MAUÁ	PREENCHA SEUS DADOS NOS CAMPOS COM X	
Disc.: **ECM251 – LINGUAGEM DE PROGRAMAÇÃO I**		
Curso: **ENGENHARIA DA COMPUTAÇÃO**		

Aluno: X				
Curso: X	Série: X	Período: X	RG: X	
São Caetano do Sul, 19 de setembro de 2023.			RA: X	
Assinatura:			Nota:	

Instruções da Prova

1. Esta prova é individual e prática, devendo ser realizada nos computadores do IMT, sendo permitido ao aluno, se assim desejar, utilizar seu próprio computador, sob sua inteira responsabilidade;
2. Não poderá haver acesso à Internet, sob nenhuma circunstância, exceto ao Open LMS do próprio aluno e, mesmo assim, somente em duas etapas: para receber ("baixar") as questões de sua prova na máquina em uso; e para entregar ("subir") as resoluções das questões de sua prova no devido local de entrega, na mesma plataforma.

   IMPORTANTE: O aluno deverá informar ao professor quando fará os dois acessos permitidos a ele ao Open LMS, pela Internet;
3. Poderá haver consulta a qualquer material do próprio aluno, seja ele físico (livros, artigos, material de aula etc.) ou virtual (livros, artigos, material de aula, exercícios, resoluções etc.), desde que esse material esteja previamente armazenado em seu computador ou em qualquer dispositivo de armazenamento externo (*pendrive*, *hd* externo etc.).

   IMPORTANTE: Não será permitido o acesso pela Internet a pastas compartilhadas (*Google Drive*, *OneDrive* etc.), nem a repositórios virtuais (*GitHub*, *GitLab*, *BitBucket* etc.), mesmo sendo de posse e administração do próprio aluno;
4. O aluno deverá responder às questões da prova no próprio arquivo da prova (*.docx*) devidamente identificado (RA e Nome completo do Aluno) e quando for o caso, gerando os códigos necessários, somente na linguagem de programação JAVA e utilizando sempre e somente o paradigma da Programação Orientada à Objetos visto nas aulas;
5. Para realizar a entrega da prova na plataforma Open LMS, o aluno deverá gerar e entregar um único arquivo compactado (*.rar* ou *.zip*), tendo seu RA e seu NOME completo como nome desse arquivo. Nesse arquivo compactado o aluno deverá fornecer, obrigatoriamente, os seus dados acadêmicos preenchidos na prova, bem como a resolução das questões nela solicitadas (*.docx*), além dos arquivos e códigos gerados em suas resoluções, uma pasta para cada questão, contendo as suas classes e demais arquivos que possibilitem sua posterior execução pelo professor, durante a resolução.

   IMPORTANTE: Na correção, para executar o código gerado pelo aluno em sua prova, o professor seguirá exatamente as instruções fornecidas pelos alunos e contidas nas resoluções das provas! Caso não obtenha sucesso, a questão será considerada errada.
6. Não poderá haver troca de informações, nem de materiais, sejam físicos ou virtuais, entre os alunos durante a prova;
7. Não é permitido ao estudante se ausentar da sala antes da entrega da prova;
8. Celulares e outros equipamentos eletrônicos devem permanecer desligados enquanto o estudante estiver na sala;
9. O tempo limite para realização da prova é de **90** minutos;
10. Mantenha sobre a carteira apenas um documento com foto, caneta, lápis e borracha;
11. O entendimento das questões faz parte da avaliação;
12. O tempo mínimo de permanência na sala é de **30** minutos;
13. O estudante que chegar atrasado em até **30** minutos do início da prova poderá fazê-la.

Questões de 1 à 3: (*1 ponto cada*)

Verifique se cada afirmação, de 1 à 9, é *Verdadeira* ou *Falsa* e depois responda as questões, de 1 à 3:

AFIRMAÇÕES:

A1 – Na aula 17 sobre *Criptografia em Java*, toda vez que era executado o código para cifrar uma mensagem utilizando a classe *CryptoAES* eram gerados arquivos de chaves públicas e privadas diferentes dos gerados nas execuções anteriores.

A2 – Na aula 17 sobre *Criptografia em Java*, a classe *CryptoRSA* se utiliza de classes próprias do *Java* para gerar as chaves públicas e privadas, bem como para realizar a cifragem com o algoritmo *RSA*.

A3 – Na aula 14 sobre *Manipulação de Ícones, Senhas e Datas*, utilizando-se a classe *Locale* do *Java*, é possível converter data e hora lidas do computador da formatação de Data/Hora local para a formatação de Data/Hora de qualquer outro país, informando, apenas, qual o idioma falado naquele país.

A4 – Na aula 15 sobre *Internacionalizar em Java*, para alterar o idioma de uma aplicação, deve ser criado um único arquivo *Bundle*, contendo a tradução de cada palavra utilizada na aplicação para todos os idiomas que se deseja internacionalizar.

A5 – Na aula 15 sobre *Internacionalizar em Java*, é possível se criar um arquivo da aplicação do tipo *.jar*, desde que todos os arquivos *.properties* utilizados por ela estejam presentes e indicados no *classpath* para serem carregados.

A6 – Na aula 16 sobre *Acesso a Arquivos de Texto em Java*, no *Exemplo 2* (início dele na página 7), pelo fato de se estar utilizando a classe *Scanner* para entrada de dados do tipo *int*, *String* e *double*, e gravar esses valores digitados no arquivo *clients.txt*, não é possível realizar, posteriormente, a leitura desse mesmo arquivo caractere a caractere (*char*), ou seja, *byte* a *byte*.

A7 – Na aula 18 sobre *Testes Unitários*, para se realizar o cálculo do método de *checksum* através da *Soma e Complemento 2*, é necessário converter manualmente, através de um método de conversão de *char* para binário, cada caractere lido de um vetor de *char* para seu valor binário antes de realizar as operações aritméticas pertinentes, como soma e complemento.

A8 – Na aula 18 sobre *Testes Unitários*, para se realizar o cálculo do *checksum* através da *Soma e Complemento 2*, utiliza-se o mesmo método para o cálculo, lendo caractere por caractere, vindos de um vetor do tipo *char* ou vindos de um arquivo texto, pois, afinal de contas, a informação está armazenada em caracteres (*bytes*) para ambos os meios.

A9 – Na aula 19 sobre *TDD e JUnit*, foram apresentados o estilo de programação *TDD*, baseado em *Testes Unitários*, e o *framework JUnit*, para a realização de testes unitários automatizados em *Java*. Com isso, pode-se concluir que para se utilizar o *TDD*, pode-se ou não utilizar o *JUnit* em conjunto na *IDE*, dependendo do caso, porém, para se utilizar o *JUnit*, é necessário, sempre, utilizar o método *TDD* em conjunto.

QUESTÕES:

Q1 – Assinale a única alternativa correta:

a) As afirmações A1, A2 e A3 são falsas;
b) Apenas a afirmação A2 é verdadeira, dentre A1, A2 e A3;
c) Apenas a afirmação A1 é falsa, dentre A1, A2 e A3;
d) As afirmações A1, A2 e A3 são verdadeiras;
e) N.D.A.

Q2 – Assinale a única alternativa correta:

a) As afirmações A4, A5 e A6 são verdadeiras;
b) Apenas a afirmação A6 é falsa, dentre A4, A5 e A6;
c) As afirmações A4, A5 e A6 são falsas;
d) Apenas a afirmação A5 é verdadeira, dentre A4, A5 e A6;
e) N.D.A.

Q3 – Assinale a única alternativa correta:

a) Apenas a afirmação A7 é falsa, dentre A7, A8 e A9;
b) As afirmações A7, A8 e A9 são verdadeiras;
c) Apenas a afirmação A8 é verdadeira, dentre A7, A8 e A9;
d) As afirmações A7, A8 e A9 são falsas;
e) N.D.A.

Questão 4: (*4 pontos*)

Gerado por qualquer *IDE*, fornecer o código fonte, em *Java*, das classes necessárias (incluindo a de execução), implementadas sob o paradigma da *Programação Orientada a Objetos*, além do programa ***.jar*** que as executa diretamente, objetivando resolver o problema descrito a seguir.

ATENÇÃO:

i. Se o programa *.jar* fornecido não executar automaticamente a aplicação desenvolvida, a resolução da questão será invalidada (0 ponto);
ii. As classes desta questão (arquivos *.java*), deverão ser compactadas em um único arquivo (*.zip* ou *.rar*), em conjunto com o respectivo arquivo *.jar* funcional, com os arquivos de trabalho da aplicação (arquivo de chave criptográfica e arquivo cifrado gerado), além deste arquivo *.docx* da prova, contendo todas as respostas para as outras questões e a devida identificação do aluno e entregue em resposta à tarefa do *OPEN LMS* da prova.

Problema:

Utilizando-se apenas os códigos vistos e praticados em aula, nesta disciplina, através de uma aplicação desenvolvida pelo próprio aluno, deseja-se calcular, pelo método de *Soma e Complemento 2*, o valor do *checksum* dos 8 (oito) caracteres: '#', 'V', 'a', 'i', 'I', 'M', 'T', '!', preenchidos previamente num vetor de *char* de tamanho 9 (via *hard code*), dispostos em suas primeiras posições, acrescentando, automaticamente, pelo método, ao seu final (no índice 8), o resultado do *checksum* calculado.

Após isto, deve-se cifrar esse vetor (caracteres e *checksum*), através da criptografia *dummy*, vista e estudada em aula, apresentar o resultado em tela e salvar esses resultados nos arquivos gerados (arquivo de chave criptográfica e arquivo cifrado).

Questão 5: (*3 pontos*)

Num cenário de desenvolvimento de *software* em *Java*, **obrigatoriamente** utilizando a abordagem **TDD** – *Test-Driven Development*, porém, **não** podendo utilizar o *framework JUnit* por algum motivo, você foi incumbido de desenvolver um método para calcular o valor de **π** através da **Série de Leibniz**, tal como segue o embasamento matemático:

Série de Leibniz: $$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots = \frac{\pi}{4}$$

Notação da Série de Leibniz: $$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4}$$

Dada a estrutura de classes, as assinaturas dos métodos e a quantidade de iterações do *TDD* pré-definidas e fornecidas a seguir, preencha cada um deles com os códigos que você criaria para desenvolver o solicitado, passo a passo (*baby steps*), estando 100% aderente aos princípios *test first* do *TDD*: VERMELHO, VERDE e REFATORA:

CHAMADA:

```
// Não alterar os conteúdos desta classe
public class TddQuestao1
{       public static void main(String args[])
        {       SerieLeibnizTest  slt  =  SerieLeibnizTest( );
        }
}
```

Ordem cronológica dos códigos (*log*):

VERMELHO:

```
public class SerieLeibnizTest
{       public SerieLeibnizTest( )
        {       // Copie e cole aqui o seu código para o método em questão

        }
        // Copie e cole daqui para baixo o(s) seu(s) método(s) relativo(s) ao(s) caso(s) de teste(s): caso1Test( ), caso2Test( ) etc.

}

public class ValorDePi
{       public double calculaPelaSerieDeLeibniz(int n)          // Não alterar a assinatura do método
        {       // Copie e cole aqui o seu código para o método em questão

        }
}
```

VERDE - SIMPLES:

```
public class SerieLeibnizTest
{       public SerieLeibnizTest( )
        {       // Copie e cole aqui o seu código para o método em questão

        }
        // Copie e cole daqui para baixo o(s) seu(s) método(s) relativo(s) ao(s) caso(s) de teste(s): caso1Test( ), caso2Test( ) etc.

}

public class ValorDePi
{       public double calculaPelaSerieDeLeibniz(int n)          // Não alterar a assinatura do método
        {       // Copie e cole aqui o seu código para o método em questão

        }
}
```

REFATORA:

```
public class SerieLeibnizTest
{       public SerieLeibnizTest( )
        {       // Copie e cole aqui o seu código para o método em questão

        }
        // Copie e cole daqui para baixo o(s) seu(s) método(s) relativo(s) ao(s) caso(s) de teste(s): caso1Test( ), caso2Test( ) etc.

}

public class ValorDePi
{       public double calculaPelaSerieDeLeibniz(int n)          // Não alterar a assinatura do método
        {       // Copie e cole aqui o seu código para o método em questão

        }
}
```

VERDE - FINAL:

```
public class SerieLeibnizTest
{       public SerieLeibnizTest( )
        {       // Copie e cole aqui o seu código para o método em questão

        }
        // Copie e cole daqui para baixo o(s) seu(s) método(s) relativo(s) ao(s) caso(s) de teste(s): caso1Test( ), caso2Test( ) etc.

}

public class ValorDePi
{       public double calculaPelaSerieDeLeibniz(int n)          // Não alterar a assinatura do método
        {       // Copie e cole aqui o seu código para o método em questão

        }
}
```

>>> ENTREGA DO CÓDIGO CONFIÁVEL, LIMPO E TESTADO (TESTE UNITÁRIO)!