**1. O que são threads? Quais são as vantagens do uso de threads?**

Thread são tarefas realizadas por um programa, é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrencialmente.

Vantagens:

Capacidade de resposta, permite continuar a execução mesmo se parte de um processo for bloqueada.

Compartilhamento de recursos, Thread compartilham recursos do processo original (métodos e variáveis).

Economia, geralmente, têm custo menor que a criação de um processo, o chaveamento de thread causa menos sobrecarga que o de contexto.

Escalabilidade, pode obter vantagem do paralelismo em arquiteturas multiprocessadas.

**2. Por que a execução de múltiplas threads pode gerar concorrência?**

Eles concorrem pelo tempo de execução do processador e de recursos compartilhados.

**3. O que é uma seção crítica de um código?**

É aquela que contém um código que manipula um recurso compartilhado, é uma aplicação concorrente, ela é crítica pois é ela quem pode gerar alterações incompletas e causar inconsistência de dados

**4. O que são mutexes e como podem ser utilizados para proteger a concorrência?**

O mutex é uma técnica utilizada para evitar que dois processos ou threads tenham acesso simultaneamente a um recurso compartilhado. Ele realiza uma variável de bloqueio.

**5. O que são semáforos?**

Semáforos são ferramentas de sincronização que não requer espera ocupada.

**6. O que são monitores e como são implementados em Java?**

Abstração de alto nível para mecanismos de sincronização.

**7. O que são as Checked e as Unchecked Exceptions em Java?**

Checked exceptions – Representam exceções previsíveis, de acordo com a situação.

Uncheked exceptions (Runtime Exception) - representam exceções imprevisíveis que ocorrem em tempo de execução, mesmo não tendo que ser obrigatoriamente anunciadas podem ser tratadas.

**8. Descreva como as exceções em Java podem ser: anunciadas, lançadas, obtidas, tratadas e repassadas.**

Uma exceção **anunciada** (throws) define um caminho alternativo à execução do método chamado, quando o método em execução se depara com a exceção anunciada ele pode **lançar** (throw) a exceção, o que interrompe sua execução naquele momento. O chamador pode **tentar** (try) executar o método que anuncia a exceção, em caso dela ocorrer, pode **pegá-la** (catch) e **tratá-la** ou pode **repassar** a obrigação para o método que o chamar.