**Roteiro 5**

**Tema:**

Herança.

**Objetivos:**

1. Aprender como se usa herança na orientação a objetos;
2. Saber descrever o que é o relacionamento de herança em linguagem de programação;
3. Entender e saber descrever o uso da seção "protected" das classes mãe;
4. Saber identificar no enunciado de um problema se existe o relacionamento de herança entre algumas das entidades;
5. Aprender a se comunicar entre objetos que tenham relacionamento de herança com objetos que usam dependência;
6. Aprender a usar constantes de uma forma diferente de #define para que as mesmas possam aparecer no debugger;
7. Fixar conceitos e o uso de elementos da gramática da linguagem C++;

Explicação  
  
Na natureza a herança ocorre quando um ser vivo herda características importantes de seu ancestral. Exemplos: reino dos mamíferos, aves, répteis, plantas.  
  
Este conceito foi trazido e reaproveitado com grandes vantagens nas linguagens de programação orientada a objetos.  
  
Numa linguagem de programação a herança ocorre quando uma classe herda de outra "métodos e atributos (variáveis)". Implementa-se herança da seguinte forma,  
  
Sejam duas classes, A e B, onde A é a classe mãe e B é a classe filha.   
  
A classe mãe A não altera sua sintaxe:

class A  
{  
private:

...   // declaração de variáveis privadas  
...   // declaração de métodos privados

public:

A( ) { } // construtor default  
~A( ) { } // destrutor  
...   // métodos da área pública

};

A classe filha B altera sua sintaxe normal para indicar que é herdeira de A:

class B : public A   // os dois pontinhos mais a palavra **public** caracteriazam que B é herdeira ou filha de A  
{  
private:

...   // declaração de variáveis privadas  
...   // declaração de métodos privados

public:

B( ){ } // construtor default  
~B( ){ }  // destrutor  
...   // métodos da área pública de B

};

O que muda na sintaxe da classe herdeira é que ela é definida em função da mãe usando ": public" seguido do nome da classe mãe.  
  
Caso a classe B seja herdeira de duas ou mais classes, por exemplo, de A e mais das classes X e Y, repete-se a sintaxe separando as novas mães entre si por vírgula:

class B : public A, public X, public Y  
{  
private:

...   // declaração de variáveis privadas  
...   // declaração de métodos privados

public:

B( ){ } // construtor default  
~B( ){ }  // destrutor  
...   // métodos da área pública de B

};

Como o analista descobre num problema real o relacionamento de herança entre duas de suas entidades?  
  
Suponha que se deseja saber se é possível usar o relacionamento de herança num problema real entre duas entidades, E1 e E2.  
  
Para ter certeza que há o relacionamento de herança deveremos fazer a pergunta: "E1 é do tipo E2?"  ou "E2 é do tipo E1?". Caso a resposta seja SIM para qualquer uma existe um relacionamento de herança entre as mesmas. Por exemplo, se E1 é do tipo E2 logo E1 é herdeira e E2 é a entidade ancestral.  
Vejamos alguns exemplos em domínios de sistemas de informação do mundo real.  
  
O observe as perguntas a seguir:  
  
1) Secretaria é do tipo funcionário?  
2) Diretor é do tipo funcionário?  
3) Aluno é do tipo professor?  
4) Professor é do tipo funcionário?  
5) Conta corrente é do tipo conta bancária?  
6) Conte popança é do tipo conta bancária?  
7) Forma de pagamento é do tipo cheque?  
8) Cheque é um tipo de forma de pagamento?  
9)  Forma de pagamento é um tipo de cartão de crédito?  
10) Cartão de crédito é um tipo de forma de pagamento?  
  
Respostas:  
  
1) Sim, secretária é filha e funcionário é mãe  
2) Sim, diretor é herdeiro, funcionário é a classe ancestral  
3) Não. Logo não há relacionamento de herança entre aluno e professor  
4) Pode ser. Se na minha escola o professor é registrado como um tipo de funcionário então Sim. Caso contrário não. Na maioria das escolas que conheço o professor não é registrado como um tipo de funcionário portanto a resposta para a maioria das escolas é Não.  
5) Sim. Portanto há relacionamento entre conta corrente e conta bancaria sendo conta bancária a classe ancestral ou classe base e a classe herdeira ou derivada é a classe conta corrente.  
6) Sim, portanto há relacionamento de herança entre conta poupança e conta bancária sendo, conta bancária a classe mãe e conta poupança a classe filho.  
7) Não. Portanto não há relacionamento de herança entre forma de pagamento e cheque. Observe que poderá sim haver o relacionamento no sentido inverso, ou seja, de cheque para forma de pagamento. Veja a próxima pergunta.  
8) Sim. Portanto há relacionamento de herança entre cheque e forma de pagamento. O cheque é herdeiro de forma de pagamento.  
9) Não. Portanto não há relacionamento de herança entre forma de pagamento e cartão de crédito. Mas pode haver no sentido inverso. Veja a próxima pergunta.  
10) Sim. Portanto quem é mãe quem é filha? Cartão de crédito é filha. Forma de pagamento é mãe.  
  
Quais conclusões podem ser tiradas destas respostas?

Primeiro - que o tipo mais genérico passa a ser a ancestral.  
Segundo - que o tipo mais especializado passa a ser o herdeiro.  
Terceiro - que não basta fazer a pergunta apenas num sentido. Temos que faze-la nos dois sentidos.

Vejamos se voces entenderam bem como se descobre se há relacionamento do tipo herança entre entidades.

Faça a pergunta chave relacionando os seguintes objetos e de a resposta voce mesmo descobrindo se existe o relacionamento e quem é quem no mesmo.

a) Peixe e lambari  
b) Arvore e pinheiro  
c) Cachorro e vira-lata  
d) Cachorro e pastor alemão  
e) Fabricante e marca fiat  
f) Honda e fabricante  
g) Honda FIT e Honda  
h) KA e Ford

O que está por traz destas perguntas e respostas?  
  
Nos interessa esse mecanismo de descobrir se uma entidade é herdeira ou ancestral porque cada uma delas poderá ser uma classe no sistema orientado a objetos.  
  
Quando descobrirmos que há o relacionamento de herança poderemos usar a sintaxe de herança em orientação a objetos e isto nos trará grandes vantagens!!!!  
  
Quais foram as grandes vantagens que os cientistas e analistas descobriram sobre o uso de herança na linguagem de programação?  
  
A) Que se pode desenvolver um software com menos esforço de lógica uma vez que podemos herdar coisa prontas.  
B) Que ao herdar coisas prontas e já usadas em outros sistemas de informação estaremos implementando um software de maior qualidade uma vez que parte do software ( o que estamos reutilizando) já foi testado em outros sistemas de informação com sucesso.  
C) Que procedendo desta forma as empresas estarão fabricando artefatos de software reutilizáveis que muito provavelmente possibilitarão a criação de novos sistemas em menor tempo do que se o sistema todo tivesse que ser desenvolvido a partir da estaca zero.  
D) Que criando software com herança a empresa deverá colocar no mercado seus produtos mais rapidamente do que se o processo de desenvolvimento não tivesse herança.  
E) Que assim fazendo a empresa estará facilitando a manutenção do software por que haverá menos código novo a ser testado.  
F) Que a herança proporciona uma maneira mais fácil de ler e de escrever código do que o velho hábito estruturado e funcional.  
  
  
Seção Protegida  
  
A seção protegida é sempre colocada na classe mãe.  
  
É um meio do filho e a mãe se comunicarem reservadamente entre si. Nenhum objeto do filho consegue enxergar esta seção. Apenas de dentro do smétodos dos filhos ela é visível.   
  
Como usá-la?

Toda vez que a mãe tiver variáveis privadas e houver a necessidade de compartilhamento destas variáveis entre a mãe e o filho, o procedimento padrão é:

As variáveis ou métodos privados ficam na mãe.  
  
Na seção protegida deverão ficar métodos que acessarão os dados da área privada e os disponibilizarão para os filhos. Para disponibilizar dados para os filhos os métodos da área protegida poderão usar dois procedimentos já conhecidos:

1. Através do valor de retorno do método protegido.
2. O filho passará como argumento dos métodos protegidos variáveis vetoriais.

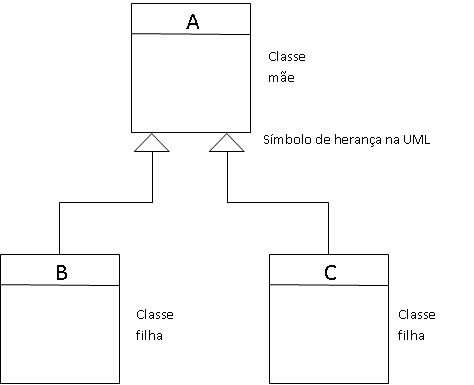
A seção protegida é como uma interface privada entre mãe e filhos. Apenas eles sabem deste canal de comunicação. Nenhum objeto filho pode acessar o seu conteúdo usando o "pontinho" de ativação de métodos!  
  
Representação da herança na UML - Unified Modeling Language  
  
Suponha que tres classes A, B e C tenham o relacionamento de herança entre elas da seguinte maneira: A é a mãe e B e C são os filhos. Graficamente se representa na UML este relacionamento através de um triangulo cujo vértice é colocado na mãe e cuja base é disposta voltada para o filho. Liga-se este triangulo com linha cheia. Veja a figura 1 a seguir.  
                              

Figura 1. Representação gráfica de herança na UML. O vértice  
indica a mãe e a base fica em frente a classe filho.

Porque o triângulo é o símbolo da herança? Porque o seu formato indica um estreitamento de funcionalidades querendo exprimir o sentimento real de que as classes ancestrais sempre possuem menos funcionalidades e dados do que as classes herdeiras. As classes herdeiras sempre terão mais "coisas" do que as respectivas ancestrais.   
  
Uma outra forma de chamar o relacionamento de herança é relacioamento de especialização-generalização por causa das classes mãe possuirem as partes comuns de todos os filhos enquanto que cada um além de herdar esta parte comum adiciona mais funcionalidades especializando cada filho.  
  
Para treinar a criar um sistema de informação orientado a objetos com o uso de herança nós faremos a resolução de um problema onde teremos que em primeiro lugar aprender a descobrir o relacionamento de herança, dar os nomes as classes mãe e filhas, e usar juntamente com o restante do nosso sistema e com o relacionamento de dependencia que já sabemos usar.  
  
Em resumo o que precisamos inicialmente é aprender a criar um software reutilizável através do relacionamento de herança. Mais tarde aprederemos como reutilizar um software já pronto num novo projeto através do relacionamento de herança.  
  
Enunciado (descrição) do problema

*Criar um programa OO para ler o raio de um círculo, calcular sua área e perímetro e mostrar na tela juntamente com o respectivo nome da  figura, (circulo), e as unidades de engenharia correspondentes a área (em u.e.2) e ao perímetro do círculo (em u.e.). Em seguida, repetir este mesmo procedimento para a figura do tipo retângulo. O programa deverá entrar com os dois lados de um retângulo e imprimir sua áre e perímetro bem como o nome da figura (retangulo) e as unidades de engenharia de área e perímetro do retângulo. Finalmente repetir os mesmos procedimentos para um triângulo. Neste caso, o usuário deverá entrar com os 3 lados de um triangulo qualquer e o programa deverá mostrar na tela a área, o perímetro juntamente com o nome da figura (triangulo) e as unidades de engenharia da área e perímetro correspondentes ao triangulo. Para ilustrar melhor observar o caso de teste mostrando as informações referentes ao círculo no final do roteiro cinco.*

*A solução deverá ter um menu inicial com as seguintes opções:*

*Escolha uma opção:*

1. *Circulo*
2. *Retangulo*
3. *Triangulo*
4. *Sair*

*Opcao:*

*O usuário escolhe a opção desejada: 1,2,3 ou 4. De posse da opção escolhida, o programa solicita os dados da figura conforme explicado anteriormente, lê os dados solicitados, calcula a área e o perímetro da figura e mostra os resultados na tela. Neste ponto o usuário deve apertar uma tecla qualquer para que o programa retorne ao menu anterior, limpando primeiro a tela.*

*O usuário pode repetir o cálculo da área e do perímetro de várias figuras separadamento. Quando desejar encerrar a execução do programa ele digitará a opção 4 (“Sair”).*

Fórmulas úteis para este exercicio:

Círculo

*area = Pi \* raio ao quadrado*

*perímetro = 2 \* Pi \* raio*

Onde:

Pi = 3.1415

Retangulo

*area = a \* b*

*perímetro = 2 \* (a + b)*

Onde:

   a,b são os lados do retângulo

Triângulo

A formula da area do triangulo

*perimetro = a + b + c*

Onde:

p é o semi-perímetro do triângulo  
a, b, c  são as medidas dos 3 lados do triãngulo

A é o resultado da raiz quadrada e é a área desejada

Em C++  existe uma função da biblioteca padrão, denominada sqrt( double) que calcula a raiz quadrada do número fornecido como argumento. Por exemplo, sqrt(a), calcula e devolve para quem chamou a raiz quadrada do núemro "a" o qual deve ser do tipo double.  
  
Para usá-la, é necessário antes da linha onde for adicionar a chamada a esta função, declarar o seu nome como sendo uma função. Assim o compilador não reclamará. Para isto deve ser incluído o arquivo <math.h> no Parametro.h na sequencia dos includes lá existentes.

Fórmula para descobrir se os 3 lados fornecidos podem ou não formar um triângulo:

Verificar se se trata de triângulo ou não

O aluno deverá acompanhar a partir deste ponto a aula juntamente com o oprofessor.  
  
Massa de teste para verificar a entrada de dados do círculo (Raio, Nome, Unidade de area, Unidade de Perímetro, Area, Perimetro):  
  
