Tratamento de Falhas e Exceções

No desenvolvimento de software, é inevitável que programas apresentem erros em algum momento. Para garantir a robustez e a continuidade do funcionamento das aplicações, é fundamental compreender e diferenciar dois conceitos chave: falhas e exceções. Falhas são erros inesperados que surgem devido a fatores externos, como problemas de hardware ou interrupções elétricas, e não estão sob o controle direto da pessoa desenvolvedora. Exceções, por outro lado, são erros previsíveis e controláveis que ocorrem dentro da lógica do programa, como tentativas de acesso a valores nulos ou argumentos inválidos. Nesta aula, nosso objetivo é entender o que são falhas e exceções, e aprender na prática como tratá-las de forma eficiente e eficaz.

Conceitos Básicos de Falhas e Exceções

Falhas e exceções são conceitos fundamentais na programação, pois inevitavelmente, em algum momento, o seu programa apresentará erros. É crucial entender a diferença entre esses dois termos e como identificá-los em seu projeto.

Falhas referem-se a erros inesperados que ocorrem devido a fatores externos fora do controle direto da pessoa desenvolvedora, como problemas de hardware, alimentação elétrica ou outros fatores externos. Exemplos comuns incluem falhas de hardware, como um disco rígido corrompido, ou interrupções de energia. Quando ocorre uma falha, o sistema frequentemente retorna uma mensagem genérica de “erro inesperado”, pois não consegue determinar a causa exata do problema.

Exceções, por outro lado, são erros que ocorrem dentro da lógica do programa, sendo mais previsíveis e controláveis. Elas surgem de condições previstas ou imprevistas no código, como tentativas de acesso a um valor nulo ou argumentos inválidos. As exceções permitem que a pessoa desenvolvedora implemente ações específicas para lidar com esses erros, garantindo que o programa possa continuar funcionando de forma controlada.

Exemplos de Tratamento de Falhas e Exceções

Para tratar falhas e exceções de forma eficiente, utilizamos blocos de código específicos, como o `try`, `catch` e `finally`. No Java, por exemplo, o bloco `try` é usado para tentar executar um código que pode gerar uma falha ou exceção. Se ocorrer um erro, o bloco `catch` captura a exceção e permite que você trate o erro de maneira adequada, seja registrando um log, enviando uma mensagem de erro ou tomando outra ação necessária. O bloco `finally`, embora opcional, é executado independentemente de ter ocorrido uma falha ou não, sendo útil para liberar recursos ou executar ações finais.

Exemplo prático em Java:

```java
try {
    // Código que pode gerar uma exceção
    int result = 10 / 0;
} catch (ArithmeticException e) {
    // Tratamento da exceção
    System.out.println("Erro: Divisão por zero.");
} finally {
    // Bloco finally, executado independentemente
    System.out.println("Operação finalizada.");
}
```

Neste exemplo, a divisão por zero gera uma `ArithmeticException`, que é capturada e tratada no bloco `catch`, enquanto o bloco `finally` é executado sempre.

Adição de Tratamento de Exceção no Projeto

Para adicionar tratamento de exceções em um projeto, é necessário identificar os pontos do código onde as exceções podem ocorrer. Em um sistema que interage com um banco de dados, por exemplo, as exceções podem surgir em operações de busca ou atualização de dados. Uma vez identificados esses pontos, você deve determinar qual tipo de exceção pode ocorrer e como deseja tratá-la, seja interrompendo a execução, registrando um log ou retornando uma mensagem específica.

Exemplo de implementação em um serviço Java:

```java
public User findUserById(Long id) {
    try {
        return userRepository.findById(id).orElseThrow(() -> new
EntityNotFoundException("Usuário não encontrado"));
    } catch (EntityNotFoundException e) {
        log.error(e.getMessage());
        throw new CustomException("Erro ao buscar usuário", e);
    }
}
```

Neste exemplo, se o usuário não for encontrado, uma `EntityNotFoundException` é lançada e tratada, registrando o erro e lançando uma `CustomException` com uma mensagem personalizada.

Personalização de Exceções

Personalizar exceções permite que você crie mensagens de erro específicas e mais informativas para facilitar a manutenção e depuração do código. No Java, você pode criar suas próprias classes de exceção estendendo a classe `Exception` ou `RuntimeException`.

Exemplo de personalização:

```java
public class CustomException extends RuntimeException {
    private final int errorCode;

    public CustomException(String message, int errorCode) {
        super(message);
        this.errorCode = errorCode;
    }

    public int getErrorCode() {
        return errorCode;
    }
}
```

Ao lançar essa exceção em seu código, você pode fornecer um código de erro específico que ajuda a identificar e tratar o problema de forma mais eficiente.

Exemplo de uso:

```java
public void someMethod() {
    try {
        // Código que pode gerar uma exceção
    } catch (SomeSpecificException e) {
        throw new CustomException("Erro especifico ocorreu", 1001);
    }
}
```

Neste exemplo, a `CustomException` personaliza a mensagem de erro e inclui um código de erro específico, tornando o tratamento de erros mais detalhado e útil para a depuração.

Compreender e implementar o tratamento adequado de falhas e exceções é essencial para desenvolver aplicações robustas e resilientes.

GitHub da Disciplina

Você pode acessar o repositório da disciplina no GitHub a partir do seguinte link:

https://github.com/FaculdadeDescomplica/Framework. Esse espaço é o seu portal para mergulhar fundo no universo da aprendizagem interativa. Nele, você encontrará todos os códigos, além dos links para os arquivos e dados.

Conteúdo Bônus

Um excelente conteúdo bônus gratuito para alunos de graduação que estão aprendendo sobre Frameworks e Tratamento de Falhas e Exceções é a série de tutoriais e documentação da Oracle sobre Java Exception Handling. Este recurso é ideal porque fornece explicações detalhadas, exemplos práticos e é amplamente utilizado em muitos frameworks populares, como Spring Boot.

Título: Oracle Java Tutorials - Exception Handling

Plataforma: Docs.oracle

Este tutorial abrange desde os conceitos básicos até práticas avançadas de tratamento de exceções, incluindo:

Diferença entre exceções verificadas e não verificadas.

Uso dos blocos `try`, `catch` e `finally`.

Criação de exceções personalizadas.

Boas práticas para o tratamento de exceções em projetos reais.

A documentação é bem organizada, rica em exemplos e oferece uma base sólida para entender como lidar com falhas e exceções de maneira eficiente em aplicações Java.

Referência Bibliográfica

CARDOSO, L. da C. **Design de aplicativos**. Intersaberes: 2022

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7.ed. Pearson: 2018.

JOÃO, B. do N. **Usabilidade e interface homem-máquina**. Pearson: 2017

LEAL, G. C. L. **Linguagem, programação e banco de dados**: guia prático de aprendizagem. Intersaberes: 2015.

MEDEIROS, L. F. de. **Banco de dados**: princípios e prática. Intersaberes: 2013.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de dados**: implementação em SQL, PL/SQL e Oracle 11g. Pearson: 2013.

SETZER, V. W.; SILVA, F. S. C. **Bancos de dados**. Blucher: 2005.

VICCI, C. (Org.). **Banco de dados**. Pearson: 2014.

Ir para exercício