Universidade do Estado de Santa Catarina
Ciência da Computação - Programação Orientada a Objetos
Exame – 2018 / 2

Nome: _____ **Nota:** _____

Questão 1 **(3.0)** O paradigma de programação orientado a objetos introduziu uma série de conceitos. Neste sentido, responda o que se pede a seguir:

a) (0.4) O que é o conceito de encapsulamento? Apresente e diferencie as formas de encapsulamento que foram vistas.

b) (0.6) Diga o que significa o conceito de interface. Apresente um exemplo prático e o explique.

c) (0.8) Diga o que significa o conceito de sobrecarga (*overloading*) de métodos, e de operadores. Apresente um exemplo prático de sobrecarga de método e de sobrecarga de operador em Java ou C++ e explique-os.

d) (0.4) Diga o que significa o conceito de sobrescrita (*override*) de métodos. Apresente um exemplo e o explique.

e) (0.8) Defina polimorfismo vinculando-o com os conceitos de *upcasting* e *downcasting.* Apresente um exemplo prático destes conceitos.

Questão 2 **(7.0)** Elabore um sistema de gestão acadêmico simples utilizando a linguagem de programação Java conforme os requisitos apresentados:

- **(0.4)** Crie uma classe denominada Curso que possui dois atributos: `nome` (texto) e `nível` (pode assumir apenas dois valores: graduação ou mestrado). Garanta que um novo curso só possa ser instanciado se forem fornecidos estes valores e que, uma vez instanciado, não seja possível modifica-los, apenas consultá-los;

- **(0.4)** Crie uma classe denominada Disciplina que possui os atributos: `nome` (nome da disciplina), `sigla` (um vetor de caracteres com tamanho fixo de sete caracteres) e `curso` (um objeto que representa o curso no qual a disciplina está integralizada na grade). Crie uma constante – acessível no pacote e não somente na classe – que define/especifica o tamanho correto das siglas que é de sete caracteres;

- **(0.3)** Faça com que uma nova disciplina só possa ser instanciada se forem fornecidos todos os dados para os seus atributos (sendo sigla um vetor de caracteres) e garanta que os atributos desta classe possam ser acessados e modificados externamente apenas através de métodos *getters & setters*, sendo que o método `getSigla()` deverá retornar um objeto `String` e o método `setSigla()` deverá gerar um clone do vetor passado por parâmetro à função, isto é, utilize o método clone() próprio dos vetores (é um método de um vetor);

- **(0.3)** Ainda para a classe Disciplina, caso a sigla informada para o método construtor não tenha sete caracteres, gere e arremesse uma exceção do tipo `IllegalArgumentException` (trata-se de uma exceção do próprio Java, do pacote pré-embutido `java.lang`);

- **(0.3)** Crie uma classe denominada Aluno, a qual não pode ser instanciada, com os atributos: `nome` (texto), `matricula` (inteiro), `semestre ingresso` (inteiro), `ano ingresso` (inteiro), `curso` (objeto Curso) e `score` (ponto flutuante). O nome deve ser modificável em qualquer local do projeto, mas os demais atributos devem ser manipuláveis diretamente apenas na classe;

- **(0.3)** Garanta que um objeto subtipo de aluno só possa ser instanciado se forem fornecidos todos os seus dados com exceção de matrícula e score, sendo que a matrícula deverá ser um valor inteiro positivo único e incrementável por aluno, começando em 1 – isto é, o primeiro aluno terá matrícula 1, o segundo terá matricula 2, e assim sucessivamente – e o score deve iniciar com um valor inicial zero;

- **(0.3)** Ainda para a classe Aluno, crie métodos *getters* para matricula, curso e score, além de um *getter* diferenciado para semestre de ingresso, o qual retorna um objeto `String` que é a concatenação do ano de ingresso e o semestre de ingresso. Crie também um método setter para o atributo curso, um método denominado `incrementaScore` que recebe um ponto flutuante e incrementa o score com base neste valor;

- **(0.3)** Crie uma classe denominada Turma que possui três atributos: qual a disciplina (Disciplina) da turma, qual o semestre (String) de referência que será informado no formato *aaaa-s* (onde *aaaa* representa o ano e *s* o número do semestre) e um *array* de Alunos. Defina também 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)** Na 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 no vetor de alunos. Para este último *getter*, trate a possível exceção de `ArrayIndexOutOfBoundsException` que pode ocorrer ao acessar o vetor. Caso ocorra tal exceção, retorne nulo e imprima uma mensagem / log de erro usando o método `System.err.println`;

- **(0.3)** Faça com que uma Turma só possa ser instanciada recebendo os parâmetros `disciplina` e `semestre`, sendo que o *array* de Alunos deve ser inicializado com o tamanho máximo apropriado – deve ser 40 para turmas de um curso de graduação ou 20 para turmas de um curso de mestrado;

- **(0.5)** Sabe-se que nos cursos de graduação os conceitos, ou notas, são determinados por valores inteiros, mas nos cursos de mestrado, tais valores são definidos por letras. Crie uma classe denominada Nota que se utiliza de um tipo genérico `T` que representa o tipo de dado utilizado como conceito/nota. Esta classe deve ter dois atributos privados: a turma referente a nota (Turma) e o valor da nota (`T`). Crie um construtor que recebe um objeto turma e o valor da nota e os atribui devidamente. Faça com que todos os valores sejam acessados externamente através de *getters*, mas garanta que apenas o valor da nota/conceito possa ser alterado externamente;

- **(0.3)** Crie uma classe denominada Graduando que herda da classe Aluno e possui como atributo uma coleção (utilize qualquer coleção de *Collections*) de notas (Nota) que utiliza um padrão inteiro para as notas – por exemplo, 98 representa 9,8 – isto é, o tipo genérico de nota deve ser utilizado como inteiro. Garanta que um graduando só possa ser instanciado com todas as suas informações básicas de Aluno, e que sua coleção de notas seja instanciada;

- **(0.3)** Crie uma classe denominada Mestrando que herda da classe Aluno e possui como atributo o nome do seu orientador (String) e uma coleção de notas (Nota) que utilizam um padrão de caracter (`char`) para as notas – por exemplo, 'A', 'B' e 'C'– isto é, o tipo genérico de nota deve ser utilizado como caracter. Garanta que um mestrando só possa ser instanciado com todas as suas informações básicas de Aluno, com o nome do orientador e coleção de notas instanciada;

- **(0.4)** Na classe Aluno, adicione um método abstrato `getNota` que recebe como parâmetro o nome (String) da disciplina e deve recuperar e retornar o objeto Nota mais recente referente a nota do aluno na disciplina. Implemente este método nas subclasses de Aluno. Caso não houver um registro de nota para a disciplina informada, retorne *null*. Caso existam mais de um registro para a disciplina, retorne aquela vinculada a uma turma que possui o maior valor de semestre (compare Strings com o método `compareTo` do objeto String para descobrir qual é o maior);

- **(0.5)** Adicione na classe Turma, um método de classe denominado `matricular`, que recebe como parâmetro um objeto de turma e um objeto de aluno. 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 graduação 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). Deve-se procurar por alguma posição com valor nulo no vetor de alunos e caso seja encontrada uma posição deve-se associar o objeto aluno a posição do vetor. Este método deverá retornar um valor booleano indicando se a matricula foi efetivada ou não.

- **(0.3)** 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 cinco coleções de dados (utilize qualquer coleção): 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;

- **(0.3)** Ainda para a classe Universidade, crie um método denominado `getDiscentes` que cria uma coleção com todos os alunos (graduandos e mestrandos) e a retorna;

- **(0.8)** Crie uma janela/formulário que permite a visualização de cursos já cadastrados e a realização do cadastro de novos cursos. O formulário de cadastro deve conter os campos de preenchimento para nome do curso e um elemento de seleção (*radio button*, *checklist*, ou caixa de seleção) com os níveis de graduação e mestrado. Além disso, deve existir um botão com rótulo "Adicionar" que confirma se um nome foi digitado e se uma opção de nível foi escolhida e realizada o cadastro do novo curso. A parte de visualização dos cursos já cadastros pode ser implementada como uma tabela que é atualizada a cada nova inserção ou como um subformulário que exibe um registro de curso por vez, permitindo a iteração sobre eles através de botões.