Herança - Conceitos Básicos

1. Introdução

Herança é a capacidade que instâncias de uma classe filha ou Subclasse de acessar dados e procedimentos ou métodos associados com uma Classe Parente ou Superclasse. Ou seja, uma Subclasse herdará todos os métodos e dados da Superclasse. Além disso, a Subclasse poderá definir métodos e dados, ou mesmo em alguns casos, redefinir alguns métodos da Superclasse.

A Herança é sempre transitiva. Ou seja, uma classe pode herdar aspectos de Superclasses que estão a muitos níveis de distância. Em outras palavras, uma classe Filha pode herdar métodos da classe Pai, classe Avô, classe Bisavô, classe Tataravô e assim sucessivamente.

Para identificar se duas classes têm o relacionamento de herança, basta utilizar o teste “IS-A” ou “é-um(a)”.

2. Teste “IS-A”

Se soar de bom grado dizer que o conceito A “is a” conceito B, então existe uma relação de herança entre A e B. São exemplos de conceitos que passam no teste “IS-A”:

- Carro é um Meio de Transporte
- Professor é uma Pessoa
- Aluno é uma Pessoa
- Pássaro é um Animal
- Gato é um Animal
- Lista de Alunos é uma Lista
- Triângulo é um Objeto Gráfico
- Quadrado é um Objeto Gráfico
- Cantor é um Músico
- Guitarrista é um Músico
- Músico é uma Pessoa

Por outro lado, alguns relacionamentos são reprovados no teste “IS-A”. São alguns exemplos de relacionamentos que são reprovados no teste:

- Motor é um Carro
- Caixa de Edição é uma Janela
- Vídeo é uma Música
- Professor é uma Disciplina
- Livro é um Animal
- Livro é um Biblioteca

2. Por que utilizar Herança

Duas motivações justificam o seu uso:

- Reuso de Código: Como a Subclasse herda os atributos e métodos da Superclasse, não há a necessidade de redigitar parte ou todo o código.
- Reuso de Conceitos: Isso ocorre quando a Subclasse redefine algum método da Superclasse. Apesar de não haver um reuso completo do código da Superclasse, a Subclasse e a Superclasse compartilham apenas a definição do método.

2. Representação UML

A Representação no diagrama UML da relação de Herança entre classe é feita por meio de uma linha ligando as duas classes e um símbolo triângulo apontando para Superclasse. As Figuras 1 e 2, a seguir, exemplificam isso.

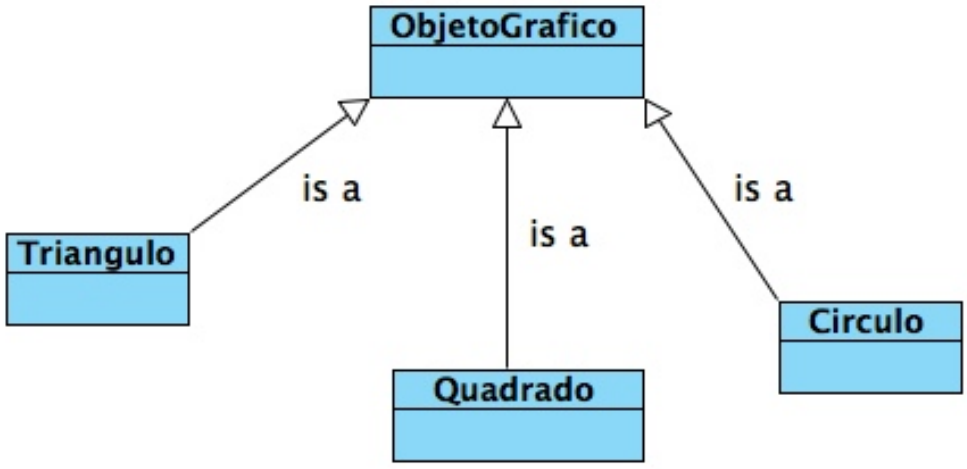

Figura 1 - Exemplo UML de Representação de Herança entre as Subclasses Triângulo, Circulo e Quadrado com a Superclasse ObjetoGrafico

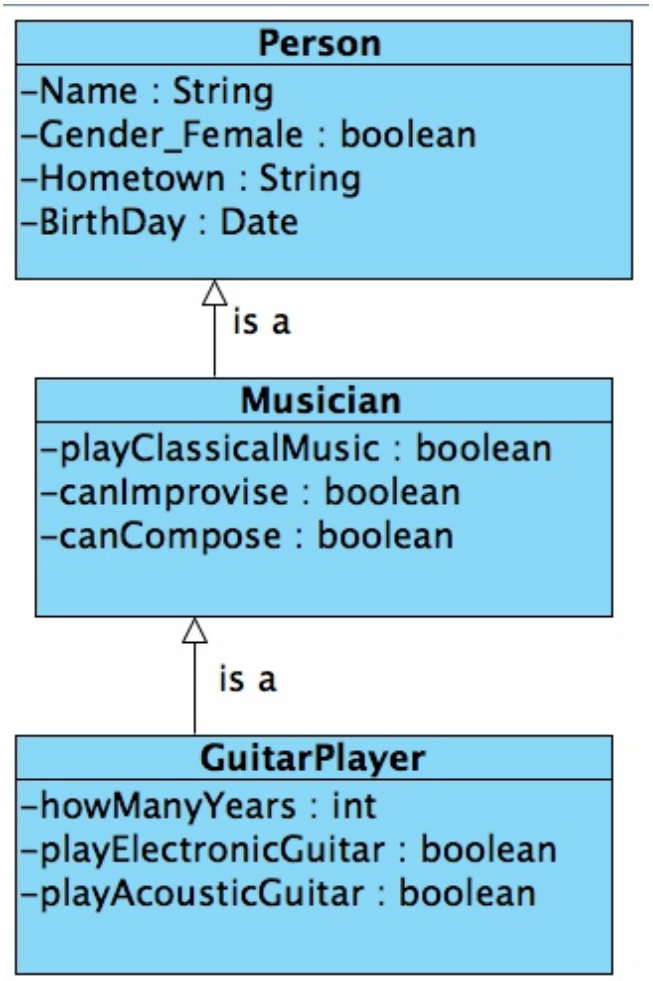

Figura 2 - Exemplo UML de Representação de Herança de 2 níveis.
(ps: o código da implementação desse exemplo encontra-se no site)

Na Figura 1, as Subclasses Triângulo, Quadrado e Circulo herdam todos os métodos e atributos da classe ObjetoGrafico. Esses atributos e métodos foram omitidos da representação. Na Figura 2, por sua vez, a classe Musician herda todos os atributos e métodos da classe Person. Além disso, a classe Musician adiciona novos atributos. Na Figura 2, a classe GuitarPlayer, herda tudo o que a classe Musician possui, ou seja, os atributos caImprovise, canCompose e playClassicalMusic estão disponíveis para a classe GuitarPlayer. Além disso, a classe GuitarPlayer também tem acesso aos atributos Name, Gender_Female e BirthDay por meio da herança entre a classe Musician e Person.

Destaca-se neste momento, que os relacionamento Associação e Dependência entre as classes continuam podendo ser empregados na definição do projeto.

Por exemplo, um GuitarPlayer pode ter um “0..*” Álbuns gravados formando a sua discografia e um Álbum pode ter sido gravado por “1..*” GuitarPlayer. A Figura 3 ilustra isso. A classe Person, por sua vez, relaciona-se com a classe Address. Ou seja, uma instância da classe Person possui uma instância da classe Address.

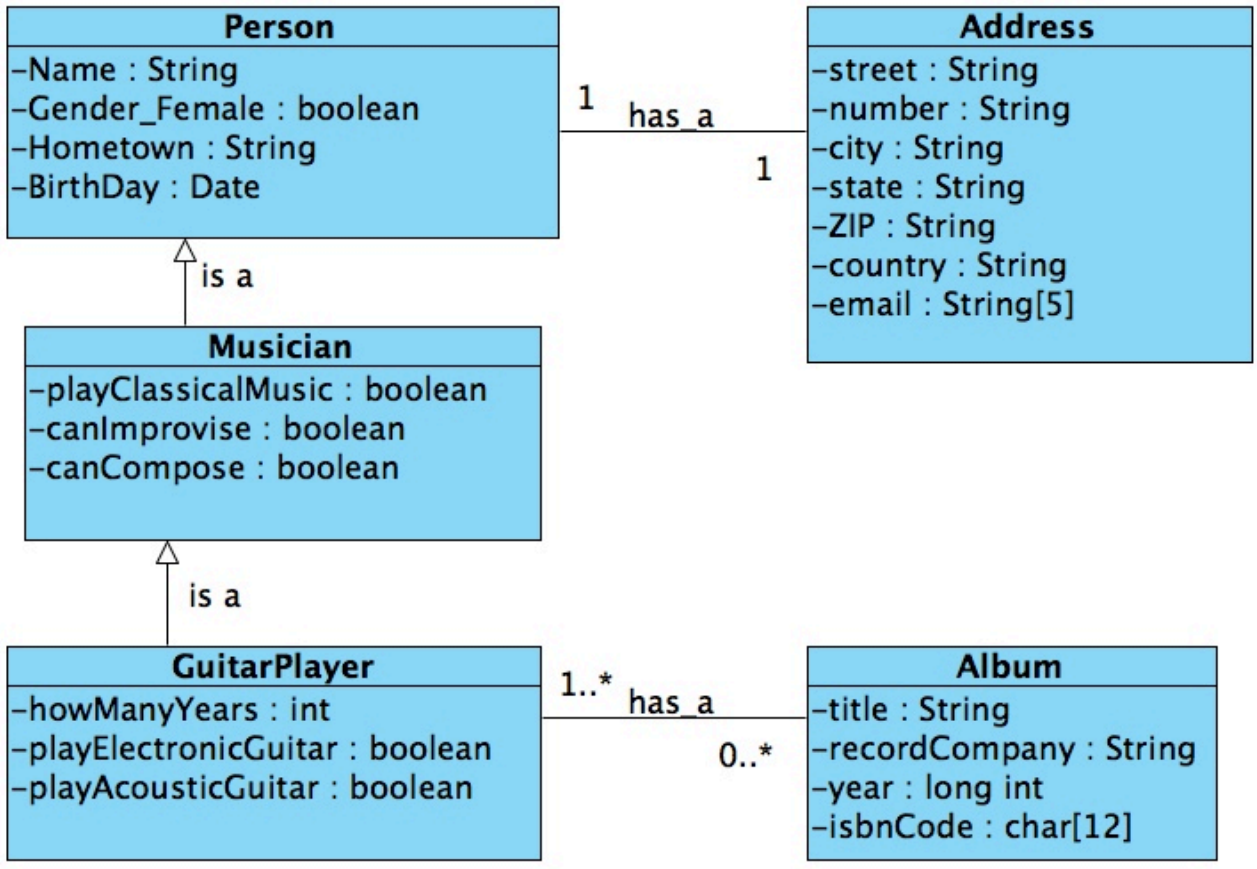

Figura 3 - Exemplo UML de Representação de Herança ou Genralização e Associação.

3. Herança na linguagem Java

Na linguagem Java a Herança é representada da seguinte forma:

```
public class SubClasse extends SuperClasse
{  ..... }
```

4. Tarefa

Implementar na linguagem Java o diagrama da Figura 3.