Universidade do Estado de Santa Catarina
Ciência da Computação
Programação Orientada a Objetos
Exame – 2018 / 1

Nome: _____ **Nota:** _____

Atenção: neste exame você poderá utilizar um computador sem acesso a internet para produzir os seus códigos. Nenhum tipo de consulta é permitida durante o exame.

Questão 1 **(4.0)** O paradigma de programação orientado a objetos introduziu uma série de conceitos. Neste sentido, responda o que se pede a seguir:

a) Defina classe e objeto apresentando um exemplo de cada;

b) Defina encapsulamento apresentando e diferenciando as formas de encapsulamento que foram vistas durante o semestre – isto é, as formas que utilizamos na linguagem de programação Java. Diga ainda quais elementos básicos podem ser encapsulados.

c) Diga o que significa o conceito de herança. Apresente um exemplo e o explique.

d) Defina o que são classes e métodos abstratos. Quais são as implicações sobre uma classe que é abstrata? E sobre um método abstrato?

e) Diga o que significa o conceito de interface. Apresente um exemplo e o explique.

f) Diga o que significa o conceito de sobrecarga (*overloading*) de métodos, e de operadores. Apresente um exemplo e o explique.

g) Diga o que significa o conceito de sobrescrita (*override*) de métodos. Apresente um exemplo e o explique.

h) Defina polimorfismo vinculando-o com os conceitos de *upcasting* e *downcasting*.

Questão 2 **(6.0)** Elabore um sistema de gestão acadêmico simples utilizando a linguagem de programação Java conforme os requisitos apresentados:

- **(0.15)** Crie um enumerador público denominado Nivel que enumera dois níveis de escolaridade: graduação e mestrado;

- **(0.25)** Crie uma classe denominada Curso que possui dois atributos: nome (String) e nível (enumerador Nivel). Crie ainda métodos *getters* e faça com que um curso só possa ser instanciado se forem fornecidos o nome e o nível do curso.

- **(0.25)** Crie uma classe denominada Disciplina que possui os atributos: nome (String), sigla (array de char) e curso (Curso). Crie ainda métodos *getters*, sendo que o método `getSigla` deverá retornar uma String e não um *array* de caracteres;

- **(0.25)** Ainda para a classe Disciplina, crie uma constante que define o tamanho correto das siglas que é sete caracteres. Faça também com que uma nova disciplina só possa ser instanciada se forem fornecidos todos os dados para os seus atributos. Caso a sigla informada não tenha sete caracteres, gere uma exceção do tipo `IllegalArgumentException`;

- **(0.15)** Crie uma classe abstrata Aluno com os atributos: nome (String), matricula (int), semestre ingresso (int), ano ingresso (int), curso (Curso) e score (int);

- **(0.15)** Para a classe Aluno, garanta que um aluno só possa ser instanciado se forem fornecidos todos os seus dados com exceção de score que deve sempre iniciar com o valor zero;

- **(0.3)** Ainda para a classe Aluno, crie métodos *setters* para os atributos nome e curso, um método incrementa score que recebe um inteiro e incrementa o score com base neste valor, e também métodos *getters* para nome, matricula, curso e score. Por fim, crie um *getter* diferenciado para semestre de ingresso, o qual retorna uma String que é a concatenação do atributo ano de ingresso, com uma barra '/' e o semestre de ingresso;

- **(0.35)** Crie uma classe denominada Nota que se utiliza de um tipo genérico T, o qual representa o tipo de dado utilizado como conceito / nota. Esta classe deve ter três atributos: a disciplina referente (Disciplina), o ano-semestre de referência (String) e o valor da nota (T);

- **(0.2)** Ainda para a classe Nota, crie um construtor que recebe uma disciplina, o ano-semestre de referência e uma nota, e atribui estes valores aos atributos da classe. Crie ainda os métodos *getters* e apenas um *setter* para alterar o valor da nota;

Universidade do Estado de Santa Catarina
Ciência da Computação
Programação Orientada a Objetos
Exame – 2018 / 1

- **(0.3)** Crie uma classe denominada Graduando que herda da classe Aluno e possui como atributo um *array* de notas (Nota[]) que utiliza um padrão inteiro para as notas – por exemplo, 98 representa 9,8. Defina ainda uma constante (int) para o número máximo de notas que um aluno graduando pode ter. Atribua este valor como 100.

- **(0.25)** Ainda para a classe Graduando, garanta que um graduando só possa ser instanciado com todas as suas informações básicas de Aluno, e que seu vetor de notas seja instanciado com o tamanho máximo padrão definido pela constante da classe;

- **(0.3)** Crie uma classe denominada Mestrando que herda da classe Aluno e possui como atributo o nome do seu orientador (String) e um vetor de notas (Nota) que utilizam um padrão de caracter para as notas – por exemplo, 'A' representa uma nota entre 9,0 e 10,0, 'B' representa uma nota entre 7,0 e 8,9. Defina ainda uma constante para o número máximo de notas que um aluno mestrando pode ter. Atribua este valor como 25.

- **(0.25)** Ainda para a classe Mestrando, garanta que um mestrando só possa ser instanciado com todas as suas informações básicas de Aluno, com o nome de seu orientador, e que seu vetor de notas seja instanciado com o tamanho máximo padrão definido na constante da classe;

- **(0.35)** Na classe Aluno, adicione um método abstrato `getNota` que recebe como parâmetro uma disciplina e deve recuperar e retornar o registro da nota do aluno naquela disciplina. Caso não houver um registro de nota para a disciplina informada, retornar *null*. Retorne a primeira nota encontrada implementando o método nas classes Graduando e Mestrando;

- **(0.25)** Crie uma classe denominada Turma que possui três atributos: qual a disciplina (Disciplina), qual o semestre (String) e um *array* de Alunos. Esta classe deve ter também a definição de duas constantes de classe que se referem ao tamanho máximo de uma turma de graduação (40) e de uma turma de mestrado (20);

- **(0.4)** Ainda para a classe Turma, crie métodos *getters* para disciplina e semestre, e um *getter* especial para Aluno, o qual recebe um valor inteiro indicando o índice do aluno a ser consultado. Faça com que uma Turma só possa ser instanciada se for fornecida a disciplina e o semestre, sendo que o *array* de Alunos deve ser inicializado com o tamanho máximo apropriado. Isto é, se a turma for referente a uma disciplina de graduação, o tamanho do *array* deve ser 40, caso seja de mestrado, o tamanho do *array* deve ser 20;

- **(0.5)** Adicione ainda na classe Turma, um método denominado matricular, que recebe como parâmetro o aluno que está se tentando matricular. O método deve verificar se o aluno está matriculado em um curso de grau compatível com a disciplina da turma que se está tentando matricular, e também verificar se existe vaga na turma (por exemplo: um aluno de gradução não pode se matricular em uma turma referente a uma disciplina de mestrado, e um aluno não pode se matricular caso a turma já esteja lotada). Este método deverá retornar um valor booleano indicando se a matricula foi efetivada ou não.

- **(0.5)** Crie uma classe denominada Universidade utilizando o padrão de projeto Singleton para garantir que apenas uma instância possa ser criada e seja acessível. Esta classe deve ter três coleções de dados (utilize alguma coleção da biblioteca *collections* do Java): cursos, graduandos, mestrandos, disciplinas e turmas;

- **(0.3)** Ainda para a classe Universidade, crie um método polimórfico denominado cadastrarAluno que recebe um Aluno (graduando ou mestrando) e o insere na coleção apropriada;

- **(0.55)** Ainda para a classe Universidade, crie um método (`getHistorico`) que recebe o valor a matrícula de um aluno – que pode ser graduando ou mestrando – e retorne uma coleção de disciplinas nas quais ele foi aprovado. Considere que um graduando é considerado aprovado caso tenha uma nota igual ou superior a sete e um mestrando é considerado aprovado caso tenha uma nota (conceito) igual a 'A' ou 'B'.

```
if( you.haveStudied() ) printf("--- Bom exame! ---");
else printf("--- Boa sorte! ---");
```