```
/*
Object-Oriented Programing

Dvd Player
----------------
Height
Weight
Width
Color

Play
Fast Forward
Rewind
Pause

DVD
-------
Movie Length
Image
Size
*/

class Student{
    constructor (firstName,lastName,phoneNumber,grade) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.phoneNumber= phoneNumber;
        this.grade= grade;
    }
    introduce(){
        console.log(`${this.firstName} ${this.lastName} can be reached at ${this.phoneNumber}`);
    }
}

let student = new Student();
let student1 = new Student('Tom', 'Sawyer','32434553434','A');
let student2 = new Student('Sam', 'Smith', '8989057497','B');

student1.introduce();
student2.introduce();

class NotificationSender {
    constructor(status){
        this.status =status;
    }
    sendNotification(notification){
        console.log('Sending;'+ notification);
        }

    findUserWithStatus(status) {
        let users = getUsers(status);
        return users;
    }
}

class PromotionSender extends NotificationSender {
    constructor (status){
        super(status);

    }
    ////////moved to NotificationSender///////////
    // sendNotification(notification){
    //     console.log('Sending;'+ notification);
    //     }

    // findUserWithStatus(status) {
    //     let users = getUsers(status);
    //     return users;
    // }
    calculateDiscount(status){
        if (status ==='GOLD'){
            return .3;
        } else if (status ==='SILVER'){
        return .15;
        }
        return 0;
    }
}
class CollectionSender extends NotificationSender {
    constructor(status){
        super(status);
    }
    /////////Moved to NotificationSender above/////////
    // sendNoticication(notification){
    //     console.log('Sending'+ notification);
    //     }
    // findUserWithStatus(status) {
    //     let user = getUser(status);
    //     return user;
    // }
    caluculateDiscount(status){
        if (status === 'OVERDUE'){
            return 10;
        }else if (status === 'DELIQUENT'){
            return 25;
        }
        return 5;
    }
}

let collectionSender = new CollectionSender ('OVERDUE');
collectionSender.sendNotification ('This is a test collection');
//////////////////////////////////////
try {
list.push ('Hello');
}catch (err){
    console.log(err);
    }

    console.log('goodbye');

/////////////////Menu/////////////////////////////////////////

class Player{
    constructor(name,position){
        this.name =name;
        this.position = position;

    }
    descripe(){
        return `${this.name}, plays ${this.position}.`;
    }
}

class Team {
    constructor(name){
        this.name=name;
        this.player =[];
    }

    addPlay(player){
        if(player instanceof Player){
            this.players.push(player);
        }else {
            throw new Error(`You can only addan instance of Player. Argument is not a player: 
${player}`);
        }
    }
    describe(){
        return `${this.name} has ${this.players.length} players`;
    }
}

class Menu {
    constructor(){
        this.teams =[];
        this.selectedTeam = null;
    }
    start(){
        let selection = this.showMainMenuOptions();
        while (selection != 0){
            switch (selection){
                case '1':
                this.createTeam();
                break;
                case '2':
                    this.viewTeam();
                break;
                case '3':
                    this.deleteTeam();
                break;
                case '4':
                    this.displayTeams();
                break;
                default:
                    selection = 0;

            }
        selection= this.showMenuOptions();
        }

        alert('Goodbye');
    }
showMainMenuOptions(){
    return prompt(`
    0) exit
    1) create new team
    2) view team
    3) delete team
    4) display all teams
    `);
}
showTeamMenuOptions(teamInfo){
    return prompt(`
    0) back
    1) create player
    2) delete player
    ---------------------
    ${teamInfo}
    `)
}
displayTeams(){
    let teamString ='';
    for(let i =0; i< this.teams.length;i++){
        treamString += i+ ')' + this.teams[i].name + '/n';
    }
    SubtleCrypto(teamString);
    }

createTeam(){
    let name = prompt ('Entername for new team:');
    this.team.push(new Team(name));
    }
viewTeam(){
    let index = prompt('Enter the index of the team you wish to view:');
    if(index >-1 && index < this.teams.length){
        this.selectedTeam = this.teams[index];
        let description = 'Team Name: '+ this.selectedTeam.name + '\n';

        for (let i=0;i< this.selectedTeam.players.length; i++);
        description += i + ')'+ this.selectedTeam.players[i].name + '-' +
this.selectedTeam.players[i].position + '\n';
    }
    let selection = this.showTeamMenuOptions(description)
    switch (selection){
        case '1':
            this.createPlayer();
            break;
        case '2':
            this.deletePlayer();
        }
    }
    deleteTeam(){
        let index = prompt('Enter the index of the team you wish to delete');
        if (index > -1 && index < this.Teams.name.length){
            this.teams.splice(index,1);

        }
    }
    createPlayer(){
        let name = prompt('Enter name for new player');
        let position = prompt ('Enter position for new player:');
        this.selectedTeam.player.push(new Player(name, position));
    }
    deletePlayer(){
        let index = prompt('Enter the index of the player your wish to delete:');
        if (index > -1 && index < this.selectedTeam.players.length) {
            this.selectedTeam.players.splice(index,1);
        }
    }
}
let menu = new Menu();
menu.start();
```