Universidade Federal do Ceará
Centro de Ciências/Departamento de Computação
Código da Disciplina: CK0236
Professor: Ismayle de Sousa Santos

Aula 17

Técnica de Programação II

Programação Defensiva

 qpg4p5x  ismaylesantos@great.ufc.br  @IsmayleSantos

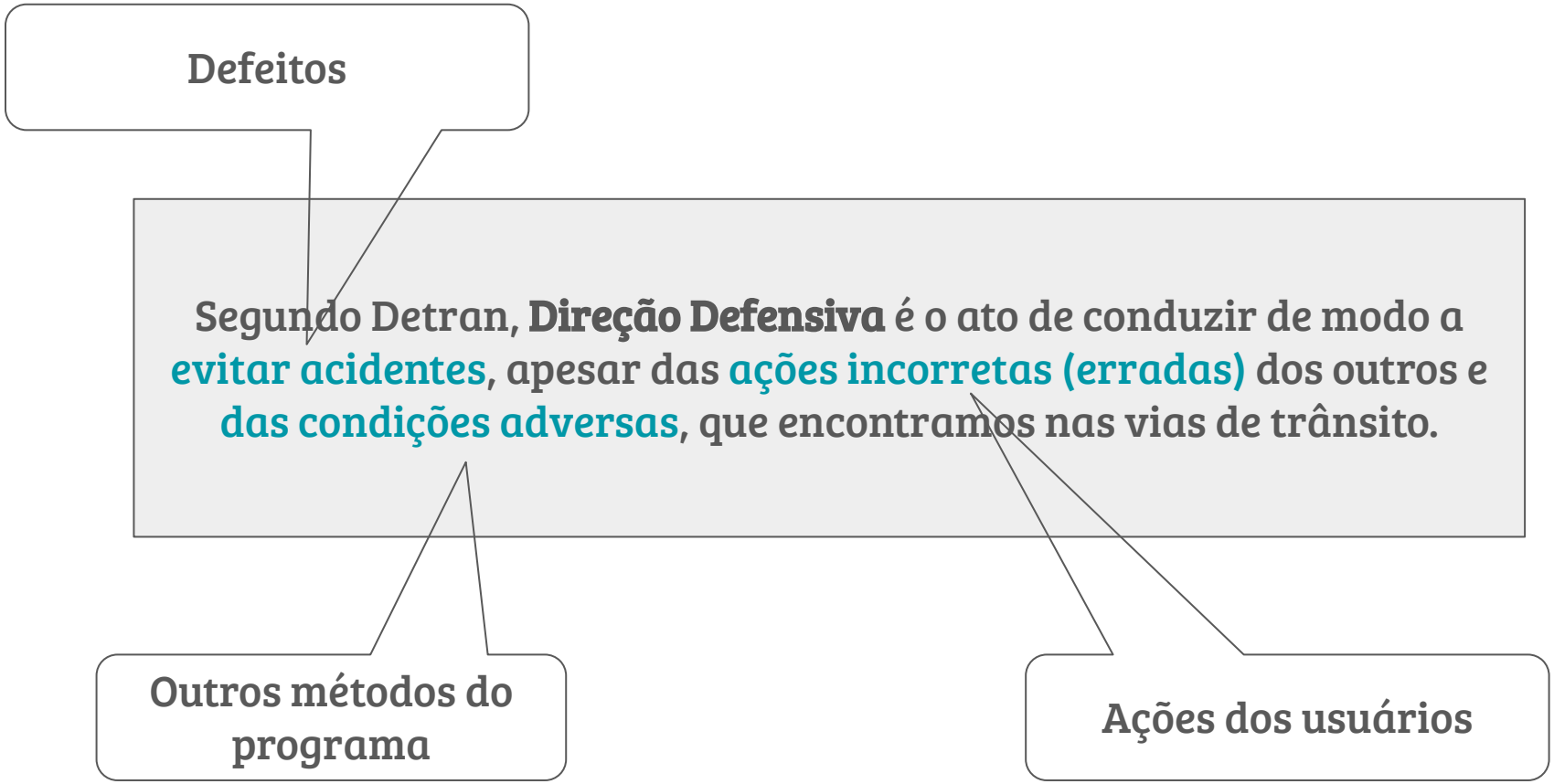
Programação Defensiva

- Baseado na ideia de Direção Defensiva

> Segundo Detran, **Direção Defensiva** é o ato de conduzir de modo a evitar acidentes, apesar das ações incorretas (erradas) dos outros e das condições adversas, que encontramos nas vias de trânsito.

Programação Defensiva

- Pensando agora em desenvolvimento de software



Programação Defensiva

- Refere-se ao uso de técnicas para garantir o funcionamento contínuo de um software, mesmo diante de
  - dados inválidos
  - eventos que nunca acontecem
  - erros de outros programadores
  - etc...

Programação Defensiva

- Para melhorar a qualidade do software, o ideal é combinar programação defensiva com outra técnicas
  - Desenvolvimento orientado a testes
  - Inspeções de Design
  - Uso de princípios de projeto de software
  - etc ..

Programação Defensiva

- Estratégias
  - Validação das entradas
  - Asserções
  - Tratamento de Erros
  - Tratamento de Exceções
  - Barricadas

Validação das Entradas

- Entradas
  - Diferentes Fontes
    - Arquivo
    - Usuário
    - Outros Sistemas
  - Sem controle, inesperadas, Imprevisíveis
  - Podem ser mal-intencionadas
    - E.g.: Exploração de vulnerabilidades

Validação das Entradas

- Para lidar com entradas ruins/inválidas
  - Verificar os valores de todos os dados de fontes externas
    - Exemplo: verificar se valores numéricos de um arquivo ou entrada de usuário estão dentro dos limites esperados
  - Verificar os valores de todos os parâmetros de entradas de métodos
    - Verificar parâmetros trocados entre métodos, de modo similar a verificação de dados de fontes externas ao código
  - Decidir como lidar com entradas ruins
    - Uma vez detectado um parâmetro inválido, o que fazer?

Validação das Entradas

- Exemplo
  - O que poderia acontecer? Como resolver?

```
public static int fatorial(int num) {
    if(num == 0)
        return 1;
    else
            return num * fatorial(num-1);
}
```

Validação das Entradas

- Exemplo
  - O que poderia acontecer? Como resolver?

Se num = -2

```
public static int fatorial(int num) {
    if(num == 0)
        return 1;
    else
            return num * fatorial(num-1);
}
```

Validação das Entradas

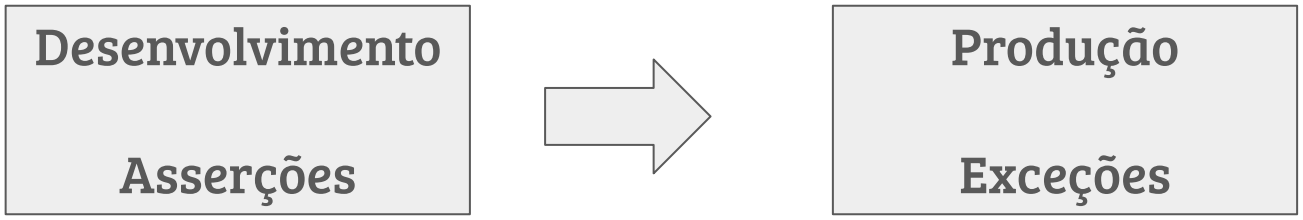
- Exemplo de verificações
  - O valor é numérico? É uma String?
  - Aceita valores negativos? Só maior que algum valor X?
  - Número de casas decimais importa?
  - Quantidade é válida?
  - Index do array é válido?

Asserções

- É um código durante desenvolvimento que é usado pelo software para verificar determinada condição durante sua execução
  - Quando o resultado da asserção é
    - true, tudo está ok
    - false, um erro inesperado foi detectado

Asserções

- São usadas durante desenvolvimento e manutenção do software
  - São removidas quando o código vai para produção
    - Asserções em produção podem comprometer o desempenho do sistema
  - Podem ajudar a reduzir o tempo de depuração



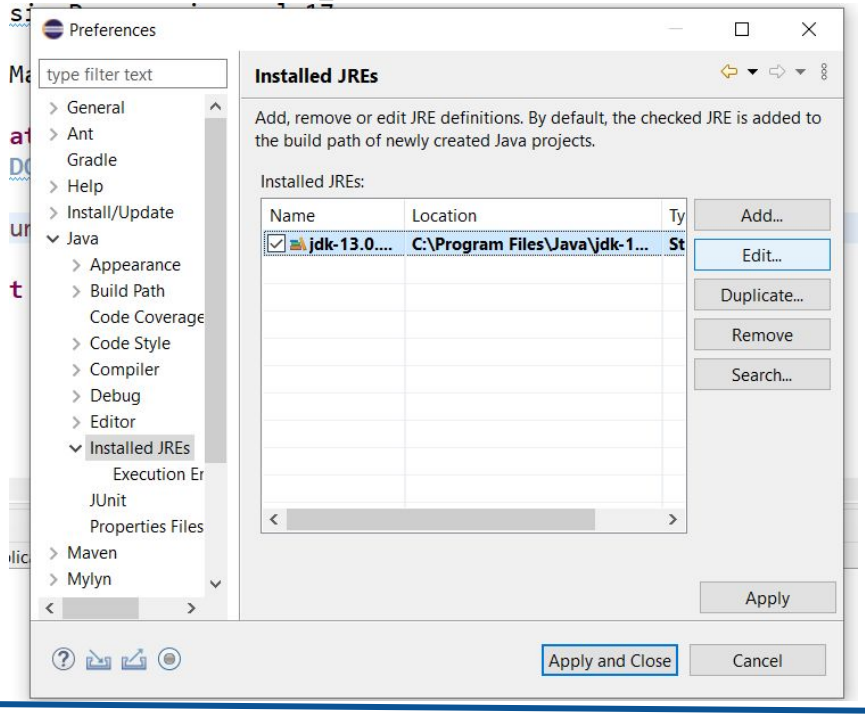
Asserções

- Argumentos
  - Expressão Booleana
    - Condição que deveria ser verdadeira
  - Mensagem
    - Exibida caso a condição não seja verdadeira

```
assert(velocidadeParticula < VELOCIDADE_LUZ):
"Velocidade da particula não pode ser maior que a velocidade da luz";
```

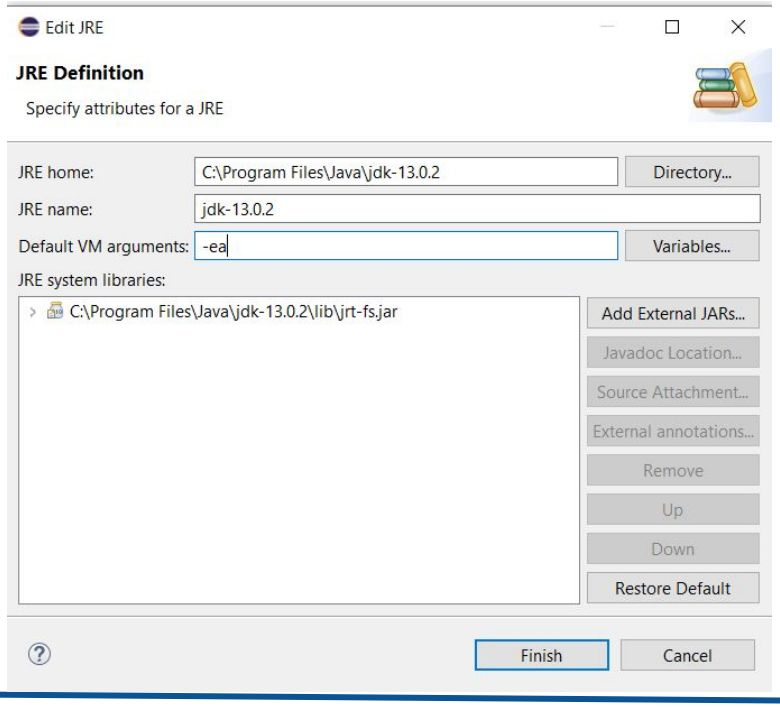
Asserções

- Exemplo no Eclipse
  - Configuração
    - Vá em Windows -> Preferences -> Java -> Installed JREs
      - Selecione a JRE que você está usando



Asserções

- Exemplo no Eclipse
  - Configuração
    - Na JRE, clique em editar e digite “-ea” (i.e, Enable Assertions) no campo Default VM arguments



Asserções

- Exemplo no Eclipse

```
3  public class Main {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7
8          int num = -1;
9
10         assert num > 0 : "Deveria ser maior que 0";
11
12     }
```

Console

<terminated> Main (3) [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (30 de jan de 2021 10:51:54 – 10:51:55)

```
Exception in thread "main" java.lang.AssertionError: Deveria ser maior que 0
        at defensiveProgramming.aula17.Main.main(Main.java:10)
```

Asserções

- Podem ser usadas para verificar condições como
  - Valores de parâmetro de entrada atendem determinado range
  - Arquivo é aberto(fechado) quando o método inicia (encerra)
  - Arquivo está aberto para somente leitura, leitura escrita ou ambos
  - Valor de uma variável de entrada não é alterado no método
  - Ponteiro não é nulo
  - etc

Guidelines para uso de Asserções

- Use Tratamento de Exceções para condições que são esperadas de acontecer e Asserções para condições que nunca podem acontecer
  - Tratamento de Exceção verifica entradas ruins
    - Mesmo com códigos corretos podemos ter uma exceção (e.g: erro de comunicação)
  - Asserções verificam erros no código
    - Se uma asserção é disparada, deve ser feita a correção no código do software

Guidelines para uso de Asserções

- Evite colocar código executável em Asserções
  - O problema é que o código pode não será executado quando você desabilitar as asserções
  - Coloque os resultados dos métodos em variáveis e faça assertions nas variáveis

```
1 | assert names.remove(null);
```

```
1 | boolean nullsRemoved = names.remove(null);
2 | assert nullsRemoved;  // Runs whether or not asserts are enabled
```

Fonte: https://www.devmedia.com.br/assertions-em-java/28781

Guidelines para uso de Asserções

- Use Asserções para documentar e verificar Pré e Pós Condições
  - Quando pré condições e pós condições são usadas, cada método forma um contrato com o resto do programa
  - Pré-condição
    - Propriedades que o código do cliente do método (ou classe) prometem serem verdadeiras antes de chamar a rotina (ou classe)
  - Pós-condição
    - Propriedades que o método (ou classe) promete serem verdadeiras quando a execução é finalizada

Guidelines para uso de Asserções

- Para sistemas complexos, pode-se usar asserções e tratamentos de exceção para o mesmo erro
  - Em sistemas complexo (e.g., Microsoft Word) nem todo defeito é encontrado e corrigido antes da liberação de versões
  - Asserções ajudam a encontrar os defeitos durante o desenvolvimento
  - Tratamento de Exceção 'tratam' o defeito na versão de produção

Tratamento de Erros

- É o processo de capturar erros que podem ocorrer durante a execução do software e tomar alguma ação

Tratamento de Erros

- Existem várias maneiras de lidar com erros
  - Retornar um valor neutro
  - Substituir pelo próximo dado válido
  - Substituir pelo valor válido mais próximo
  - Salvar o log em um arquivo
  - Retornar um código de erro
  - Exibir mensagem de erro
  - Shut Down

Tratamento de Erros

- Retornar um valor neutro
  - Para alguns sistemas é possível continuar operando e simplesmentes retornar um valor que conhecidamente não causa danos
    - Ex.:
      - Uma computação numérica retornando 0
      - Uma computação com String retornando uma string vazia

Tratamento de Erros

- Substituir pelo próximo dado válido
  - Ex.: Leitura de registros de uma base após encontrar um registro inválido pode simplesmente continuar a leitura até encontrar um registro válido
- Substituir pelo valor válido mais próximo
  - Ex.: Termômetro calibrado entre 0 e 100 ºC
    - Se a leitura for menor do que 0, pode substituir por 0 ºC
    - Se a leitura for maior que 100, pode substituir por 100 ºC

Tratamento de Erros

- Salvar uma mensagem de alerta em um arquivo
  - Quando um dado inválido for detectado você pode escolher por salvar a mensagem de alerta (log) em um arquivo e continuar
  - Pode ser usada com outras estratégias
    - E.g.: Junto com substituir pelo valor válido mais próximo

Tratamento de Erros

- Retornar um código de erro
  - Você pode decidir que algumas partes do sistema não vão tratar os erros localmente
    - Eles vão reportar os erros para outros métodos superiores na hierarquia de chamadas de métodos, os quais vão tratar o erro
  - Algumas formas de informar o erro
    - Alterar o valor de uma variável de status
    - Retornar o status como o valor de retorno da função
    - Disparar uma exceção

Tratamento de Erros

- Exibe uma mensagem de erro
  - Alerta o usuário onde está o erro
  - Minimiza o overhead do tratamento de erros
  - Implicações
    - Espalhamento de mensagens na interface de usuário em toda a aplicação
      - Pode dificultar a criação de UI consistentes
    - Pode fornecer informações demais para possíveis ataques à segurança do sistema

Tratamento de Erros

- Shut Down
  - Para aplicações críticas, a escolha pode ser shut down o software toda vez que detectar um erro
  - Ex.: Aplicação de controle de radiação para tratamento de câncer

Robustez vs Corretude

- Processamento de erros implica em mais corretude ou robustez
  - Corretude
    - Nunca retornar um valor incorreto
    - Retornar nada é melhor do que retornar algo impreciso
  - Robustez
    - Sempre tentar fazer algo para o software continuar operando, mesmo levando a resultados imprecisos
- A escolha do tratamento do erro depende do tipo de software
  - O tratamento para um video game é diferente do tratamento de erro de um software de controle de raio-X

Exceções

- O que é?
  - Algum evento inesperado que acontece durante a execução do programa
- É o meio pelo qual um código pode repassar erros ou eventos excepcionais para o código que chamou ele
- Tratamento de Exceções
  - Mecanismo para identificar e tratar uma exceção

Exceções

- O que pode gerar uma exceção?
  - Tentar de abrir um arquivo inexistente ou corrompido
  - Timeout de comunicação com outro software
  - Tentar acessar posição inválida de um vetor

Note que nem sempre o que causa uma exceção é um problema no código-fonte

Tratamento de Exceções

- Estrutura Básica
  - Um método usa throw para lançar a exceção
  - Código em outro método acima na hierarquia de chamadas de métodos captura a exceção com um bloco try-catch
    - try
      - bloco onde pode ocorrer a exceção
    - catch
      - bloco responsável para tratar a exceção

Tratamento de Exceções

- Exemplo try-catch

```
public void chamaMetodosComExceção() {
    try {
        // Chamando um método que dispara uma exceção
        metodoComExceção();
    } catch (FileNotFoundException e) {
        // Tratamento da Exceção
        System.out.println("Arquivo não encontrado");
    }
}

public void metodoComExceção () throws FileNotFoundException {
    //Exemplo de método que pode retornar uma Exceção durante execução
    BufferedReader reader = new BufferedReader(new FileReader("C://local//naoexiste"));
}
```

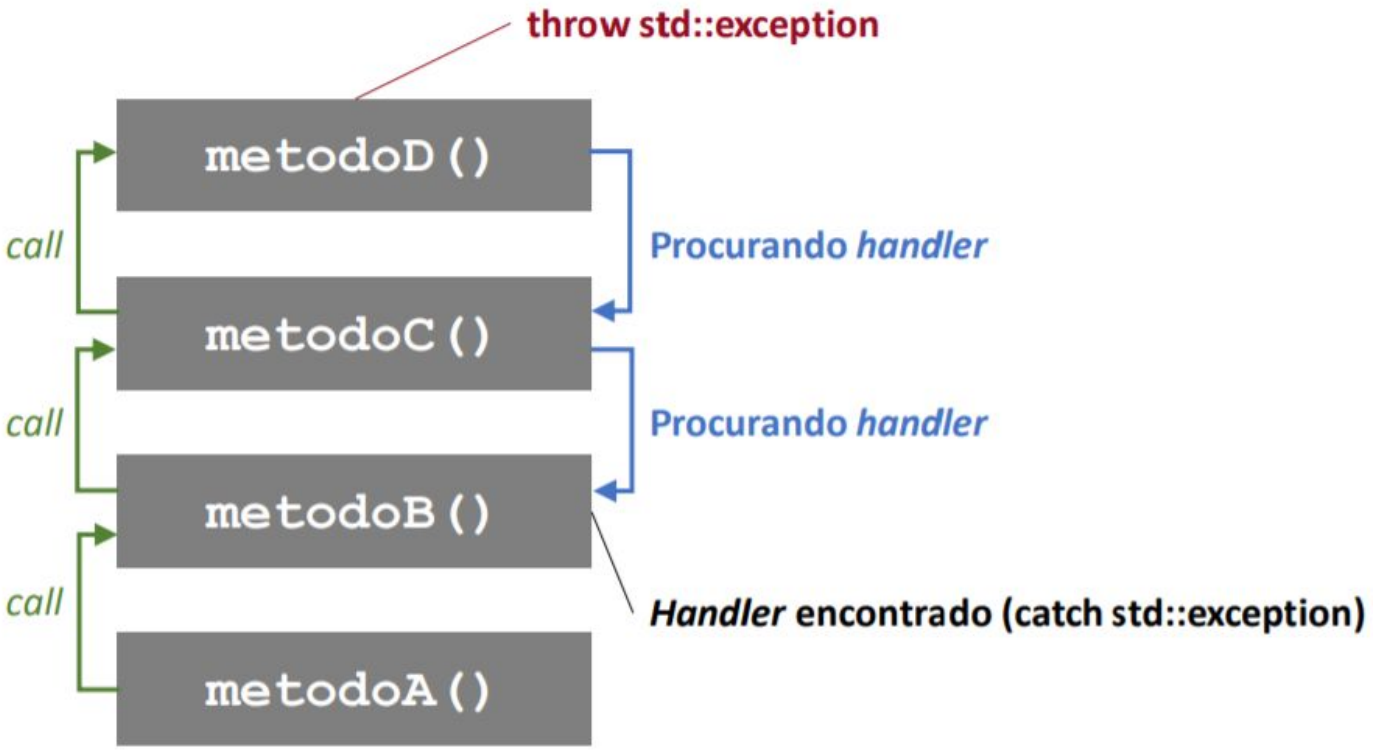
Tratamento de Exceções

- Exemplo try-catch-finally

```
//bloco try-catch-finally
try {
    metodoComExceçãoNullPoint();
}catch (NullPointerException e) {
    // Tratamento da Exceção
    System.out.println("Exceção foi tratada");
} finally {
    System.out.println("Sempre é executado");
}
```

Tratamento de Exceções

- Pilha de Exceção



Fonte: https://www.devmedia.com.br/assertions-em-java/28781

Tratamento de Exceções

- O que acontece se a exceção não é capturada ?

```
public void exemploComExceçãoNãoCapturada() {
    int[] vetor = new int[4];

    try {
        int i = vetor[4]; // vai disparar exceção

        System.out.println("Comandos dentro do bloco try");
    }catch (NullPointerException e) {
        System.out.println("Exceção capturada");
    }

    System.out.println("Resto do Programa");
}
```

Tratamento de Exceções

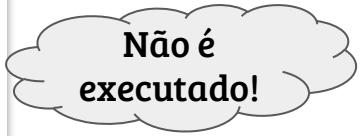
- O que acontece se a exceção não é capturada ?

```
public void exemploComExceçãoNãoCapturada() {
    int[] vetor = new int[4];

    try {
        int i = vetor[4]; // vai disparar exceção

        System.out.println("Comandos dentro do bloco try");
    }catch (NullPointerException e) {
        System.out.println("Exceção capturada");
    }

    System.out.println("Resto do Programa");
}
```




```
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4
        at defensiveProgramming.aula17.ExemplosExceções.exemploComExceçãoNãoCapturada(ExemplosExceções.java:76)
        at defensiveProgramming.aula17.ExemplosExceções.chamaMetodosComExceção(ExemplosExceções.java:38)
        at defensiveProgramming.aula17.Main.main(Main.java:20)
```

Dicas para usar Exceções

- Use exceções para notificar outras partes do software sobre erros que não deveriam ser ignorados
  - O benefício da exceção é sinalizar uma condição de erro de forma que ele não pode ser ignorado
  - Em outras abordagens de tratamento de erro, o erro pode passar despercebido pelo código base
    - E.g: nas técnicas de substituições a entrada inválida é substituída por uma válida

Dicas para usar Exceções

- Lance exceções somente para condições que são verdadeiramente excepcionais
  - Exceções são usadas para eventos não frequentes e que nunca deveriam acontecer

Dicas para usar Exceções

- Não use exceções para “passar a bola”
  - Prefira tratar os erros localmente, se possível
    - Sem lançar exceção para métodos superiores na hierarquia de chamadas
    - Facilita a manutenção

Dicas para usar Exceções

- Inclua na mensagem da exceção todas as informações que levaram a exceção
  - A mensagem tem que ter a informação necessária para o entendimento do porque a exceção foi lançada
  - Ec.: Se a exceção foi devido a erro de indexação de um array , inclua na mensagem os valores inferior e superior do array, além do valor do index inválido

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4
        at defensiveProgramming.aula17.ExemplosExceções.exemploComExceçãoNãoCapturada(ExemplosExceções.java:76)
        at defensiveProgramming.aula17.ExemplosExceções.chamaMetodosComExceção(ExemplosExceções.java:38)
        at defensiveProgramming.aula17.Main.main(Main.java:20)
```

Dicas para usar Exceções

- Evite blocos de catch vazios
  - Se o bloco catch está vazio
    - O bloco try está errado, levantando uma exceção sem motivo OU
    - O bloco catch está errado por não tratar a exceção
  - Se for o caso de não tratar o erro, pelo menos registre o log em algum arquivo

```
//Sugerido - tratar a exceção no bloco catch ou pelo menos registrar no log
try {
    //Código
    throw new Exception();
}catch (Exception e) {
    //Salvar no arquivo de Log a exceção
    Logger logger = Logger.getAnonymousLogger();
    logger.log(Level.SEVERE, "execeção grave");
```

Dicas para usar Exceções

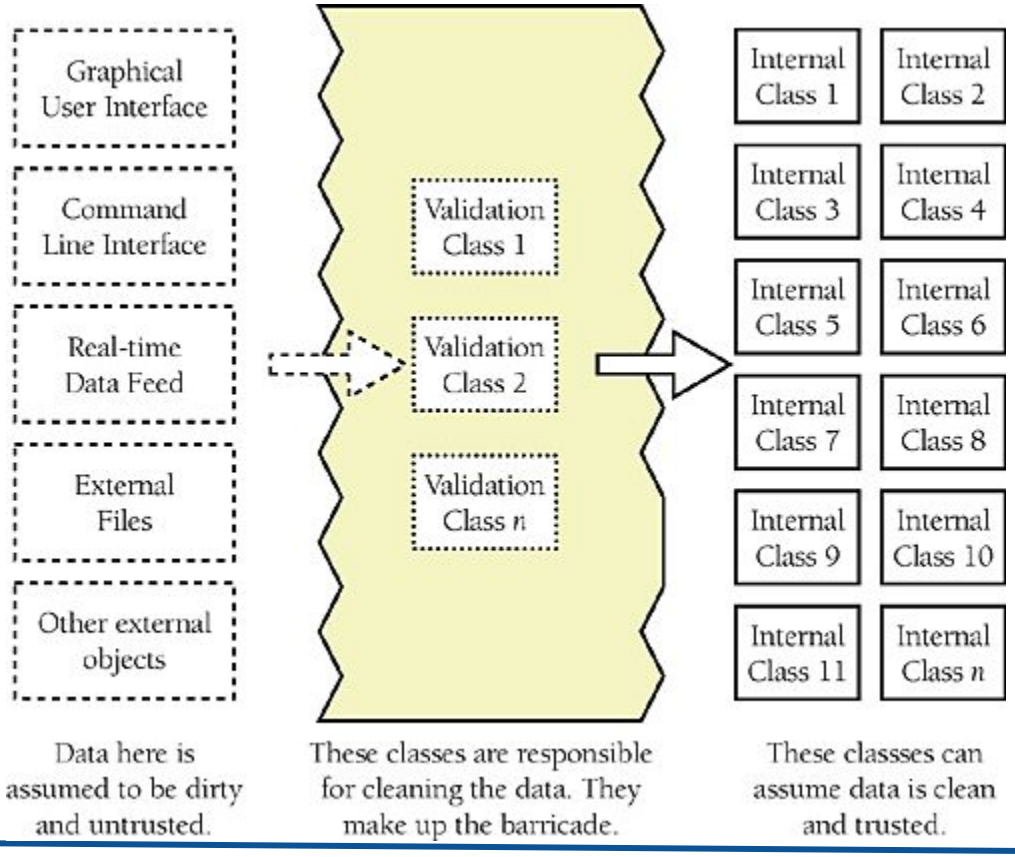
- Conheça as exceções que seu código e bibliotecas usadas estão lançando
  - Falhar em tratar exceções lançadas pelo seu código e bibliotecas usadas pode resultar em crash da sua aplicação

Dicas para usar Exceções

- Considere desenvolver um tratamento de exceção centralizado
  - Fornece um repositório central
    - Para os tipos de exceção tratados
    - Como cada exceção deve ser tratada
    - Formatação das mensagens de exceção
  - Desvantagem
    - Acoplamento do software com a classe que faz o tratamento global das exceções

Barricadas

- Estratégias de contenção de danos
- Uma opção é definir interfaces como limites para “áreas seguras”



Barricadas

- Um método público que assume que o dado que ela recebe pode não ser “seguro”, pode verificar e tratar esses dados
  - Converte os tipos de dados de entrada nos formatos apropriados
  - Assim, os métodos privados podem assumir que esses dados estão seguros
- Asserções vs Exceções
  - Considere usar
    - Exceções para os métodos públicos
    - Asserções para métodos privados

Programação Defensiva no Código em Produção

- Código em produção deve lidar com os erros de maneira mais sofisticada
  - E.g.: Mensagens de erro devem ser apropriadas para o usuário final
- Registre as falhas identificadas
  - Log de erros
- Código em desenvolvimento têm menos restrições do que código em produção
  - Muita programação defensiva pode comprometer o desempenho do sistema
    - Pense onde você precisa ser defensivo

Trabalho Prático - TP8

- Prática de Programação Defensiva
  - Utilizar pelo menos 2 técnicas de Programação Defensiva
  - Pode ser no projeto final ou mesmo usando outro código
  - Deadline: 08/02/2021

Obrigado!

Por hoje é só pessoal...

Dúvidas?

qpg4p5x

ismaylesantos@great.ufc.br

@IsmayleSantos