Técnicas de Programação

Boas Práticas de Programação e Projeto

Profa. Elaine Venson

elainevenson@unb.br

Semestre: 2012-1

Conteúdo

- Boas práticas
  - Simplicidade
  - Modularidade
  - Extensibilidade
  - Evitar redundância
  - Portabilidade
  - **Internacionalização e localização**
  - Desenvolvimento e uso de API

Internacionalização e localização

- I18N -> *Internacionalization*:
  - Consiste no desenvolvimento e preparação de um software que **seja capaz de potencialmente lidar** com múltiplas linguagens.

- L10N -> *Localization*:
  - É o processo de customizar uma aplicação internacionalizada para **uma linguagem, região ou cultura específica**.

Internacionalização e Localização

- São dois passos no desenvolvimento de um software para torná-lo mais acessível a uma gama maior de países, culturas e linguagens
  - Geralmente a aplicação com internacionalização é desenvolvida por um grupo ou equipe
  - Em seguida, a localização é contratada de outro grupo que seja familiar com uma localização específica

Internacionalização e Localização

- O crescente acesso a mercados internacionais vem permitindo que desenvolvedores alcancem **clientes em diferentes países** e regiões cada vez mais distantes

- Para que o software seja adequado ao uso em cada local, o desenvolvimento do software deve ser feito movendo sempre que possível as informações de localização para **fora do código executável**

Internacionalização e Localização

- **Strings**:
  - A maior diferença entre regiões é a **linguagem**
  - Em sistemas não internacionalizados informações em linguagem específica são definidas em strings e **embutidas** no código
  - Exemplo:

```
void load_file( const char *filename )
{
   FILE *fp = fopen( filename, "r" );
   if ( fp == 0 )
   {
      /* this assumes that the error is displayed to an English-
      language user! */
      my_error( "Could not open file '%s'", filename )
      return;
   }
   . . .
}
```

Internacionalização e Localização

- **Strings**:
  - Idealmente as strings deveriam ser carregadas dinamicamente com base na configuração de localização do sistema
    - O Microsoft Windows
      - utiliza *string tables*
      - Strings são carregadas através de uma chamada de uma API por exemplo “LoadString()”
    - O Mac OS X
      - utiliza *string files* (para textos embutidos em programas) e arquivos .nib (para interface de usuário)
      - Strings são carregadas através de uma chamada de uma API por exemplo “NSLocalizedString()"
    -

Internacionalização e Localização

- **Strings**:
  - Utilizar APIs específicas de plataformas diminui a **portabilidade**
  - Uma outra abordagem é criar uma implementação de tabela de strings **específica da aplicação** para fazer o mapeamento de uma string para outra com base na localização

```
[English]
"Yes"= "Yes"
"No"="No"
[French]
"Yes"="Oui"
"No"="Non"
[Spanish]
"Yes"="Si"
"No"="No"
```

```
International_SetLocale( "French" );
my_message_box( International_GetString( "Do you wish to quit?" ),
                International_GetString( "Yes",
                International_GetString(  No" ) );
```

Internacionalização e Localização

- **Moeda**:
  - Há mais de 180 moedas em uso no mundo hoje, com diferentes nomes e convenções de formatação
  - Uma mesma moeda pode ter representações distintas entre países
    - $1.23 nos Estados Unidos
    - 1,23 na Europa

Internacionalização e Localização

- **Data e Hora**:
  - Data e hora também possuem muitas representações diferentes
    - 13:30
    - 1:30 PM
    - 13.30 (Finlândia)
  - Datas podem ter variadas permutações de dia, mês e ano
    - Separadores também podem mudar
    - Nomes dos meses, quando por extenso, de acordo com a linguagem
  - Um código que seja internacionalizado deve manter data e hora numericamente e utilizar uma **função de formatação** para locais específicos para fazer a conversão

Internacionalização e Localização

- **Elementos de Interface**:
  - Localização não se limita apenas a texto
  - Aplicações modernas dependem cada vez mais de saídas gráficas
  - Um mesmo ícone pode ter significados difentes para duas regiões distintas
    - Uma mão aberta significa “Pare” nos Estados Unidos e Europa, mas no oeste da África pode ser um insulto
  - Estes elementos devem ser configurados para que possam ser facilmente substituídos durante a localização

Conteúdo

- Boas práticas
  - Simplicidade
  - Modularidade
  - Extensibilidade
  - Evitar redundância
  - Portabilidade
  - Internacionalização e localização
  - **Desenvolvimento e uso de API**
- Nível de detalhamento do projeto
- Formas de representação

Desenvolvimento e Uso de API

- **Módulos** ajudam a particionar o problema
- Cada módulo define uma interface, que funciona como uma **fachada** por trás da qual se esconde a implementação interna
- O conjunto de operações disponíveis pode denominar uma API (***Application Programming Interface***)

Desenvolvimento e Uso de API

- Passos para a criação de boas interfaces:
  1. Identificar o **cliente** e o que ele ***quer*** fazer
  2. Identificar o **fornecedor** e o que ele é ***capaz*** de fazer
     - Projetos ruins definem operações nos locais errados levando a aumento de acoplamento e baixa coesão
  3. Inferir o **tipo de interface** requerida
     - Pode ser uma biblioteca, uma função, uma classe, um protocolo de rede, um objeto CORBA, etc.
  4. Determinar a **natureza** da operação
     - Que funcionalidades realmente precisam ser fornecidas

Desenvolvimento e Uso de API

- **Princípios**:
  - **Particionamento**
    - A interface define um **ponto de contato**, mas também é uma linha de **separação** entre o cliente e o implementador.
  - **Abstração**
    - A abstração permite ao observador se concentrar em decisões importantes, ignorando seletivamente certos detalhes
    - Organiza a realidade em uma representação mais simples, ajudando a lidar com a complexidade
    - Ao desenhar uma interface, uma abstração é criada escolhendo-se exatamente o que é importante para o usuário e o que deve ser escondido

Desenvolvimento e Uso de API

- **Princípios**:
  - **Compressão**
    - É a habilidade de uma interface em representar uma operação grande com algo simples
    - Compressão é geralmente o resultado de fazer boas abstrações
  - **Substituibilidade**
    - Você pode substituir uma implementação de uma interface por outra, se atender ao mesmo contrato
    - Em uma interface de ordenação qualquer algoritmo pode estar por trás: quicksort, heapsort, etc. O algoritmo pode ser trocado a qualquer momento desde que o comportamento visível através da interface seja o mesmo

Exercício

1) Desenvolver uma aplicação de um Quiz em Java com no mínimo 10 questões referentes à disciplina de Ténicas de Programação
2) Realizar a internacionalização da aplicação do Quiz
3) Realizar a localização para uma outra linguagem (por exemplo: Inglês)