ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia de Computação e Sistemas Digitais

PCS 3111 - Laboratório de Programação Orientada a Objetos para Engenharia Elétrica
2020

Aula 08 – Classes Abstratas, Herança Múltipla e métodos e atributos estáticos

Atenção

1. Código inicial a ser usado na resolução dos exercícios encontra-se **disponível no e-Disciplinas**.
2. Os nomes, os atributos, os métodos, e as respectivas assinaturas das classes dadas **devem seguir o especificado** em cada exercício para fins de correção automática.

AVISO: para evitar problemas de compilação no Judge, envie os exercícios à medida que implementá-los!

Exercício 1
Implemente a classe **Participante** de forma que ela seja abstrata.

```
Participante(string nome);
virtual ~Participante();

virtual void imprimir();
virtual string getNome();
```

A classe **Participante** possui apenas um atributo nome e um método `void imprimir` (que não será avaliado – use-o para testes). Escolha o método apropriado para ser abstrato.

Crie e implemente também uma classe concreta **Pessoa**, filha de **Participante**, que recebe em seu construtor os parâmetros `nome` e `NUSP`. Ela possui os seguintes métodos específicos:

```
Pessoa(string nome, int nusp);
virtual ~Pessoa();

virtual int getNusp();
```

A classe **Equipe** fornecida trabalha com um vetor de `int` que armazena os `NUSP` dos integrantes. Modifique-a para que ela seja concreta e filha de **Participante** (faça as adaptações necessárias) e utilize um vetor de **Pessoas**, de forma que sua definição fique com os seguintes métodos específicos:

```
Equipe(string nome, int maxValor);
virtual ~Equipe();

virtual bool adicionar(Pessoa* p);
virtual Pessoa** getPessoas();
```

Mantenha o mesmo comportamento do método `adicionar` (retorne falso caso não haja espaço no vetor ou a pessoa já tenha sido adicionada anteriormente).

Exercício 2

a) Implemente a classe **CentroAcademico** (cuja definição é entregue – adicione-a ao projeto), que possui um `presidente` e `sigla`. A classe possui a seguinte declaração:

```
CentroAcademico(Pessoa* presidente, string sigla);
virtual ~CentroAcademico();

virtual string getSigla();
virtual Pessoa* getPresidente();
```

b) Crie também a classe **EquipeDoCA** (equipe do centro acadêmico) que deve ser derivada de **Equipe** e também ser de **CentroAcademico**. A classe **EquipeDoCA** possui apenas o construtor (e o destrutor), que deverá ser igual ao seguinte modelo:

```
EquipeDoCA(string nome, string sigla, Pessoa* presidente, int
maxValor);
```

Exercício 3

Considere agora que a classe **Pessoa** possua uma sobrecarga de seu construtor, que não receba o parâmetro `nusp`.

Adicione e implemente o novo construtor e um método estático getNuspAuto, deixando os já existentes. Ou seja, a classe deve possuir os seguintes métodos públicos:

```
Pessoa(string nome);
Pessoa(string nome, int nusp);
virtual ~Pessoa();

virtual int getNusp();
static int getNuspAuto();
```

Ao se chamar esse construtor, a **Pessoa** deve receber um nusp gerado automaticamente, ou seja, o método `getNusp` deve retornar o nusp gerado para aquela pessoa.

A cada Pessoa criada o nusp gerado automaticamente deve ser incrementado em 1. Utilize o método **getNuspAuto** para obtê-lo. O nusp gerado automaticamente deve ter seu valor inicial igual a **12345678** (crie um atributo para isso).

Ou seja, a primeira pessoa criada com esse novo construtor terá o seu nusp = 12345678, a próxima pessoa terá o nusp = 12345679.

Testes do Judge

Exercício 1

- Participante: é classe abstrata
- Equipe: é classe derivada de Participante
- Equipe: vetor de Pessoa
- Pessoa: getters

Exercício 2

- CentroAcademico: getters
- EquipeDoCA: é classe derivada de Equipe e de CentroAcademico

Exercício 3

- Pessoa: getNuspAuto é método estático
- Pessoa: primeira pessoa com valor inicial
- Pessoa: nuspAuto é atributo estático
- Pessoa: getters pessoas com nuspAuto