EOOP Final Project – Maciej Zajkowski

This program is for owners of Fitness Club and employees responsible for administration and registration of new customers. Program provides:

- Ability to create database of customers, employees, and Sport Equipment
- Tools for managing club Budged
- Tools for managing club personnel to customers ratio(1 employee – 5 customers) , and other vital for club parameters
- May provide optimization solutions for your Club.
- Helps to organise usage of sport equipment within a club
- Provides shift working mechanism, you are sure that no of your employees will work more than 8 hours.
- Provides 2 carnet types (in dependence of witch hour customer want to work out 8 – 16 or 16 - 22).
- Provides data witch equipment was used by witch customer.
- Helps organising human resources .

Assumptions and restrictions:

1. You can't give ID = 0 to Equipment;
2. You can't offer services to customer if you don't have type of equipment, that your potential customer wants to train on.
3. Your Employees cannot work more than 8 hours per day (in two shift at the same time)
4. Customer can buy 2 carnet types.
5. You can't hire a employee if you can't afford him.
6. You cannot buy Equipment if you can't afford it.
7. You can not input new Customer with already taken ID
8. You can't hire new Employee with already taken ID
9. You can't swap employees that doesn't exist.
10. You can't sell membership to someone who can't afford it
11. You can't chouse carnet type that doesn't exist
12. You can't sell Equipment that doesn't exist
13. You can't sell Equipment that is occupied
14. You can't input Price/wage/Customer Budget with negative number

There are 8 classes, main one is **FitnessClub** it points to two shift classes and to head of list of Equipment(SportEquipmentP). Moreover it contains data: number of customers, employees and Budget. **Every function that user is able to use is in public in that class.**

Class **Shift** points to head of list of Customers (CustomerP), and to the head of list of Employees(EmployeeP) working on that shift. Class shift contains data like price of carnet or number of shift(which will be used only for allocating customer to right occupancy).

Customer class is used to store Customers data like budged, name, preferable type of equipment and unique ID

CustomerP – This class is responsible for handling pointers of customer (it points to class Customer, to Sport Equipment that Customer is using and to next and previous place in the customers List).

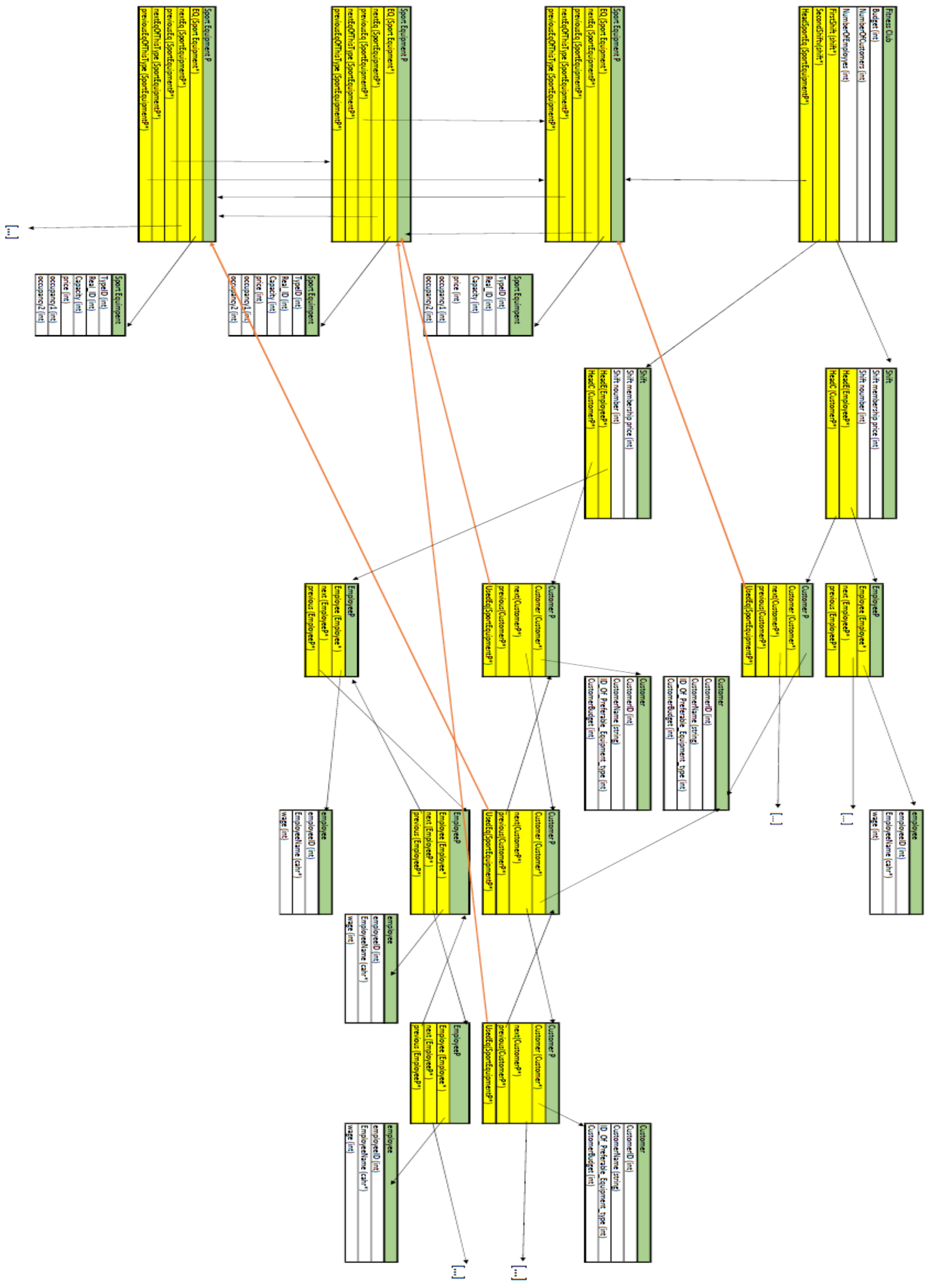
Employee – this class is used to store Employee Data – name, wage and ID

EmployeeP – This class is responsible of handling pointers to next and previous place of Employee List and for pointing to Employee data (Employee class)

SportEquipment class is class used to store data like capacity, occupancy during each shift, price of equipment, type of equipment and it's unique ID.

SportEquipmentP - Points to Sport Equipment Data (class SportEquipment), next and previous place of list. Moreover it points to next and previous Equipment of the same type.

Memory map: (there is provided another pdf with better quality)



Classes

```
 class Employee{
    int employeeID;
    string EmployeeName;
    int wage;
public:
    Employee(int ID, string n, int wage); // constructor of employee
    int& RefWage(); // returns reference of Wage
    int& RefEmployeeID(); // Returns Reference of ID
    bool PutEmployeeName(string n);// puts Name of employee
    string GetEmployeeName();// returns name of employee
};
class EmployeeP{
    Employee* employee; // pointer to stored employees data
    EmployeeP* next; // pointer to next element of list
    EmployeeP* previous;// pointer to previous element of list
public:
    void changeEmpl(Employee* Employee); // changes pointer to employees data
    void PrintEmployee();// prints employee
    Employee* getEmployee(); // returns pointer to stored employees data
    EmployeeP (Employee *e); // constructor of employee
    EmployeeP* GetNextEmployee();// returns reference to next element of list
    EmployeeP* GetPrevEmployee(); // returns reference to previous element of the list
    void ChangePrev_EP(EmployeeP* E);
    void ChangeNext_EP(EmployeeP* E);
} ;
```

```
 class SportEquipment{
    int TypeID;// it is kind of equipment that is strongly correlated with needs of customer
    // customer want to train using treadmill as a type of equipment but not using this specific treadmill
    int Real_ID; // real and unique ID for specific Equipment
    int Capacity;// number of  how much customers this equipment can handle without need to buy one more
    int price; // price of equipment
    int occupancy1; // occupancy of equipment during 1 shift
    int occupancy2; // occupancy of equipment during 2 shift
public:
    SportEquipment(int TypeID, int Real_ID, int Capacity, int price);
    int getTypeID(); // returns TypeID
    int getRealEqID(); // returns reference to real ID
    int GetEqCapacity(); // returns capacity
    int GetPrice(); // returns price
    int& RefOccupancy(int num); // returns occupancy of SportEquipment due to passed number (occupancy 1 for 1 and occupancy 2 for 2)

 };
class SportEquipmentP{
    SportEquipment* Equipment;
    SportEquipmentP* nextEq; // pointer to next Sport Equipment in the list
    SportEquipmentP* previousEq; // pointer to previous Sport Equipment in the list
    SportEquipmentP* nextEqOfThisType; // pointer to previous Sport Equipment of the same type
    SportEquipmentP* previousEqOfThisType; // pointer to next Sport Equipment of the same type
    public:
    void PrintEquipment();
    SportEquipment* getEq();
    void ChangePrev_EQ(SportEquipmentP* E); // changes pointer of previous to given pointer
    void ChangeNext_EQ(SportEquipmentP* E); // changes next of previous to given pointer
    void ChangePrev_Type_EQ(SportEquipmentP* E); // changes pointer of previous type to given pointer
    void ChangeNext_Type_EQ(SportEquipmentP* E); // changes pointer of next type to given pointer
    SportEquipmentP(SportEquipment* Eq); // constructor of class SportEquipment
    SportEquipmentP* GetNextEq(); // returns reference to next element of list
    SportEquipmentP* GetPrevEq(); // returns reference to previous Sport Equipment in the list
    SportEquipmentP* GetNextEqOfType(); // returns reference to previous Sport Equipment of the same type
    SportEquipmentP* GetPrevEqOfType(); // returns reference to next Sport Equipment of the same type
 };
```

```cpp
class Customer{
    int CustomerID;
    string CustomerName;
    int ID_Of_Preferable_Equipment_type; // ID of Equipment type  that customer will train on
    int CustomerBudget;
public:
    Customer(int CustomerID, string Name, int IDEq, int CustomerBudget); // constructor of customer
    string GetCustomerName(); // returns customer name
    int& RefCustomerID(); // returns reference to customer ID
    int& RefCustomerBudget(); // returns reference Customers Budget
    int GetCustomerEq(); // returns customers ID_Of_Preferable_Equipment
} ;
class CustomerP{
    Customer* customer;
    CustomerP* next; // pointer to next element of list
    CustomerP* previous; // pointer to previous element of list
    SportEquipmentP* UsedEquipment;// pointer to equipment that is used by this customer
public:
    void ChangeCustomer(Customer* c); //changes customer pointer  to given pointer
    void PrintCustomer();
    void ChangeUsedEq(SportEquipmentP* E); //changes Used Equipment pointer  to given pointer
    CustomerP(Customer * C);
    Customer* getCustomer(); // returns
    CustomerP* GetNextCustomerP(); // returns reference to next element of list
    CustomerP* GetPrevCustomerP(); // returns reference to previous element of the list
    void ChangePrev_CP(CustomerP* C);// changes pointer of previous to given pointer
    void ChangeNext_CP(CustomerP* C);// changes pointer of next to given pointer
    SportEquipmentP* GetUsedEq(); // returns reference equipment that is used by this customer
};


class Shift{
    int CarnetPrice; // price of carnet
    int ShiftNumber; // type of shift ("1" for 8-16 or "2" for 16-22)
    EmployeeP* headE; // head of list of employees
    CustomerP* headC; // head of list of customers
public:
     int HowManyEmployees(); // returns number of Employees in shift
     int howMannyCustomers(); // returns number of Customers in shift
    CustomerP* GetHeadC(); //
    EmployeeP* GetHeadE(); //
    CustomerP* FindCustomer(int CustomerID); // finds and returns reference to CustomerP
    EmployeeP* FindEmployee(int ID); // finds and returns reference to EmployeeP
    bool deleteptr(); // deletes all pointers of used Eq in whole list
    EmployeeP* GetLastE(); // returns last Employee in list
    bool CheckRatio(); // checks if ratio of customers / employees will be saved if we add new customer
    bool Check_Changed_Ratio();// checks if ratio of customers / employees will be saved if we remove Employee
    int Sum_Wages(); // sum wages of all shift
    bool isCustomer(int ID); // checks if there is such Customer -returns true if there is
    bool isEmployee(int ID); // checks if there is such Employee -returns true if there is
    Shift(int n, int price); // constructor of shift
    bool Hire_Employee(Employee* e); // adds employee to list
    bool Fire_Employee(int ID);// removes Employee from list
    bool PutCustomer(Customer* c);// adds Customer to list
    bool End_Membership(int CustomerID); // removes customer from list
    int& RefCarnetPrice(); // returns reference to Carnet Price
    int GetShiftNum();
    ~Shift();
};



class FitnessClub{
    int NumberOfCustomers;
    int NumberOfEmployyes;
    int Budget;
    Shift* FirstShift; // pinter to first shift
    Shift* SecondShift; // pointer to Second Shift
    SportEquipmentP* headEq; // pointer to head of Equipment list
    bool IsNotOccupiedEq(int Type, int CarnetType);
    bool setpointers(); // sets pointers of all customers to the Equipment they will use
    bool IsEquipment(int RealID);// returns true if there exist eq with this real ID
    SportEquipmentP* FindEq(int Real_ID); // returns reference to Eq with this real ID
    int SumWages(); // sums wages of all employees
    bool deletePointers();// deletes pointers of all customers to the Equipment they were using
    bool IS_TYPE_EQ(int TYPE_ID,int Excluded_ID);// checks if there is this type of ID in list doesn't count Equipment witch Real ID is excluded
    // if there is no Excluded ID needed type it as 0 - this function is the reason of why 0 can not be Equipments real ID
    SportEquipmentP* Find_EQ_Type(int Type);// returns pointer to first equipment of this type in the list
    bool IS_EMPLOYEE(int ID); // checks if there is employee in with this ID in a club returns true if there is
    Shift* GetShift(int num);
    // returns pointer to Shift due to passed number (occupancy 1 for 1 and occupancy 2 for 2)
    int diffrent_shift(int x);
    Employee* Find_Employee(int ID); // returns pointer to employee with this ID if there is no employee with this ID returns nullptr
    Customer* Find_Customer(int ID);// analogic as in Find_Employee(int ID) but with customer
    int Find_E_Shift(int ID); // returns number of shift in witch is employee if there is no such employee returns 0
    int Find_C_Shift(int ID); // same as Find_E_Shift(int ID) but with customer
    bool ChangeShiftR(int Shift_num); // changes shift for last employee in list
    int& RefNumOfEmployees(); // returns reference to number of employees
    int& RefNumOfCustomers(); // returns reference number of customers
public:
    bool PrintCustomers();// prints all customers and  their data
    bool PrintEmployees();// prints all Employees and  their data
    bool PrintEquipment();// prints all of Equipment data
    bool PrintClub(); // prints all of clubs data (uses  PrintCustomers(),PrintEmployees(), PrintEquipment())
    FitnessClub(int Budget, int price1, int price2); // constructor of fitness club enter budget and carnet prices
    bool HireEmployee(int ShiftNoumber, int ID, string n, int wage);// checks restrictions and assumptions ,hires employee to given class, does not work if ex
    // inceases NumberOfEmployyes by one
    bool FireEmployee(int employeeID); // fires Employee
    bool SellMembership(int CarnetType, int CustomerID, string Name, int ID_Of_Preferable_Equipment_type, int CustomerBudget);
    // sells carnet to Customer, does not work if exist customer with same ID increases NumberOfCustomers by one
    // if there is not enough employees in wanted shift and if there is employee with nothing do do in diffrend shift
    // his shift is automatically changed
    bool EndContract(int CustomerID); // ends contract with customer (removes him from both shifts)
    bool SellEquipment(int Real_ID); // sells Equipment, you cant sell eq that is occupied or that doesn't exist
    bool BuyEquipment(int TypeID, int Real_ID, int Capacity, int price); // if there isn't Eq with the same Real ID number
    //  creates new Equipment and adds int to the list
    bool CheckWorkingConditions(); // checks if there are no employees working on 2 shifts
    bool UsersOfEq(int Real_ID); // prints users of specify equipment
    bool DiagnoseClub();// write what should we do to make club more profitable
    bool swapEmployees(int Employee1ID, int Employee2ID); // swaps employees between shifts and between places in list
    bool ChangeShift(int EmployeeID);// last employee in the list changes shift if there is enough workers to work in current one
    bool ChangeWage(int EmployeeID, int newwage); // changes wage of employee;
    bool ExpandContract(int CustomerID); // expands contract to another carnet for customer (works if there is place, stuff and hight enough customers budget)
    int getNumberOfCustomers(); // returns number of customers
    int getNumberOfEmployes(); // returns number of Employees
    int& RefClubBudget(); // returns reference to Club budget
    int  BudgetBalance();// returns clubs incomes - expenses
    int HowManyEq(); // returns number of used Eq;
    ~FitnessClub();// destructor, leaves every pointer to nullptr and every intiger to 0, uses ~Shift();
};
```

Testing

```
bool Test_1();// adding employee/customer adds new place in appropriate list
bool Test_2();// Budget Balance returns right value
bool Test_3();// diagnose club works properly
bool Test_4();//you can not sel membership if you have no right eq
bool Test_5();//you cant fire employee that is working
bool Test_6();//you cant sell eq that is occupied
bool Test_7();//you can not fire employee that doesn't exist
bool Test_8();//you cannot add customer with ID that  already exist in the shift
bool Test_9();//if you remove every element from club it is empty
bool Test_10(); // you can't sell membership before you hire employees or buy equipment
bool Test_11();// Checks if there is Change shift works properly
bool Test_12(); // you cannot buy eq over your budget
void Test_13();// you can print empty club
void Test_14(); // you can print empty Equipment list
void Test_15(); // you can print empty Customer list
void Test_16(); // you can print empty Employee list
void Test_17(); // printing after removing
void Test_18(); // printing after using deconstructor
bool Restriction_1();
bool Restriction_2();
bool Restriction_3();
bool Restriction_4();
bool Restriction_5();
bool Restriction_6();
bool Restriction_7();
bool Restriction_8();
bool Restriction_9();
bool Restriction_10();
bool Restriction_11();
bool Restriction_12();
bool Restriction_13();
bool Restriction_14();
void Testing();
```