Projeto Prático – Barbeiro Dorminhoco

Professor:	Alexandre Huff
Disciplina:	Programação Orientada a Objetos 2
Meio para entrega:	Moodle

Descrição da Atividade

A atividade consiste em implementar um problema clássico de programação concorrente, o Barbeiro Dorminhoco. Este problema está abstraído no contexto de uma barbearia em que trabalham um barbeiro e um recepcionista. Existem 5 cadeiras disponíveis em uma sala de espera que os clientes devem aguardar até que o barbeiro esteja livre para iniciar o corte de cabelo. Novos clientes chegam na barbearia em intervalos aleatórios e somente entram na barbearia se existirem cadeiras vagas na sala de espera. Se não existir cadeira disponível, o cliente desiste e vai embora.

Utilize a API de concorrência da alto nível do Java que foi vista em sala de aula. Use a classe `ReentrantLock` e a interface `Condition` para gerenciar a sincronização entre as threads. A fila de clientes pode ser abstraída utilizando uma fila FIFO que deve ser *thread-safe*.

Entrega: Compactar o diretório **src** e enviar pelo Moodle.

Especificação

- Classe `Barbeiro` – `Runnable`
  - Sincronização com `await`/`signal`
  - Atributos:
    - `Cliente` que o barbeiro está cortando o cabelo
  - O barbeiro adormece em sua cadeira (`await`) sempre que não há clientes para serem atendidos
  - Ao ser acordado por um cliente, o barbeiro retira o primeiro cliente da fila de espera e inicia o corte de cabelo.
  - O barbeiro corta o cabelo do cliente por um tempo aleatório entre 1ms e 10ms. Use o método `sleep` para simular o corte de cabelo.
  - Após o barbeiro terminar o corte de cabelo, ele sinaliza para o cliente que o corte está pronto (`signal`).
  - O barbeiro continua cortando cabelo dos clientes enquanto a barbearia estiver aberta ou existirem clientes na fila de espera.
  - Após encerrar o expediente o barbeiro vai embora (thread é encerrada).
- Classe `SalaEspera`
  - Simula uma fila de cadeiras FIFO *thread-safe* com tamanho máximo de 5 cadeiras.
  - Use uma implementação de fila disponível no pacote `java.util.concurrent`.

- Classe `Cliente` – `Runnable`
  - Sincronização com `await`/`signal`
  - Se tiver cadeira vaga na fila de espera, o cliente espera (`await`) a sua vez.
  - Se não tiver onde sentar, o cliente vai embora e não volta mais (encerra a thread)
  - Se o barbeiro estiver dormindo, o cliente senta na cadeira, acorda o barbeiro (`signal`) e depois aguarda a sua vez (`await`).
- Classe `Recepcionista` – `Runnable`
  - Instancia novos clientes em intervalos de tempo aleatórios entre 1ms e 10ms enquanto a barbearia estiver aberta.
- Classe `Barbearia` – responsável por instanciar a aplicação
  - Método `abrir()` deve ser usado para abrir a barbearia e iniciar os trabalhos (instanciar barbeiro e recepcionista)
  - Método `fechar()` deve ser usado para fechar a barbearia e encerrar os trabalhos. Novos clientes não podem ser aceitos após fechar a barbearia. Todos os clientes que estiverem dentro da barbearia no instante em que a barbearia foi fechada, devem ter os seus cabelos cortados antes de encerrar a aplicação.
  - Método `main()` cria a aplicação, abre a barbearia, aguarda 1 minuto e solicita o fechamento da barbearia.

Dica 1: Durante o desenvolvimento, apresente mensagens de *log* que permitem identificar cada passo dos clientes e barbeiro, de modo que seja possível visualizar o que está acontecendo.
Utilize a classe `java.util.logging.Logger` para gerar as mensagens de log no console.
Não utilizar `System.out.println()`. Durante o desenvolvimento utilize o nível de debug (FINE). Quando o código estiver pronto, altere o nível de log para o padrão (INFO). Assim não é necessário comentar as linhas de *log* que você não quer que apareça na tela.

Dica 2: Durante o desenvolvimento, aumente o tempo que a recepcionista instancia novos clientes e que o barbeiro corta os cabelos. Utilize um tempo aleatório entre 1s e 3s para facilitar a visualização dos *logs*. Quando o código estiver pronto para testar e entregar, retorne os tempos de corte e instanciação de clientes conforme descrito na especificação das classes.