1. O que é uma exceção em Java e qual é o propósito de usá-las em programas?

Em Java, uma exceção é um evento que interrompe o fluxo normal de execução de um programa, como quando ocorre uma divisão por zero ou ao acessar um índice fora da faixa de um array, por exemplo. O propósito das exceções é lidar com erros e situações excepcionais de maneira estruturada, permitindo que o programa trate essas condições de forma apropriada, evitando quebras inesperadas ou resultados indesejados.

2. Diferença entre exceções verificadas e não verificadas em Java. Dê exemplos de cada uma.

Exceções Verificadas (checked): São exceções que o compilador obriga você a lidar explicitamente, seja através de tratamento (try-catch) ou declarando que seu método pode lançar a exceção (usando a palavra-chave throws). Exemplos incluem IOException e SQLException.

```
try {
        File file = new File("meuArquivo.txt");
        Scanner scanner = new Scanner(file);

  } catch (FileNotFoundException e) {
        System.err.println("Arquivo não encontrado: " + e.getMessage());
   }
```

No exemplo acima o construtor FileReader pode lançar uma exceção do tipo IOException caso ocorra algum problema ao tentar abrir o arquivo, como o arquivo não existir ou não ter permissões de leitura.

As exceções não verificadas (unchecked) são exceções de tempo de execução, não exigem tratamento obrigatório. Exemplos incluem NullPointerException e ArithmeticException.

```
int[] array = {1, 2, 3};
System.out.println(array[5]);
```

Isso lançará a exceção ArrayIndexOutOfBoundsException
No exemplo acima, o acesso a array[5] está além do tamanho do array, resultando em uma exceção ArrayIndexOutOfBoundsException.

3. Como lidar com exceções em Java? Quais são as palavras-chave e as práticas comuns para tratamento de exceções?

Lidar com exceções em Java desempenha um papel crucial no desenvolvimento de software, assegurando que um programa possa responder eficientemente a situações inesperadas. As principais palavras-chave são: try-catch, finally, throws, throw.

4. O que é o bloco "try-catch" em Java? Como ele funciona e por que é importante usá-lo ao lidar com exceções?

O bloco try-catch em Java é uma construção que permite que você lide com exceções durante a execução de um bloco de código. Ele é usado para envolver um conjunto de instruções que pode lançar exceções, permitindo que você capture e trate essas exceções de uma maneira controlada.

```
try {
    // Código que pode lançar exceções
} catch (TipoDeExcecao e) {
    // Tratamento da exceção
} finally {
    // Bloco opcional: Código a ser executado independentemente de ocorrer uma exceção ou não
}
```

5. Quando é apropriado criar suas próprias exceções personalizadas em Java e como você pode fazer isso? Dê um exemplo de quando e por que você criaria uma exceção personalizada.

Criar exceções personalizadas é apropriado para lidar com situações específicas que não são adequadamente representadas pelas exceções padrão fornecidas e personalizar a lógica de tratamento de erros no código. Em outras palavras, você pode estender a classe Exception ou uma de suas subclasses para criar sua própria exceção personalizada.

Um exemplo de quando criar uma exceção personalizada seria em um programa de pedidos, onde você pode ter uma classe Pedido com um método processar que verifica se o estoque é suficiente para atender ao pedido. Se não for, uma exceção personalizada, como EstoqueInsuficienteException, poderia ser lançada para indicar esse problema específico.

```
public class Pedido {
    public void processar() throws EstoqueInsuficienteException {
        // Verificar o estoque...

        if (estoqueInsuficiente) {
            throw new EstoqueInsuficienteException("Não há estoque suficiente para processar o pedido.");
        }

        // Processar o pedido normalmente...
    }
}
```