Universidade Federal Rural de Pernambuco
Bacharelado em Sistemas de Informação

Tratamento de exceção

Rodrigo Félix da Silva

1 Exceção

Durante a execução de um programa, qualquer evento que quebre o fluxo normal ou esperado de uma instrução é chamado de exceção.

1.1 Tratamento de exceções

Ao ser acionada a exceção, o melhor sistema deveria ter os meios necessários para que as ações a seguir sejam executadas:

1- Identificar a origem do erro;
2- Identificar que componente do sistema causou o erro, ou se o erro é cometido por um componente externo;
3- Identificado o componente causador, isolá-lo para evitar a continuidade da falha;
4- Identificar todas as partes afetadas pelo erro;
5- Recuperar o sistema dos danos causados pelo erro, corrigindo o estado interno do sistema para eliminar o erro;
6- Reconfigurar o sistema para que ele possa retomar suas rotinas sem passar pelo componente causador do erro;
7- Reiniciar o processamento normal do sistema.

1.2 Componente ideal com tolerância a falhas

O desenvolvimento de sistemas utilizando arquitetura em camadas é visto com bons olhos para a implementação de mecanismos com tolerância a falhas, visto que essa arquitetura permite um particionamento do modelo de falhas do sistema como um todo em diversos modelos menores, que serão mais fáceis de serem tratados. A facilidade se deve ao fato que nesta estrutura cada camada só tratará as falhas derivadas da camada imediatamente inferior.

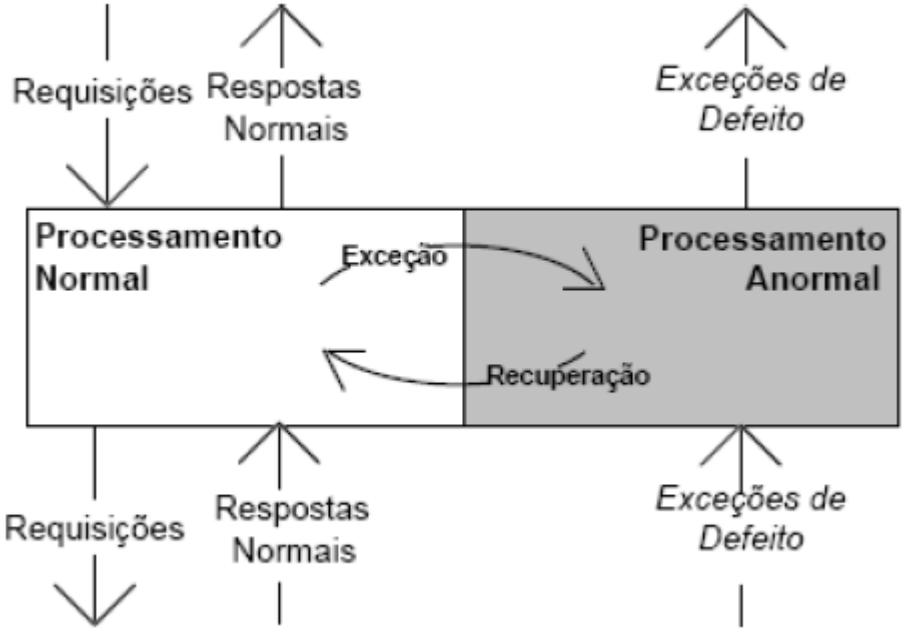
Exemplo: Em o sistema que utiliza banco de dados, a camada mais inferior deveria ter o tratamento de falhas como gravação e leitura, e consequentemente o modo que o sistema irá se recuperar caso ocorra essas falhas. Assim, as camadas mais altas, como a interface com o usuário, não necessitam ter esse tipo de falha no seu modelo de falhas.

2 Mecanismos de tratamento de exceções

As linguagens de programação em sua maioria não tem um sistema conveniente para tratamento de exceções, apenas interrompe o processamento ao detectar o erro. Essas interrupções podem trazer grandes custos, como impedir a realização de rotinas necessárias para o negócio do cliente, alto valor gasto em suporte técnico, diagnóstico de falhas, recuperação do estado antes da interrupção, dentre outros.

Em diversos sistemas, como controle de tráfego aéreo, interrupções são intoleráveis. Esses sistemas obrigam que as linguagens de programação utilizadas permitam o tratamento

das exceções de forma automática pelo próprio sistema, a fim de sanar quaisquer problemas que venham a acontecer e garantindo o funcionamento em tempo integral.



Componente ideal tolerante a falhas

A linguagem C é uma linguagem antiga que oferece um bom suporte para tratamento de exceção e por esse motivo é uma das mais utilizadas em sistemas com altos requisitos de tolerância a falhas. Pela grande importância desse mecanismo, as linguagens modernas, como Java, adotaram essa tendência.

2.1 Representação das exceções

Um mecanismo de tratamento de exceção tem como principal tarefa tornar simples a codificação, separando rotinas de tratamento das exceções das rotinas de processamento normal. Dessa forma, sempre que uma condição excepcional for detectada, automaticamente deve ser codificada e criada uma representação que possa se tratada em outro ponto do sistema. As duas formas para isso ser feito são:

(i) Utilizar símbolos que identificam o tipo de exceção, como OVERFLOW ou IO_ERROR;

(ii) Utilizar um modo mais especificado de representação para cada tipo de exceção, como mensagens ou objetos com atributos que descrevem a exceção.

2.2 Assinaturas de métodos

Nas linguagens orientadas a objetos, há divergências sobre os tipos das exceções que podem ser recebidas como consequência da chamada de um método e se devem fazer parte da assinatura do método. Há quatro alternativas para essa questão:

(i) Não incluir na assinatura do método os tipos de suas exceções externas;
(ii) Permitir, por opção, que os tipos de exceções externas sejam incluídos na assinatura do método;
(iii) Incluir, obrigatoriamente, todos os tipos de exceções externas na assinatura do método;
(iv) Incluir, obrigatoriamente, apenas alguns tipos de exceções externas e deixar opcional a inclusão de outros tipos na assinatura do método.

A inclusão é obrigatória para assegurar o tratamento das exceções de maneira clara pelo sistema, eliminando a possibilidade de ocorrer exceções que não tenha um tratamento adequado previsto. Porém essa obrigatoriedade faz crescer o número de rotinas para o tratamento de exceção.

3 Contexto dos tratadores de exceções

Em mecanismos de tratamento de exceções que não foram totalmente desenvolvidos, como o do MS-DOS, que é baseado em interrupções, é permitido que seja declarada apenas uma rotina de tratamento para cada tipo de exceção, independentemente do contexto. Por exemplo, qualquer exceção identificada como ARITHMETIC_OVERFLOW será tratada pela mesma rotina, que deve ser capaz de recuperar o contexto onde a exceção ocorreu e decidir pelo procedimento a ser adotado naquele caso.

Mecanismos mais avançados permitem que uma mesma exceção possa ter vários tratadores, que estarão cada um associado a um tipo de contexto de execução. Esse contexto pode ser uma classe ou apenas um comando da linguagem, onde vai depender do mecanismo e da linguagem orientada a objetos que foi adotada.

3.1 Propagação de exceções

Uma exceção deve ser tratada em um ponto do sistema que seja o mais próximo de onde ela foi criada, quando possível. É ideal que o código tratador esteja dentro do mesmo método que criou a exceção, assim é evitado que uma resposta anormal seja recebida por quem o chamou. Quando não for possível, o ideal é que a exceção seja tratada imediatamente por quem chamou o método.

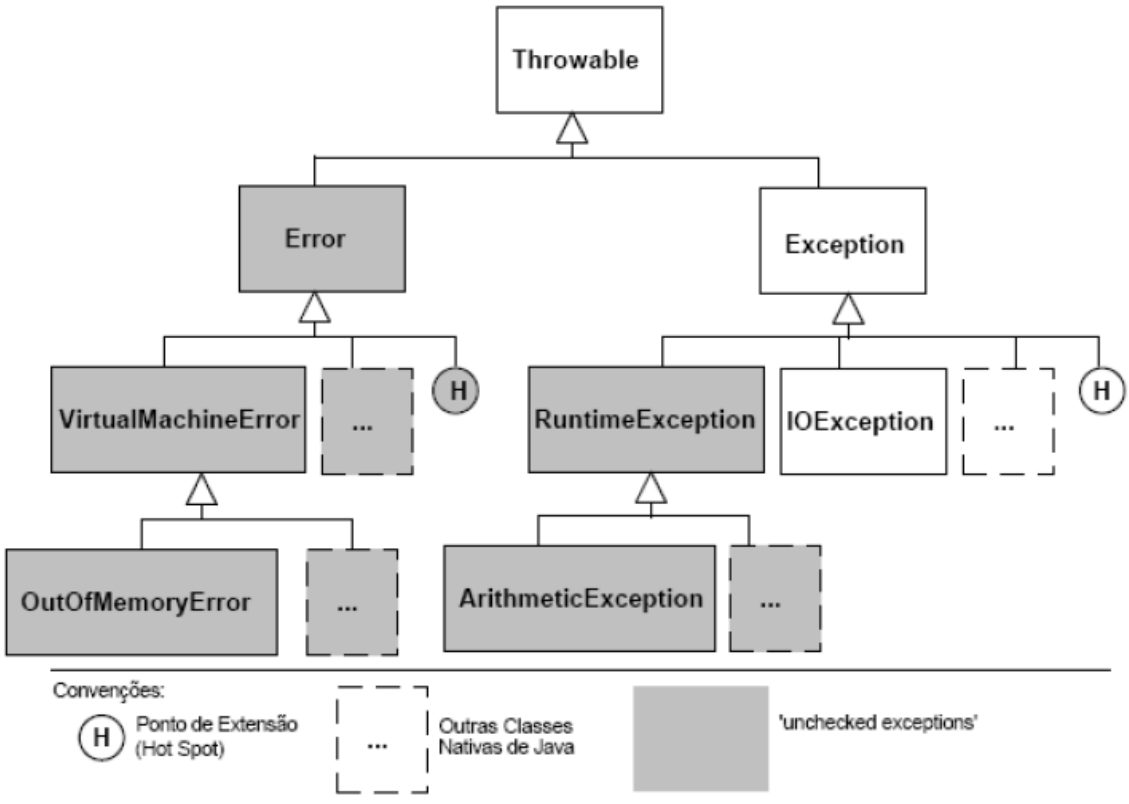
3.2 Continuação do fluxo de controle

Os mecanismos de tratamento de exceções tem com um dos aspectos mais importantes o modo como a sequencia de execução do programa será afetada depois do lançamento de uma exceção. Existem dois modelos: o modelo de continuação (resumption) e o de terminação (termination). Nos dois, quando uma exceção é lançada, o fluxo de execução é desviado para um tratador apropriado. A diferença entre os dois é a sequencia do fluxo após o término do tratador.

No modelo de continuação (resumption), a execução prossegue a partir do comando seguinte ao que provocou a exceção, ou seja, é mantido o fluxo normal de execução.

No modelo de terminação (termination), a execução prossegue a partir do final do bloco que contém o comando que provocou a exceção, ou seja, o fluxo de execução normal é interrompido no ponto em que ocorreu a exceção e reiniciado a partir do próximo bloco.

O modelo de terminação é mais próximo da realidade, pois dificilmente o fluxo normal pode ser mantido sem alterações após uma exceção.



Tratamento de exceções em Java. Hierarquia da classe Throwable.