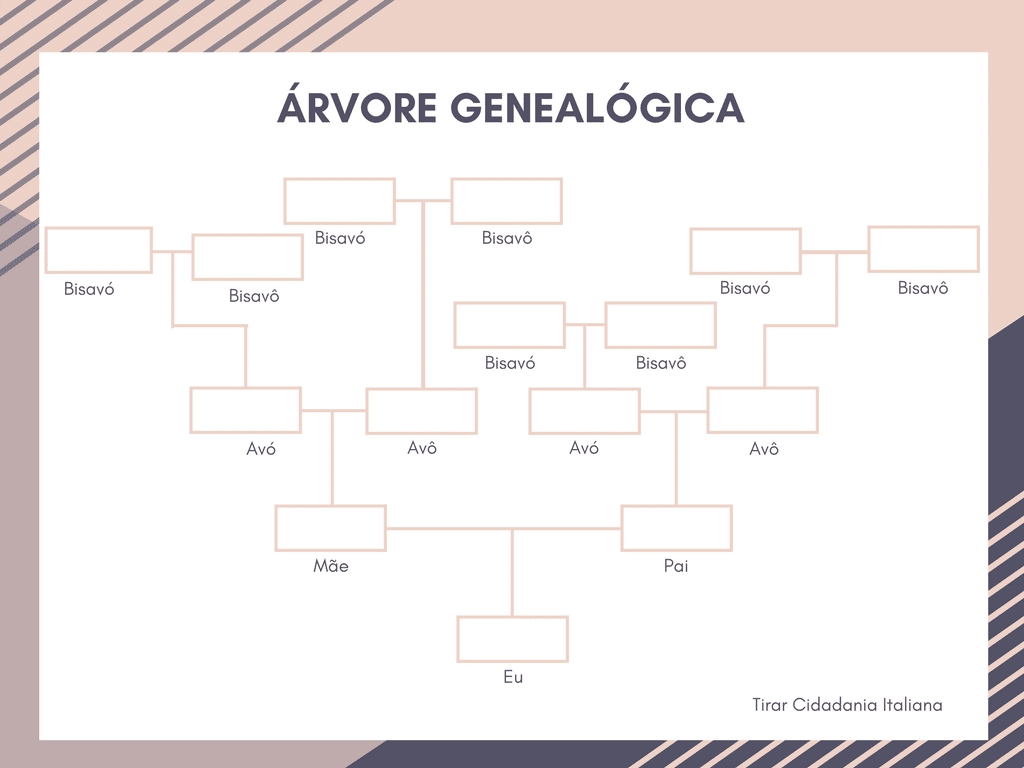
INSTITUTO POLITÉCNICO DE SANTARÉM

Escola Superior de Gestão e Tecnologias de Santarém

Licenciatura de Informática

Linguagens de Programação

Discentes:

Diogo Nogueira, Nº 170100223

João Agostinho, Nº 170100278

Docentes:

Prof. António Roberto e Prof. Ricardo Rodrigues

Santarém, 10 de junho de 2020

**Resumo**

O objetivo deste trabalho é apresentar a árvore genealógica de uma família.

O projeto será feito em Java, constituído por 2 classes. A classe Main que servirá para executar o código e a classe PersonNode(Nó de cada Pessoa) onde guarda as informações sobre a família pretendida. A classe PersonNode tem todos os atributos e métodos necessários paras as operações a seguir enunciadas.

A nossa árvore realiza as seguintes operações:

* Procura por qualquer pessoa pelo nome
* Devolve o grau de parentesco dos filhos
* Adiciona um filho a pessoa que pretendemos
* Desenha a árvore para o utilizador (com nome e ano das pessoas)
* Responde a perguntas relacionadas com arvore e com os seus filhos

**Introdução**

O presente trabalho consiste no relatório do trabalho prático, efetuado na unidade curricular de Linguagens de Programação da Licenciatura Informática, da ESGTS.

Neste trabalho tínhamos de utilizar uma das estruturas dadas em aula e sugerida pelos docentes. Tínhamos duas alternativas: as árvores ou os grafos. Nós optamos pela árvore, mais especificamente por uma árvore binária, que é uma estrutura de dados e que nos pareceu a mais indicada para o tipo de trabalho que nós iriamos realizar e porque tínhamos mais conhecimentos sobre essa estrutura.

O tema do nosso trabalho é uma arvore genológica que pretende mostrar a estrutura de uma família em geral. Decidimos generalizar para uma família qualquer invés de optar por uma família específica.

**Desenvolvimento**

No início do projeto para decidirmos o tema tivemos uma reunião para debater ideias e para perceber se as ideias eram viáveis e de como as poderíamos pôr em prática.

Depois da ideia estar decidida fomos começar a trabalhar no código e foi nessa parte é que sentimos algumas dificuldades, nem foi tanto na parte de escrever código foi mais na maneira de como iriamos ligar a arvore e os seus respetivos filhos. Pensamos em implementar uma classe que iria servir para interligar os filhos á arvore e assim com duas classes ficávamos com o problema resolvido, mas isto não resultou muito bem porque não estávamos a fazer bem a interligação. Após esta tentativa falhada optamos por tudo numa classe e esta ideia pareceu-nos muito bem.

Começamos por criar a classe e pensar quais os atributos necessários então chegamos a conclusão que teríamos de ter dois atributos (name e year) que iriam guardar informação sobre o nome de cada pessoa e o seu ano.

Uma imagem com pássaro

Descrição gerada automaticamente

Figura - Atributos

Na Figura 1 pode-se verificar que optámos por colocar os atributos privados por causa da segurança e para não se poderem acederem diretamente. Críamos um arrayList para guardar os filhos da árvore.

Após os atributos definidos fomos criar os construtores.

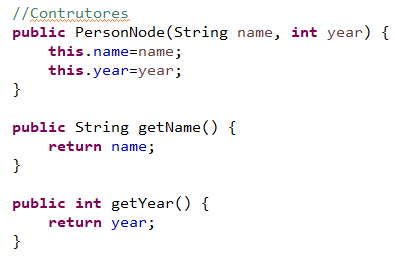


Figura - Construtores

A Figura 2 demonstra apenas criamos os get’s dos atributos porque não pretendemos que se possa alterar o valor das variáveis, mas sim ir buscar esses valores.

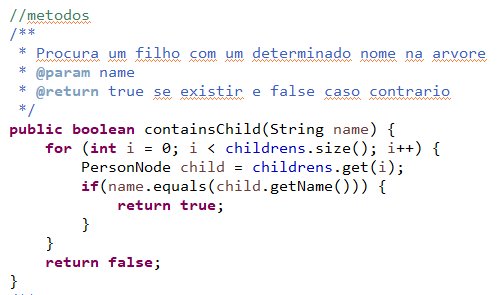


Figura - Método procura pelo filho

A Figura 3 mostra o método de procurar pelo filho em que passamos a variável name como parâmetro. Utilizamos um for para procurar em todos os filhos(childrens.size()) e depois comparamos o nome do filho com o nome pretendido.

Uma imagem com captura de ecrã, pássaro

Descrição gerada automaticamente

Figura - Método adicionar

Na Figura 4 passamos como parâmetro um novo filho e adicionamos esse filho ao nosso arrayList de filhos.

**Uma imagem com captura de ecrã

Descrição gerada automaticamente**

Figura - Procurar por qualquer pessoa

Na Figura 5 passamos como parâmetro o nome da pessoa e comparamos com o nome que pretendemos depois fazemos um ciclo(for) para verificar cada pessoa e depois uma condição(if) para verificar se for igual devolve o nome, se não for não devolve nada.

**Uma imagem com captura de ecrã

Descrição gerada automaticamente**

Figura - Devolve o grau de parentesco dos filhos

Na Figura 6 utilizámos dois métodos para conseguir devolver o grau de parentesco. Para conseguir devolver o grau optámos por fazê-lo de forma recursiva porque nos pareceu ser mais simples.

**Uma imagem com captura de ecrã, pássaro

Descrição gerada automaticamente**

Figura - Inserir um filho a pessoa pretendida

**Uma imagem com captura de ecrã

Descrição gerada automaticamente**Na Figura 7 criamos um método para inserir um filho a pessoa que nós pretendêssemos foi necessário o nome(name) e o filho em questão(PersonNode newChild) primeiro tínhamos de comprar se era o filho que queríamos inserir fizemos isso através do nome e depois se fosse criamos um ciclo para inserir o filho.

Figura - Métodos complementares

Na Figura 8 pode-se ver que criamos um método String para organizar a forma de ver no output, e os outros três métodos são para desenhar a árvore em que precisamos de dois métodos auxiliares (privados por boa ética).

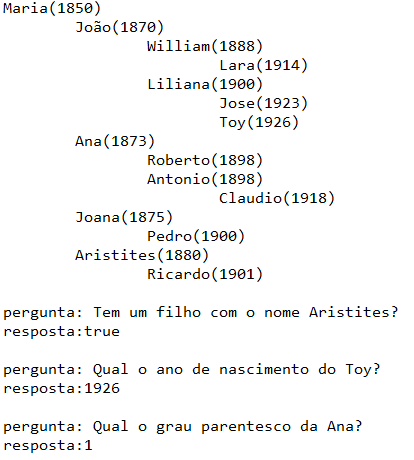
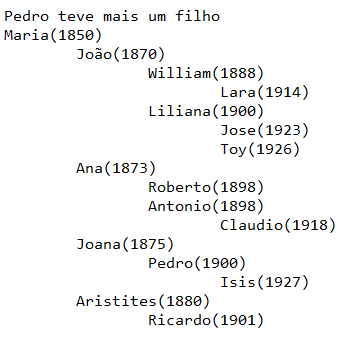


Figura – Output

A Figura 9 demonstra a execução de todo o código explicado até este ponto. Começa por demonstrar graficamente como está a arvore depois temos algumas perguntas predefinas que respondem com booleanos (verdadeiros os falsos), com a data de nascimento da pessoa e com o grau de parentesco. Na parte direta figura inserimos um filho ao pedro e aparece representado graficamente na árvore.

**Conclusão**

Gostámos imenso de realizar esta trabalho e percebemos o quão importante são as estruturas de dados, mais especificamente as árvores binárias. Tivemos algumas dificuldades ao nível da implementação, mas com esforço e com paciência conseguimos cumprir todos os prazos e achamos ter superado os objetivos do trabalho.

O mais complicado neste trabalho foi perceber como poderíamos ligar a árvore aos seus filhos após essa pequena complicação que é mais a nível de pensamento e de lógica fomos ultrapassando os pequenos erros que iam surgindo.

Concluímos que o trabalho foi bastante enriquecedor e importante porque nos deu a verdadeira noção de que como se programa com estruturas de dados mais complexas.