UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

PROCESSO SELETIVO – 005/2022

Área de Conhecimento: Engenharia de Software

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 1: Padrões de Projetos

O problema de manter a implementação fornecida pela questão é que cada método que precisar registrar eventos no sistema poderá criar sua própria instância de *Logger*. Segundo VALENTE (2020 – Capítulo 6), isto gera uma proliferação de objetos *Logger* e que, neste caso, todo novo objeto instanciado acabará sobrescrevendo o conteúdo do arquivo *logs.txt*. Como a finalidade é que este arquivo *logs.txt* registre todas as ações do sistema, é importante que exista, no máximo, uma única instância dessa classe e que ela seja usada em todas as partes do sistema que precisam registrar algum evento.
O uso do padrão de projeto *Singleton* resolve este problema, pois define como implementar classes que terão no máximo uma instância (VALENTE, 2020 – Capítulo 6). A seguir é apresentada uma nova versão da classe* *Logger* que atende ao padrão *Singleton*:

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class Logger {
  FileWriter f;

  private Logger() throws IOException { // private proíbe clientes chamar new Logger()
    f = new FileWriter("logs.txt"); //abre um arquivo chamado logs.txt
  }

  private static Logger instance; // instância única

  public static Logger getInstance() throws IOException {
    if (instance == null) // 1a vez que chama-se getInstance
      instance = new Logger();
    return instance;
  }

  public void println(String msg) throws IOException {
    PrintWriter p = new PrintWriter(f);
    p.println(msg);//efetua a escrita da mensagem recebida no arquivo
  }

  public void close() throws IOException {
      f.close();
  }
 }
```

*A apresentação da classe não é mandatória segundo o enunciado, o candidato poderia apenas descrever o que deveria ser feito

A alteração requer transformar o construtor *default* em privado. Assim, um erro de compilação ocorrerá quando qualquer código fora da classe tentar instanciar esta classe (através de new Logger()). Além disso, um atributo estático armazena a instância única da classe. Quando uma instância de *Logger* for necessária, será preciso chamar o método público e estático getInstance() que será a única forma de obter uma instância desta classe.

Membros da Banca:

Carla Diacui Medeiros Berkenbrock

Fabiano Baldo

Rebeca Schroeder Freitas

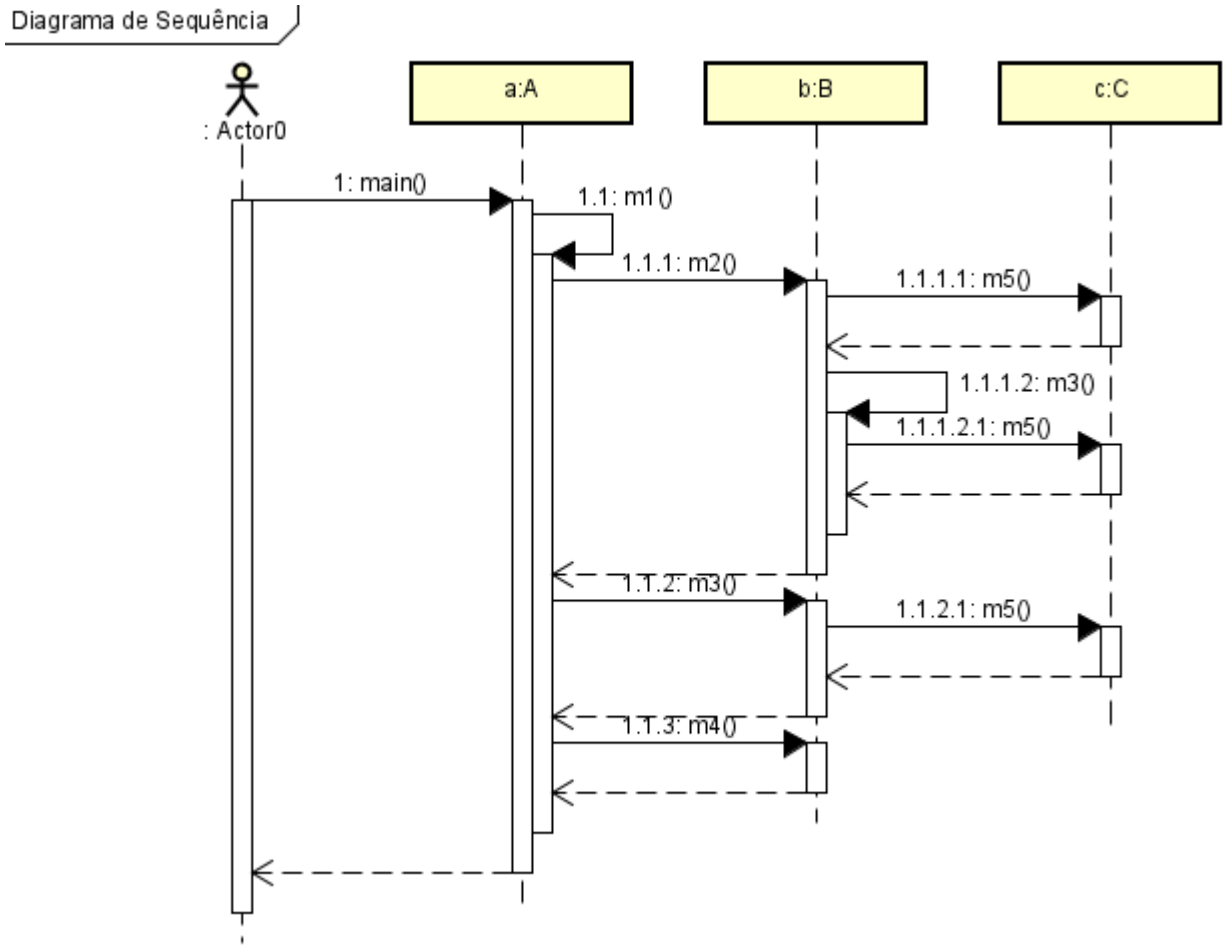
PROCESSO SELETIVO – 005/2022

Área de Conhecimento: Engenharia de Software

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 2: Modelagem de Sistemas

O diagrama de sequência que atende minimamente ao enunciado é apresentado a seguir. A construção do mesmo atende às diretrizes para sua elaboração apontadas por Sommerville (2003 – Capítulo 5) e Valente (2020 – Capítulo 4):



Membros da Banca:

Carla Diacui Medeiros Berkenbrock — **Fabiano Baldo**

Rebeca Schroeder Freitas

PROCESSO SELETIVO – 005/2022

Área de Conhecimento: Engenharia de Software

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 3: Ciclo de Vida de Software

Com base na página 41 do livro do Sommerville (2003):

1. Embora a ideia de envolvimento do cliente no processo de desenvolvimento seja atraente, seu sucesso depende de um cliente disposto e capaz de passar o tempo com a equipe de desenvolvimento, e que possa representar todos os stakeholders do sistema. Frequentemente, os representantes dos clientes estão sujeitos a diversas pressões e não podem participar plenamente do desenvolvimento de software.
2. Membros individuais da equipe podem não ter personalidade adequada para o intenso envolvimento que é típico dos métodos ágeis e, portanto, não interagem bem com outros membros da equipe.
3. Priorizar as mudanças pode ser extremamente difícil, especialmente em sistemas nos quais existem muitos stakeholders. Normalmente, cada stakeholder dá prioridades diferentes para mudanças diferentes.
4. Manter a simplicidade exige um trabalho extra. Sob a pressão de cronogramas de entrega, os membros da equipe podem não ter tempo para fazer as simplificações desejáveis.
5. Muitas organizações, principalmente as grandes empresas, passaram anos mudando sua cultura para que os processos fossem definidos e seguidos. É difícil para eles mudar de um modelo de trabalho em que os processos são informais e definidos pelas equipes de desenvolvimento.

Outro problema não técnico — que é um problema geral do desenvolvimento e da entrega incrementai — ocorre quando o cliente do sistema usa uma organização externa para desenvolver o sistema. O documento de requisitos do software é normalmente parte do contrato entre o cliente e o fornecedor. Como a especificação incrementai é inerente aos métodos ágeis, escrever contratos para esse tipo de desenvolvimento pode ser difícil.

Membros da Banca:

Carla Diacui Medeiros Berkenbrock

Fabiano Baldo

Rebeca Schroeder Freitas

PROCESSO SELETIVO – 005/2022

Área de Conhecimento: Engenharia de Software

PROVA ESCRITA – PADRÃO DE RESPOSTA

QUESTÃO 4: Engenharia de Requisitos

(Primeira parte) O que são histórias de usuário e quais são as suas partes?

Segundo VALENTE, M.T (Capítulo 3 - Requisitos) Documentos de requisitos tradicionais, como aqueles produzidos quando se usa Waterfall, possuem centenas de páginas e levam às vezes mais de um ano para ficarem prontos. Além disso, eles sofrem dos seguintes problemas: (1) durante o desenvolvimento, os requisitos mudam e os documentos ficam obsoletos; (2) descrições em linguagem natural são ambíguas e incompletas; então os desenvolvedores têm que voltar a conversar com os clientes durante o desenvolvimento para tirar dúvidas; (3) quando essas conversas intermediárias não ocorrem, os riscos são ainda maiores: no final da codificação, o cliente pode simplesmente concluir que esse não é mais o sistema que ele queria, pois suas prioridades mudaram, sua visão do negócio mudou, os processos internos de sua empresa mudaram, etc. Por isso, uma longa fase inicial de especificação de requisitos é cada vez mais rara, pelo menos em sistemas comerciais. Os profissionais da indústria que propuseram métodos ágeis perceberam — ou sofreram com — tais problemas e propuseram uma técnica pragmática para solucioná-los, que ficou conhecida pelo nome de Histórias de Usuários.

Uma história de usuário é composta por três partes, todas começando com a letra C e que vamos documentar usando a seguinte equação:

História de Usuário = Cartão + Conversas + Confirmação

O **Cartão** é usado pelos clientes para escrever, na sua linguagem e em poucas sentenças, uma funcionalidade que esperam ver implementada no sistema.

Conversas entre clientes e desenvolvedores, por meio das quais os clientes explicam e detalham o que escreveram em cada cartão. Para facilitar essas conversas, métodos ágeis incluem nos times de desenvolvimento um representante dos clientes, que participa do time em tempo integral.

Confirmação, que é basicamente um teste de alto nível — de novo especificado pelo cliente — para verificar se a história foi implementada conforme esperado. Ele inclui a descrição dos cenários, exemplos e casos de teste que o cliente irá usar para confirmar a implementação da história. Por isso, são também chamados de testes de aceitação de histórias.

(Segunda parte) Ainda a respeito de História de Usuário, suponha uma rede social como o Facebook. (1) Escreva um conjunto de 5 histórias para essa rede, assumindo o papel de um usuário típico; (2) Pense agora em mais um papel de usuário e escreva pelo menos duas histórias para ele.

VALENTE, M.T (2020) apresenta exemplos de histórias para um sistema de controle de bibliotecas. Elas estão associadas a três tipos de usuários: usuário típico, professor e funcionário da biblioteca. Uma possível solução para a segunda parte dessa questão é a seguinte:

(1)

Como usuário normal, eu gostaria de postar mensagens.

Como usuário normal, eu gostaria de incluir imagens nas minhas mensagens.

Como usuário normal, eu gostaria de dar "likes" em mensagens de outros usuários

Como usuário normal, eu gostaria de "compartilhar" mensagens de outros usuários

Como usuário normal, eu gostaria de obter dados sobre minhas mensagens (número de visualizações, etc)

(2)

Como administrador da rede, eu gostaria de apagar mensagens

Como administrador da rede, eu gostaria de remover usuários com perfis inadequados

Membros da Banca:

Carla Diacui Medeiros Berkenbrock

Fabiano Baldo

Rebeca Schroeder Freitas

Assinaturas do documento

Código para verificação: **B9WBC186**

Este documento foi assinado digitalmente pelos seguintes signatários nas datas indicadas:

REBECA SCHROEDER (CPF: 036.XXX.099-XX) em 12/12/2022 às 08:31:20
Emitido por: "SGP-e", emitido em 13/07/2018 - 14:59:17 e válido até 13/07/2118 - 14:59:17.
(Assinatura do sistema)

CARLA DIACUI MEDEIROS BERKENBROCK (CPF: 025.XXX.559-XX) em 12/12/2022 às 08:43:06
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:39:40 e válido até 30/03/2118 - 12:39:40.
(Assinatura do sistema)

FABIANO BALDO (CPF: 028.XXX.209-XX) em 12/12/2022 às 10:35:28
Emitido por: "SGP-e", emitido em 30/03/2018 - 12:47:06 e válido até 30/03/2118 - 12:47:06.
(Assinatura do sistema)

Para verificar a autenticidade desta cópia, acesse o link https://portal.sgpe.sea.sc.gov.br/portal-externo/conferencia-documento/VURFU0NfMTIwMjJfMDAwNTU1MTZfNTU2MDNfMjAyMl9COVdCQzE4Ng== ou o site https://portal.sgpe.sea.sc.gov.br/portal-externo e informe o processo **UDESC 00055516/2022** e o código **B9WBC186** ou aponte a câmera para o QR Code presente nesta página para realizar a conferência.