Lista 3: Nivelamento – Introdução a Programação por Objetos e Arquivo Texto

Observações:

- Cópias serão desconsideradas, ou seja, a nota será igual a 0 (zero).
- Implemente os programas utilizando a linguagem C#.
- Na resolução dos exercícios só podem ser utilizados comandos vistos nas aulas.

1) Considere a classe `Estacionamento`, cuja assinatura está definida abaixo:

```
using System;

class Estacionamento {

        private String nome;  // nome do estacionamento.

        private int numTotalVagas;  // número total de vagas que o
estacionamento apresenta.

        private int numVagasLivres;  // número de vagas livres no
estacionamento.

        private String[] vaga;  // vetor que armazena cada uma das vagas do
estacionamento. Caso a vaga esteja ocupada, este vetor indica a placa do
veículo que a ocupa.


        public Estacionamento(String nome, int numTotalVagas);
        public int estacionar(String placa);
        public int obterVagaOcupada(String placa);
        public void retirarVeiculo(String placa);
        public int obterNumVagasLivres();
        public void exibirOcupacaoEstacionamento();
}
```

a) Implemente todos os métodos da classe `Estacionamento` acima, obedecendo às seguintes descrições:

- `Estacionamento(String nome, int numTotalVagas)`: Construtor que inicializa os atributos `nome` e `numTotalVagas` com os valores passados por meio dos parâmetros desse método; o atributo `numVagasLivres` com o número total de vagas do estacionamento; e os elementos do vetor `vaga` com `null`. Não se esqueça de instanciar o vetor `vaga`;

- `estacionar(String placa)`: Se existirem vagas disponíveis no estacionamento, encontra a primeira posição vazia do vetor `vaga` e insere a placa informada como parâmetro para esse método nessa posição. Se não existirem vagas disponíveis, esse método deve retornar -1;

- `obterVagaOcupada(String placa)`: Retorna o número da vaga, ou seja, a posição do vetor `vaga`, ocupada pelo veículo com a placa informada como parâmetro para esse método. Se a placa informada não for encontrada, deve retornar -1;

- `retirarVeiculo(String placa)`: Retira o veículo cuja placa corresponde à informada como parâmetro para esse método, marcando a vaga correspondente com `null`;

- `obterNumVagasLivres()`: Retorna o número de vagas disponíveis no estacionamento;

- `exibirOcupacaoEstacionamento()`: Imprime o número de cada vaga do estacionamento e a placa do veículo que a ocupa ou a informação de que a vaga está vazia.

b) Após a implementação da classe `Estacionamento`, o método `Main`, a seguir, deverá funcionar:

```
using System;

class Program {

        static void Main(string[] args) {

            int vagaOcupada;
            Estacionamento objEstacionamento = new
Estacionamento("Estacionamento de AEDs", 10);

            vagaOcupada = objEstacionamento.estacionar("HKT0098");
            vagaOcupada = objEstacionamento.estacionar("OLP4290");
            vagaOcupada = objEstacionamento.estacionar("HJB0495");
            vagaOcupada = objEstacionamento.estacionar("OWB3904");

            Console.WriteLine("Ocupação do estacionamento após as chegadas de
quatro clientes:");
            objEstacionamento.exibirOcupacaoEstacionamento(); // HKT0098
OLP4290 HJB0495 OWB3904 vazia vazia vazia vazia vazia vazia

            vagaOcupada = objEstacionamento.obterVagaOcupada("HKT0098");
            Console.WriteLine("Veículo de placa HKT0098 estacionado na vaga " +
vagaOcupada);  // 0

            objEstacionamento.retirarVeiculo("HKT0098");
            Console.WriteLine("Ocupação do estacionamento após a retirada do
veículo de placa HKT0098:");
            objEstacionamento.exibirOcupacaoEstacionamento(); //  vazia OLP4290
HJB0495 OWB3904 vazia vazia vazia vazia vazia vazia

            vagaOcupada = objEstacionamento.estacionar("HTP5619");
            vagaOcupada = objEstacionamento.estacionar("BOL4861");
            vagaOcupada = objEstacionamento.estacionar("HGT9436");
            Console.WriteLine("Ocupação do estacionamento após as chegadas de
mais três clientes:");
            objEstacionamento.exibirOcupacaoEstacionamento(); // HTP5619
OLP4290 HJB0495 OWB3904 BOL4861 HGT9436 vazia vazia vazia vazia

            Console.WriteLine("Este estacionamento apresenta {0} vagas
livres.", objEstacionamento.obterNumVagasLivres()); // 4 vagas livres
            Console.ReadLine();
        }
}
```

2) Implemente uma classe `Pessoa` que tenha como dados data de nascimento, peso e altura. Esse TAD tem como operações válidas: informar a idade atual da pessoa; e calcular seu IMC. Crie também um programa que leia dados de pessoas a partir de um arquivo texto. Nesse arquivo, cada linha contém a data de nascimento, o peso (em kg) e a altura de uma pessoa (em metros), sendo esses dados separados por ":". O programa também deve ter a opção de inserir dados de uma nova pessoa no sistema. Antes do programa ser finalizado, deve atualizar o arquivo de dados com as novas pessoas cadastradas.

Cálculo do IMC: $\frac{peso}{altura^2}$

3) Crie um programa com as seguintes classes:

Classe Jogador:

- Atributos
  - numero
  - nome
  - posicao (opções: goleiro, zagueiro, lateral, meia e atacante).
- Métodos:
  - Método construtor
  - Métodos Getters e Setters

Classe Time

- Atributos
  - nome
  - titulares (vetor de Jogador, tamanho 11)
  - quantTitulares (representa a quantidade atual de jogadores titulares)
  - reservas (vetor de Jogador, tamanho 12)
  - quantReservas (representa a quantidade atual de jogadores reservas)
  -
- Métodos
  - Métodos construtor: deve instanciar os vetores e também inicializar os atributos quantTitulares e quatReservas com 0.
  - GetNome e SetNome
  - AdicionarTitular: recebe como parâmetro um objeto Jogador e deve inseri-lo no vetor titulares. Não deve ser permitido adicionar mais de 11 jogadores titulares. Após adicionar um jogador titular o atributo quantTitulares deve ser incrementado. O método deve retornar um valor booleano para indicar se a adição foi efetuada com sucesso, ou não.
  - AdicionarReserva: recebe como parâmetro um objeto Jogador e deve inseri-lo no vetor reservas. Não deve ser permitido adicionar mais de 12 jogadores reservas.Após adicionar um jogador titular o atributo quantReserva deve ser incrementado. O método deve retornar um valor booleano para indicar se a adição foi efetuada com sucesso, ou não.
  - SubstituirTitular: recebe como parâmetro o nome do jogador titular que deve ser substituído e o novo jogador, isto é, um objeto Jogador. O método deve retornar um valor booleano para indicar se a substituição foi efetuada com sucesso, ou não.
  - SubstituirReserva: recebe como parâmetro o nome do jogador reserva que deve ser substituído e o novo jogador, isto é, um objeto Jogador. O método deve retornar um valor booleano para indicar se a substituição foi efetuada com sucesso, ou não.
  - ConsultarTitular: recebe como parâmetro o nome de um jogador titular e deve retornar true, caso o jogador esteja no time. Caso contrário deve retornar false
  - ConsultarReserva: recebe como parâmetro o nome de um jogador reserva e deve retornar true, caso o jogador esteja no time. Caso contrário deve retornar false.
  - ExcluirTitular: recebe como parâmetro o nome do jogador titular que deve ser excluído do time. Após a exclusão não deve ser deixado nenhum espaço vazio no vetor, assim todos os jogadores que estiverem nas posições posteriores a posição da exclusão, devem ser deslocados uma posição para frente. Além disso, o atributo quantTitulares deve ser decrementado.
  - ExcluirReserva: recebe como parâmetro o nome do jogador reserva que deve ser excluído do time. Após a exclusão não deve ser deixado nenhum espaço vazio no vetor, assim todos os jogadores que estiverem nas posições posteriores a posição da exclusão, devem ser deslocados uma posição para frente. Além disso, o atributo quantReservas deve ser decrementado.
  - GerarArqTime: o método deve receber como parâmetro o nome de um arquivo texto. No método deve ser criado um arquivo com o nome passado como parâmetro. Além disso, deve ser escrito neste arquivo os dados de todos os jogadores (titulares e reservas) do time.

Classe Teste

- No método Main, instancie um time e utilize todos os métodos da classe Time.

4) Implemente em C# um sistema para controle de sua biblioteca pessoal. O sistema é formado pelas classes “Livro” e “Biblioteca” com as características descritas abaixo:

Classe: **Livro**

- Atributos: titulo (título do livro, tipo String), autores (autores do livro, tipo String) e editora (editora que publicou o livro, tipo String). Todos atributos privados;
- Método construtor para inicializar os atributos;
- Métodos G*etters* para obter cada um dos atributos;
- Métodos *Setters* para alterar cada um dos atributos.

Classe: **Biblioteca**

- Atributos: livros (lista de livros da biblioteca, tipo array), numLivros (número de livros inseridos na lista, tipo int) e MAXLIV (número máximo de livros na lista, constante do tipo int com valor 50);
- Método para adicionar um livro na lista. Parâmetros: titulo, autores e editora;
- Método para adicionar um livro na lista. Parâmetro: objeto do tipo Livro;
- Método para retornar o livro cujo título é recebido como parâmetro. Retorna null se o livro não existir. Parâmetro: título do livro;
- Método para retornar a lista de livros;
- Método para retornar o número de livros da biblioteca.

Classe: Teste

- Método Main para executar as seguintes ações, usando todos os métodos definidos nas classes:
  - Criar um objeto do tipo Biblioteca;
  - Adicionar quatro livros à biblioteca;
  - Imprimir os dados de um livro da biblioteca dado seu título, ou uma mensagem de erro se o livro não existir;
  - Imprimir a relação de todos os livros da biblioteca;