Introdução à Programação

Programação e Fatores de Qualidade

Qualidade de Software

Quero que você escreva rapidamente um sistema de folha de pagamento que funcione corretamente, que seja robusto e que seja rápido. Ah! Pretendo lançar uma outra versão deste mesmo programa, daqui a uns 6 meses.

Sem problema, chefe.

Ferrou!

Tópicos da Aula

- Hoje aprenderemos alguns fatores de qualidade de software
  - Fatores de qualidade externos
  - Fatores de qualidade internos
- Depois aprenderemos algumas técnicas para melhorar alguns fatores de qualidade de um programa
  - Comentários
  - Refactoring
  - Especialização com subclasses
- Finalmente, iremos mostrar como alguns erros dinâmicos podem ser depurados com comentários
  - Erros dinâmicos
  - Depuração com comentários

Impacto econômico e social do software de qualidade

- Disponibilidade de serviços essenciais
  - *home banking*
  - telefonia
- Segurança de pessoas
  - sistemas de monitoramento de pacientes
  - sistemas de controle de tráfego aéreo
  - freios ABS

Slide 4

AS1 Adriano Sarmento; 4/3/2008

O impacto na prática

- Competitividade das empresas
- Melhores produtos a um menor custo
- Atração de novas empresas para a região
  - Investimentos na região
  - Arrecadação de impostos

Crise de Software

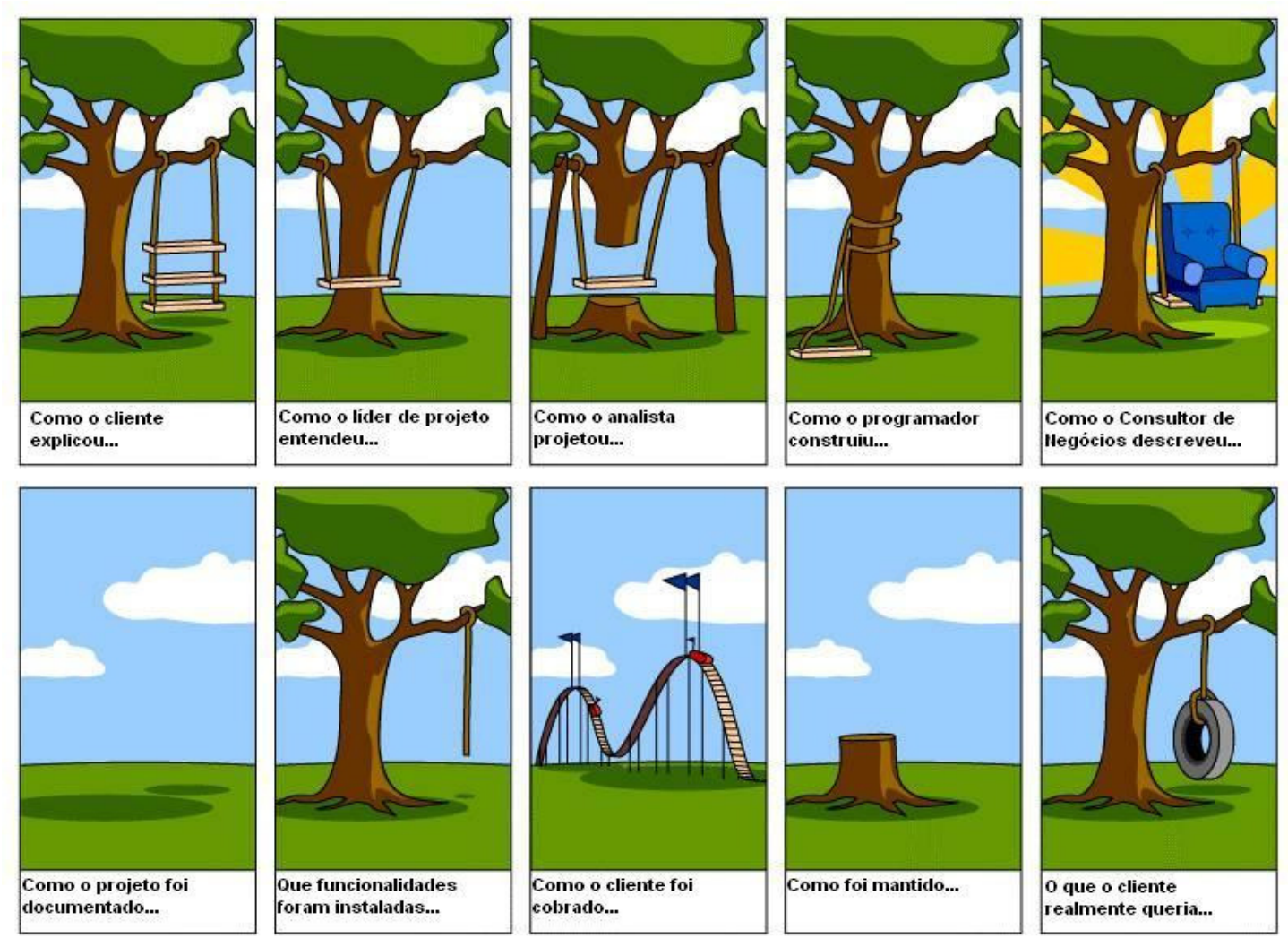
- Nas 3 primeiras décadas da Computação, a preocupação maior era hardware
  - Custo de processamento e armazenamento de dados
- Desafio atual é melhorar qualidade e custo do software
- Problemas
  - 25% dos projetos são cancelados
  - O tempo e custo de desenvolvimento é bem maior do que o estimado
    - em 53% dos projetos
  - 75% dos sistemas não funcionam como planejado
  - A manutenção e reutilização são difíceis e custosas
  - A produtividade dos profissionais da área de software não tem acompanhado a demanda por seus serviços

Causas da Crise de Software

- Essências
  - Complexidade crescente dos sistemas
  - Dificuldade e custos de formalização
    - Uma linha de código do sistema de controle de lançamento do ônibus espacial da NASA custa 1.000 dólares!

Essências não podem ser evitadas

Dificuldade de Formalização



Causas da Crise de Software

- Acidentes
  - Má qualidade das linguagens, ferramentas e metodologias
  - Problemas gerenciais (pessoas, riscos, etc.), políticos, e organizacionais
  - Pouca comunicação entre o cliente e o desenvolvedor
  - Manutenção de software, geralmente não é considerada como parte do ciclo de software

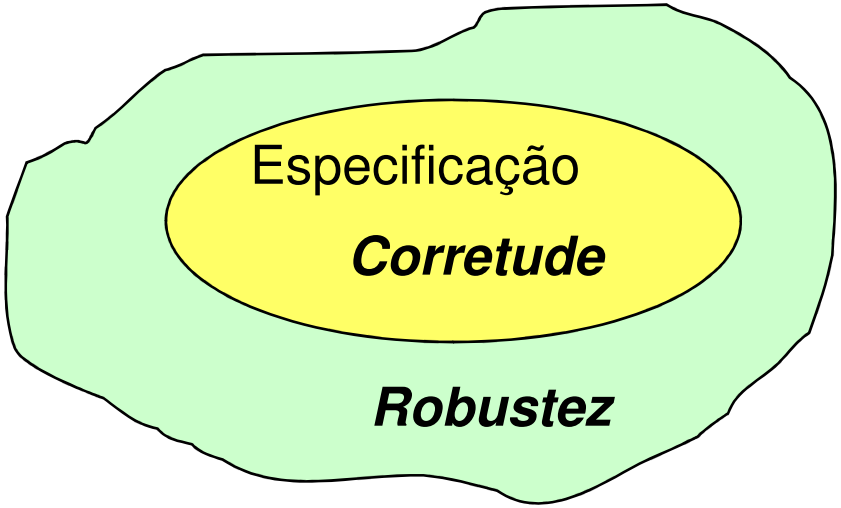
Acidentes PODEM ser evitados!!!

Fatores de Qualidade de Software

- Qualidade de software é uma combinação de vários fatores
- Estes fatores podem ser classificados em:
  - Fatores Externos
    - Qualidades percebidas pelos usuários de um software
    - Usuários incluem: usuário final e aquela pessoa que contratou o desenvolvimento do software
    - Ex: corretude, eficiência, reusabilidade, etc
  - Fatores Internos
    - Qualidades percebidas somente por profissionais de software que têm acesso ao código do software
    - Ex: modularidade, legibilidade, etc

Alguns Fatores Externos

- **Corretude** é a capacidade do software executar de forma correta todas as tarefas que foram pedidas nos requisitos e especificação do software
- **Robustez** é a capacidade do software funcionar, mesmo em situações não previstas na especificação
  - É o complemento de corretude



Alguns Fatores Externos

- **Extensibilidade** é a facilidade em que o software pode ser alterado para que atenda novas exigências da especificação
  - Software está em constante mudança
  - ≈70% de custo do software é a fase de manutenção
  - Mudanças de requisitos são responsáveis por ≈ 50% das atividades de manutenção

- **Reusabilidade** é a capacidade que o software tem de ser usado em novas aplicações.
  - Este uso pode ser de partes do software ou dele todo
  - Grande impacto no tempo de desenvolvimento de software
  - Menor tempo de desenvolvimento é importante para sucesso do software

Alguns Fatores Externos

- **Eficiência** é a capacidade do software de otimizar a utilização dos recursos de hardware
  - Muitas vezes requer um bom conhecimento do hardware onde o software será executado
  - Pode tornar a implementação do software dependente demais da plataforma alvo
- **Facilidade de Uso** é uma qualidade que diz respeito a facilidade com que usuários com diferentes perfis conseguem aprender a utilizar o software e resolver os problemas desejados
  - Engloba também o aprendizado de instalação, operação e monitoramento
  - Representa economia para a empresa que utiliza o software em relação aos custos de treinamento

Outros Fatores Externos

- Portabilidade
- Escalabilidade
- Integridade e segurança
- Compatibilidade
- Flexibilidade
- Testabilidade

Alguns Fatores Internos

- Um software é **modular** se ele é construído à partir de elementos autônomos conectados
  - Modularidade ➔ Reusabilidade e Extensibilidade

- **Legibilidade** de um programa é a facilidade com que profissionais de software conseguem entender o código do programa
  - Documentação é importante
  - Padrões de codificação devem ser seguidos
  - Legibilidade ➔ Reusabilidade e Extensibilidade

Algumas Considerações sobre Qualidade

- Fatores internos ➔Fatores externos
- Softwares são utilizados em aplicações dos mais diversos domínios
- Freqüentemente, não se pode obter todos os fatores de qualidade
  - Diferentes fatores podem ser conflitantes
    - Ex: Freqüentemente para aumentar a eficiência, código do programa deve ser mais específico a uma plataforma, o que diminui a reusabilidade do código

Algumas Considerações sobre Qualidade

Cabe ao desenvolvedor avaliar quais fatores são mais importantes de acordo com a aplicação

Comentando um Programa

- Um programa deve ser bem documentado
  - Documentação externa para usuários
  - Documentação no próprio programa
    - Aumenta a legibilidade

Comentário é um mecanismo oferecido pelas linguagens de programação que permite que o programador expresse em linguagem natural a lógica pensada para escrever um programa ou um trecho de programa

Comentando um Programa

- Usa-se o termo *inline documentation* para comentários em um programa
- Devem ser incluídos para explicar o propósito do programa e algumas partes dele
- Comentários não afetam o funcionamento do programa
  - São ignorados na hora da execução

Dica: Comente o código enquanto o estiver escrevendo, senão há uma forte probabilidade de não o fazer depois

Comentários em Java

- Há 3 formas de incluir comentários em Java:
  - Comentários de uma linha

Precedido de //

```
// Este é um comentário de uma linha
```

  - Comentários de múltiplas linhas

Deve estar entre /* */

```
/* Este tipo de comentário só termina
   quando o asterisco barra é encontrado */
```

  - Comentários Javadoc

Deve estar entre / */**

```
/** Isto é um comentário javadoc comment */
```

Javadoc é uma ferramenta que gera documentação externa

Reuso de Código

- Reusabilidade é um fator importante para acelerar o desenvolvimento de um sistema
- OO é uma metodologia que facilita o reuso
- Reuso pode se dar de várias formas:
  - Classe inteira
    - Uma classe é cliente de outra
  - Alguns atributos e/ou métodos
  - Trechos de programa ou métodos

Reuso de Código

- Existem muitas técnicas existentes para reutilizar código ou pelo menos aumentar o potencial de reuso
- Uma "técnica" infelizmente muito difundida é o *copy & paste*
  - Solução rápida e "boa" para quem não quer pensar
  - Solução péssima para a manutenção do sistema
    - Se uma alteração for requerida, esta também deverá ser propagada por todos os trechos repetidos
    - Se um bug for encontrado, terá de ser corrigido em todos os outros códigos repetidos
- Melhor do que isto é usar ***Refactoring***

Refactoring

- ***Refactoring*** consiste em uma série de técnicas que reestruturam o código do software, aumentando o potencial de reuso, extensibilidade e legibilidade
  - O comportamento do software continua o mesmo, só muda a estrutura
  - Ex: especialização com subclasses, extract method, rename, generalização com parâmetros, inline method, etc

Exemplo de Refactoring: Especialização

- Especializando com Subclasses
  - Comportamento
    - objetos da subclasse comportam-se como os objetos da superclasse
  - Reuso de Código
    - a descrição da superclasse pode ser usada para definir a subclasse
  - Extensibilidade
    - algumas operações da superclasse podem ser redefinidas na subclasse

Exemplo de Refactoring: Especialização

- Em Java, utilizamos a palavra reservada **extends** para informar que uma classe é subclasse da outra

```
public class MySubclass extends MyClass
```

- Mecanismo para definição de herança e subtipos

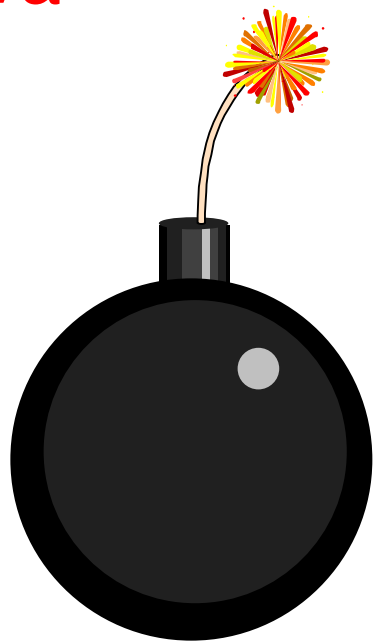
Erros Dinâmicos

- Erros que acontecem em tempo de execução, geralmente são mais difíceis de depurar do que erros estáticos
- Alguns possíveis erros
  - Tenta-se manipular algum arquivo que não exista
  - Estouro de memória
  - Divisão por zero
  - Manipular uma referência NULA

NullPointerException

- A referência obtida pode ser **`null`**
  - Não está associada a nenhum objeto
  - O método não pode ser executado
  - É levantada a exceção **`NullPointerException`** de Java

Muito cuidado!
Isso não deve acontecer!



Depurando Erros Dinâmicos

- Java possui mecanismos que facilitam tratar estes erros (exceções)
- IDEs existentes dão um bom suporte para depurar estes tipos de erros
  - Integram depuradores (debuggers)
  - Permitem inspecionar linha por linha do programa
- Outra forma, freqüentemente utilizada, é **comentar** partes do código para isolar (detectar onde está o erro)
  - Se ao comentar uma parte de código, o programa passa a funcionar normalmente, consegue-se reduzir o espaço de código a ser depurado

Resumindo ...

- Fatores de qualidade de software
  - Fatores de qualidade externos
  - Fatores de qualidade internos
- Técnicas para melhorar alguns fatores de qualidade de um programa
  - Comentários
  - Refactoring
- Depuração de erros dinâmicos
  - Erros dinâmicos
  - Depuração com comentários