED);
    this.green = randomInteger(this.MAX_GREEN);
    this.blue = randomInteger(this.MAX_BLUE);
  }

  public toResponse(): BRGBColorResponse {
    const res: BRGBColorResponse = {
      type: this.type,
      red: this.red,
      green: this.green,
      blue: this.blue
    };
    return res;
  }
}
```

In ColorSpaceFactory

```
class ColorSpaceFactory {
  private static colorPool = [HSLColor, RGBColor, BRGBColor];

  public static getColorSpace(type: ColorSpaceType): IColorSpace {
    switch (type) {
      case ColorSpaceType.HSL:
        return new HSLColor();
      case ColorSpaceType.RGB:
        return new RGBColor();
      case ColorSpaceType.BRGB:
        return new BRGBColor();
    }
  }
  ...
```

In type.

```
export interface BRGBColorResponse {
  type: ColorSpaceType;
  red: number;
  green: number;
  blue: number;
}

export type ColorSpaceResponse = HSLColorResponse | RGBColorResponse |
BRGBColorResponse;

export enum ColorSpaceType {
  HSL = 'hsl',
  RGB = 'rgb',
  BRGB = 'brgb'
}
```