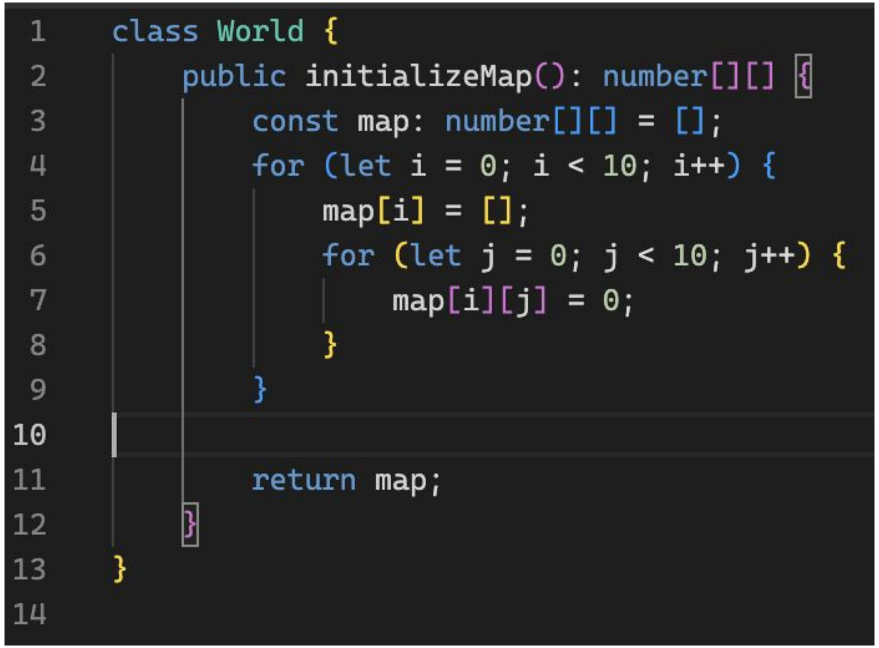
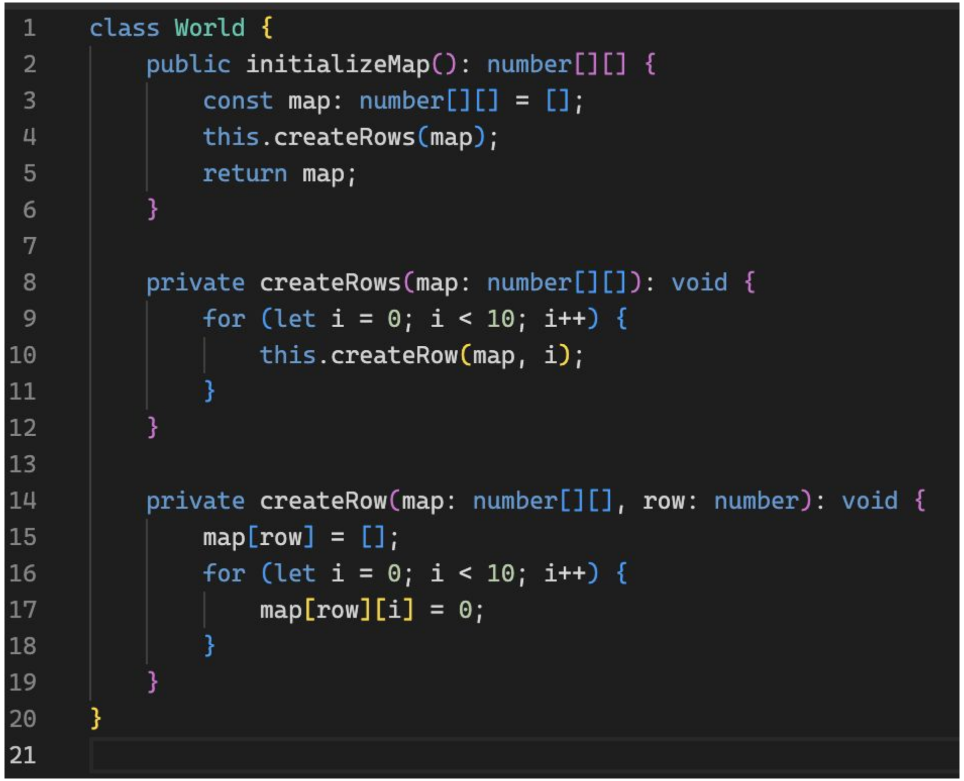
Object Calisthenics

One level indentation per method

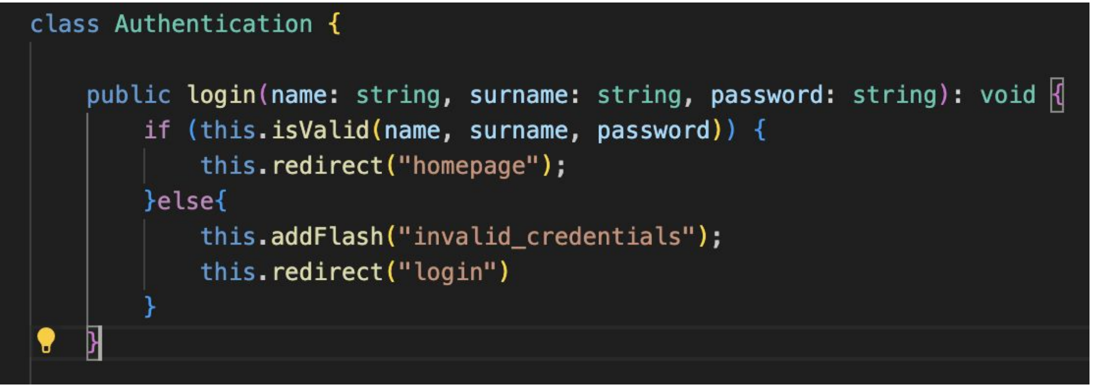


```
class World {
    public initializeMap(): number[][] {
        const map: number[][] = [];
        for (let i = 0; i < 10; i++) {
            map[i] = [];
            for (let j = 0; j < 10; j++) {
                map[i][j] = 0;
            }
        }

        return map;
    }
}

```

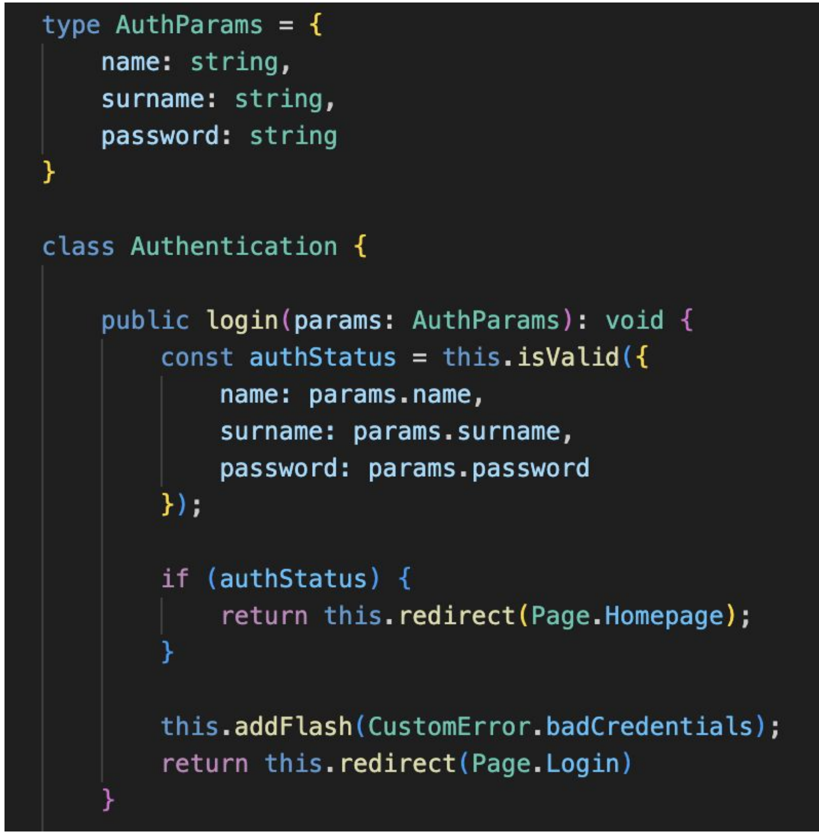


```
class World {
    public initializeMap(): number[][] {
        const map: number[][] = [];
        this.createRows(map);
        return map;
    }

    private createRows(map: number[][]): void {
        for (let i = 0; i < 10; i++) {
            this.createRow(map, i);
        }
    }

    private createRow(map: number[][], row: number): void {
        map[row] = [];
        for (let i = 0; i < 10; i++) {
            map[row][i] = 0;
        }
    }
}

```

-No primitive
-No else
-Max 2 arguments per method

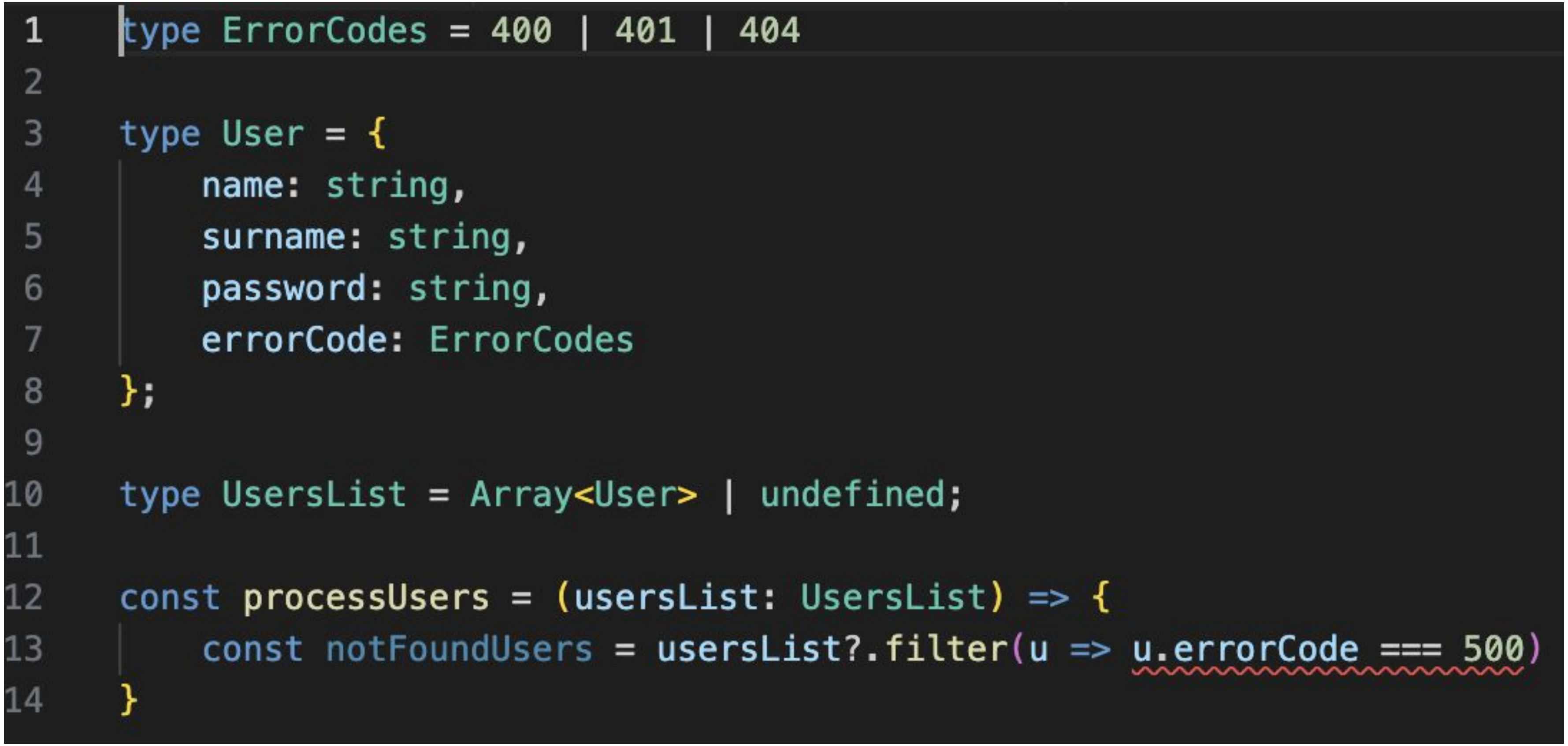


```
class Authentication {

    public login(name: string, surname: string, password: string): void {
        if (this.isValid(name, surname, password)) {
            this.redirect("homepage");
        }else{
            this.addFlash("invalid_credentials");
            this.redirect("login")
        }
    }
```



```
type AuthParams = {
    name: string,
    surname: string,
    password: string
}

class Authentication {

    public login(params: AuthParams): void {
        const authStatus = this.isValid({
            name: params.name,
            surname: params.surname,
            password: params.password
        });

        if (authStatus) {
            return this.redirect(Page.Homepage);
        }

        this.addFlash(CustomError.badCredentials);
        return this.redirect(Page.Login)
    }
```

Type system for very very very simple “wrapping” (instead of real classes)



```
type ErrorCodes = 400 | 401 | 404

type User = {
    name: string,
    surname: string,
    password: string,
    errorCode: ErrorCodes
};

type UsersList = Array<User> | undefined;

const processUsers = (usersList: UsersList) => {
    const notFoundUsers = usersList?.filter(u => u.errorCode === 500)
}
```

Conclusions

These rules, if applied by everyone in the company:

- Drastically improve code readability
- **Provide standardization**
- Improve refactor capabilities

Thank you

m.corradi@davinci.care