Universidade Federal Fluminense

Instituto de Ciência e Tecnologia

Departamento de Computação

Segunda Avaliação – Técnicas de Programação Avançada

Todas as questões devem utilizar Java

1) **3 Pontos.** Papelândia é um país que sempre evitou utilizar sistemas computacionais digitais por considerar que eles são inseguros. No entanto, recentemente diversos ministérios de Papelândia passaram a criar sistemas em Java, já que a linguagem possui mecanismos muito interessantes de tratar Exceções que possam ocorrer durante o processamento dos dados pessoais dos seus cidadãos. Numa primeira etapa deste processo de digitalização, as datas de nascimento, casamento e óbito de 15 cidadãos foram colocadas em um arquivo texto, nesta mesma ordem. Cada data é composta por 3 valores (dia, mês e ano) separados por espaço, ou seja, em cada linha devemos ter 9 valores. O funcionário que digitou as datas, porém, tem pouca experiência com computadores e pode ter digitado alguma coisa errada. Faça um programa em Java que possua uma classe Data (não pode utilizar a classe Date, padrão da linguagem). Essa classe deve implementar a interface Comparable e seu construtor deve receber 3 strings, correspondentes aos campos da data. Esse construtor da classe pode gerar 2 tipos de Exceções: DadosIncorretosException e DataInvalidaException. A primeira exceção ocorre quando os valores de qualquer uma das strings não é um número inteiro. A segunda exceção ocorre quando a data não corresponde a um dia real (ex: 30 de fevereiro). Considere que fevereiro sempre terá 28 dias; abril, junho, setembro e novembro, 30 dias; os demais meses, 31 dias. Além disso, quando os valores forem lidos do arquivo texto, a data de casamento deve ser sempre posterior à data de nascimento. A data de óbito deve ser posterior à de nascimento e de casamento. Quando uma data não atender a estas regras, deve ser gerada uma exceção DataAnteriorException, que não deve abortar o processamento do arquivo texto, assim como as duas exceções anteriores também não devem interromper definitivamente o processamento. Sempre que uma exceção ocorrer, as datas envolvidas na exceção devem ser escritas num arquivo texto “Exceções.txt” com uma mensagem de erro que identifique o problema ocorrido. O sistema não precisa fazer nada além de ler o arquivo e registrar as exceções que tenham ocorrido. O arquivo de texto que contém as datas dos 15 cidadãos deve ser criado e enviado junto com os códigos da questão.
2) **3,5 Pontos.** Um portal de notícias decidiu criar um serviço personalizado de assinatura de notícias. Cada cliente é representado por um número inteiro (ID), uma lista de assuntos do seu interesse (ex: esportes, política, cultura etc) e um valor de débito mensal. O cliente pode, a qualquer momento, decidir acrescentar um novo assunto aos interesses ou remover um assunto. Ele também pode solicitar receber uma notícia diária de **cada** um dos seus assuntos cadastrados para ele naquele momento (por exemplo: se ele tiver cadastrado esporte e política, vai receber duas mensagens: uma sobre esporte outra sobre política). Quando isso ocorre, o portal deve criar um objeto Noticia, mostrá-lo na tela (printando as mensagens correspondentes) e aumentar o débito do cliente de acordo com o(s) tipo(s) de assunto(s) cadastrados. Cada tipo de assunto tem um valor fixo diferente (por exemplo: quando um cliente recebe uma notícia de esporte, seu débito é

aumentado em R$1,20; quando recebe notícia de política, seu débito é aumentado em R$1,10). Esse objeto Noticia deve seguir o padrão de projeto Decorator, onde cada classe Decorator específica está relacionada a um dos possíveis assuntos. Os métodos de cada um desses Decorators para mostrar uma notícia na tela devem apenas fazer um println com o nome do assunto relacionado (ex: “esporte”, “política”, “cultura”). Faça um sistema que implemente essa plataforma de assinatura de notícias.

3) **3,5 Pontos.** Uma loja virtual está muito preocupada com o sistema de entrega das mercadorias compradas pelo site. Quando um cliente realiza uma compra, seu estado é considerado “aberta” até que a operadora de cartão aprove a compra. Seu estado então passa para “aprovada”. Quando as mercadorias são despachadas, o estado da compra muda para “despachada” até que o cliente receba a compra (“recebida”). Crie uma classe Compra que tenha apenas um ID informado pelo usuário, um atributo referente ao seu estado, a hora em que o estado foi atribuído (classe java.time.LocalTime), o construtor e pelo menos os métodos getId, getEstado, setAprovada, setDespachada e setRecebida. Faça um sistema que permita cadastrar uma nova compra e colocá-la num arraylist Compras. Faça 3 threads que verifiquem continuamente esse arraylist. Cada thread deve cuidar de apenas uma transição (ex: de “aberta” para “aprovada” ou de “despachada” para “recebida”). As threads vão fazer as transições de acordo com a contagem de tempo abaixo. Sempre que houver uma transição, a thread correspondente deve mostrar na tela o ID da compra em questão e seu estado atual (após a transição). O sistema deve permitir apenas a criação de novas compras, ficando as mudanças de estado a cargo das threads. O método “run()” das threads deve conter um loop verificando continuamente se a thread foi interrompida ou não (use o método isInterrupted()). Quando o usuário não quiser mais cadastrar novas compras, o programa deve interromper as 3 threads e encerrar.

   A compra passa para “aprovada” 3 segundos após ter sido “aberta” (criação do objeto compra).
   A compra passa para “despachada” 8 segundos após ter sido “aprovada”.
   A compra passa para “recebida” 15 segundos após ter sido “despachada”.
   OBS: as threads devem fazer a transição de estados o quanto antes e o código deve utilizar técnicas adequadas de programação concorrente em Java.