UNIVERSIDADE
FEDERAL DO CEARÁ
Setembro de 2024

Disciplina: Programação Orientada à Objetos **Semestre:** 2024.1
Professor: Filipe Fernandes dos Santos Brasil de Matos

Aluno: _____ **Matrícula:** _____

Instruções para uma boa prova:

- **A avaliação é individual e não é pesquisada. É proibida a comunicação e a troca de materiais entre alunos, bem como consultas a materiais bibliográficos, cadernos ou anotações de qualquer espécie;**
- **Use caneta esferográfica de tinta azul ou preta para responder às questões. Questões respondidas a lápis serão corrigidas, porém não serão aceitas reclamações sobre as correções;**
- **O aluno só poderá entregar a prova após 40 minutos do início de sua realização. Após esse momento, nenhum outro aluno poderá entrar para realizar a prova;**
- **Coloque seu nome e matrícula também nas folhas de respostas. Essa identificação é importante para vincular as respostas aos alunos que a produziram;**
- **Por favor, escreva com letra legível. Serão corrigidas somente as respostas compreensíveis;**
- **Um arquivo com os código-fontes deverão ser comprimidos em formato zip e enviados para mim via SIGAA ao final da prova. Neste momento, chame o professor para acompanhar o envio do material.**
- **O arquivo com os código-fontes deve estar estruturado da seguinte maneira:**
  a. **\[matricula]\ (pasta raiz com o numero da matricula do aluno)**
  b. **\[matricula]\qX\ (um subpasta para cada questão prática, onde X é o número da questão)**
  c. **Dentro de \[matricula]\qX\ você adiciona todos arquivos que respondem a questão X. ATENÇÃO as particularidades de cada questão.**
  d. **Não é necessário inserir arquivos .class e classes Main nas respostas das questões**

2ª Atividade Avaliativa Parcial (2ª AP)

1. **Assinale VERDADEIRO (V) ou FALSO (F) para as afirmativas abaixo: (1.0 ponto)**
   a. **A palavra-chave *super* em Java é usada para acessar os membros da classe pai a partir da classe filha.**
   b. **Atributos *static* são inicializados quando uma instância da classe é criada.**
   c. **Uma classe abstrata não pode ser instanciada diretamente.**
   d. **Interfaces em Java podem herdar múltiplas interfaces.**
   e. **A ordem dos blocos *catch* é irrelevante ao capturar exceções de diferentes tipos.**

a)	b)	c)	d)	e)

2. **Responda as perguntas abaixo resumidamente: (2.0 pontos)**
    a. **Qual a diferença entre herança simples e herança múltipla? Como o Java lida com esses conceitos?**
    b. **O que ocorre se um método *static* tenta acessar um atributo de instância da classe onde ele está declarado? Como tal problema pode ser resolvido?**
    c. **Cite e explique uma diferença e uma semelhança entre interfaces e classes abstratas em Java.**
    d. **Explique o propósito da palavra-chave throws na assinatura de métodos em Java.**

3. **Crie uma classe estática *MatrixUtils* que possui um método estático denominado *findMaxInEachColumn* que recebe uma matriz de inteiros positivos de ordem NxM e retorna um vetor com o maior valor de cada coluna da matriz. (2.0 pontos)**

4. **Sobre o assunto "Tratamento de Exceções", pede-se: (2.0 pontos)**

    a. **Desenvolva uma classe *Usuario* com as seguintes características: 1) Dois atributos: Um do tipo inteiro privado que salva a *senha* do usuário e um do tipo *String* privado que salva o *nome* do usuário. 2) Quatro métodos: Dois métodos *gets/sets* para cada atributo supracitado.**
    b. **Desenvolva uma exceção chamada *NomeInvalidoException* que deve ser lançada se o nome do usuário informado estiver vazio. A exceção deve ser tratada na hora do lançamento e o tratamento consiste em imprimir a mensagem: "O nome do usuário não pode ser vazio".**
    c. **Desenvolva uma exceção chamada *SenhaInvalidaException* que deve ser lançada se a senha do usuário iniciar com um valor 0. A exceção deve ser repassada para ser tratada pelo invocador com a mensagem padrão "A senha não pode iniciar com 0". OBS: Assuma uma senha de quatro dígitos.**

5. **Sobre o assunto "Interfaces e Classes Abstratas", pede-se: (3.0 pontos)**

    a. **Desenvolva uma classe abstrata *AbstractSeries* com essas características: 1) Dois atributos: Um inteiro protegido para armazenar o comprimento de uma série e Um inteiro constante para salvar o valor inicial da série (42); 2) Dois métodos: Um construtor para inicializar os atributos supracitados; Um abstrato *generateSeries*() que retorna um vetor que segue uma dada série numérica.**
    b. **Desenvolva duas classes concretas que estendam *AbstractSeries*:**
        i. **ArithmeticProgression: Para criar uma Progressão Aritmética (PA). Esta classe recebe a razão e o comprimento da série no construtor e deve implementar o método *generateSeries*() para criar um vetor de inteiros que obedece uma PA.**
        ii. **GeometricProgression: Para criar uma Progressão Geométrica (PG). Esta classe recebe a razão e o comprimento da série no construtor e deve implementar o método *generateSeries*() para criar um vetor de inteiros que obedece uma PG.**

UNIVERSIDADE
FEDERAL DO CEARÁ
Setembro de 2024

Disciplina: Programação Orientada à Objetos **Semestre:** 2024.1
Professor: Filipe Fernandes dos Santos Brasil de Matos

Aluno: _____ **Matrícula:** _____

Instruções para uma boa prova:

- **A avaliação é individual e não é pesquisada. É proibida a comunicação e a troca de materiais entre alunos, bem como consultas a materiais bibliográficos, cadernos ou anotações de qualquer espécie;**
- **Use caneta esferográfica de tinta azul ou preta para responder às questões. Questões respondidas a lápis serão corrigidas, porém não serão aceitas reclamações sobre as correções;**
- **O aluno só poderá entregar a prova após 40 minutos do início de sua realização. Após esse momento, nenhum outro aluno poderá entrar para realizar a prova;**
- **Coloque seu nome e matrícula também nas folhas de respostas. Essa identificação é importante para vincular as respostas aos alunos que a produziram;**
- **Por favor, escreva com letra legível. Serão corrigidas somente as respostas compreensíveis;**
- **Um arquivo com os código-fontes deverão ser comprimidos em formato zip e enviados para mim via SIGAA ao final da prova. Neste momento, chame o professor para acompanhar o envio do material.**
- **O arquivo com os código-fontes deve estar estruturado da seguinte maneira:**
  a. **\[matricula]\ (pasta raiz com o numero da matricula do aluno)**
  b. **\[matricula]\qX\ (um subpasta para cada questão prática, onde X é o número da questão)**
  c. **Dentro de \[matricula]\qX\ você adiciona todos arquivos que respondem a questão X. ATENÇÃO as particularidades de cada questão.**
  d. **Não é necessário inserir arquivos .class e classes Main nas respostas das questões**

2ª Atividade Avaliativa Parcial (2ª AP)

1. **Assinale VERDADEIRO (V) ou FALSO (F) para as afirmativas abaixo: (1.0 ponto)**
   a. **Uma classe derivada pode herdar de múltiplas classes em Java.**
   b. **É possível acessar diretamente uma constante de uma classe sem criar uma instância da classe.**
   c. **Uma classe *final* pode ser estendida em Java.**
   d. **Classes abstratas não podem ter métodos concretos (implementados).**
   e. **É possível lançar uma exceção manualmente usando a palavra-chave *throw*.**

a)	b)	c)	d)	e)

2. **Responda as perguntas abaixo resumidamente: (2.0 pontos)**
    a. **Explique a diferença entre sobrecarga e sobrescrita de métodos em Java. Como isso se relaciona com o polimorfismo?**
    b. **Explique o que são constantes em Java e como elas são declaradas. Cite uma vantagem de usar constantes em um programa?**
    c. **Por que uma classe abstrata pode ter construtores, mas não pode ser instanciada?**
    d. **O que é um bloco finally, e como ele se comporta em relação aos blocos *try* e *catch*?**

3. **Crie uma classe estática *MatrixUtils* que possui um método estático denominado *transpose* que recebe uma matriz de inteiros de ordem NxM e retorna a transposta da matriz. DICA: A transposta de uma matriz é uma nova matriz obtida ao trocar suas linhas por colunas e vice-versa. (2.0 pontos)**

4. **Sobre o assunto "Tratamento de Exceções", pede-se: (2.0 pontos)**

    a. **Desenvolva uma classe *Filme* com as seguintes características: 1) Dois atributos: Um do tipo inteiro privado que armazena o *ano* de lançamento do filme e um do tipo *String* privado que armazena o *nome* do filme. 2) Dois métodos: um construtor que inicializa os dois atributos supracitados e um método *imprimeDetalhes* que retorna uma *String* contendo as informações do filme. O formato da String deve ser assim: "Filme: <nome>; Ano <ano>".**
    b. **Desenvolva uma exceção chamada *FilmeInvalidoException* que deve ser lançada no momento da impressão se o nome do filme informado estiver vazio. A exceção deve ser tratada na hora do lançamento e o tratamento consiste em imprimir a mensagem: "O nome do filme não pode ser vazio".**
    c. **Desenvolva uma exceção chamada *AnoInvalidoException* que deve ser lançada se o ano de lançamento do filme for negativo no momento da impressão. A exceção deve ser repassada para ser tratada pelo invocador com a mensagem padrão "O ano do filme não pode ser negativo".**

5. **Sobre o assunto "Interfaces e Classes Abstratas", pede-se: (3.0 pontos)**

    a. **Crie uma interface *MatrixOperations* com um método: *performOperation()* que recebe duas matrizes de inteiros de tamanho NxM e retorna uma matriz de inteiro de tamanho NxN.**
    b. **Desenvolva duas classes concretas que implementem *MatrixOperations*:**
        i. ***SumMatrix*: Implementa o método *performOperation*() para somar as duas matrizes informadas e retornar o resultado da operação.**
        ii. ***SubMatrix*: Implementa o método *performOperation*() para subtrair as duas matrizes informadas e retornar o resultado da operação.**

    DICA: Somar/subtrair duas matrizes envolve adicionar/subtrair os elementos correspondentes de cada matriz para criar uma nova matriz com o mesmo número de linhas e colunas.