
Mestrado Integrado em Engenharia Informática e Computação

Programação

2º Semestre

Prof. Jorge Alves da Silva

Prof. Rui Ferreira da Silva

Elaborado por:

Grupo **9** da Turma **3**

Daniel dos Santos Teixeira - 100509067

Luís Guilherme Ribeiro de Castro Silva Martins - 100509105

Data de entrega:

21/05/2011

# 1 - Introdução

O projecto desenvolvido no âmbito da unidade curricular de Programação do primeiro ano do curso de Engenharia Informática e Computação consiste no desenvolvimento de um jogo de Dominó *All Fives* em linguagem C++ .

Sendo assim, o projecto dividiu-se em duas partes, a concepção e a implementação. A primeira parte consiste em "esboçar" o projecto e planear a maneira como o abordar. Esta fase é muito importante pois se não for bem pensada e bem estruturada corre-se o risco de quando se partir para a fase da implementação não haver funções suficientes ou haver funções inúteis, entre outros problemas cuja solução pode passar por uma (longa) reformulação do projecto.

A segunda fase do projecto é onde se desenvolve o código e se põe, neste caso, o jogo a funcionar. Nesta parte e que se percebem as limitações que o projecto anteriormente concebido possui e se tenta corrigir. Na parte da valorização, não nos foi possível introduzir um jogador automático no jogo mas conseguimos aplicar uma função que maximiza a janela, tornando a visualização do jogo muito mais prática. Apesar da estrutura do programa estar praticamente desenvolvida, o jogo não é jogável porque aparece um erro sempre que se tenta jogar uma peça no lado escolhido. Este erro que até à escrita deste relatório não foi corrigido e "deita por terra" tudo o que tinha sido feito em termos de regras e funções que correspondem às regras e pontuações que só é aplicado depois desse erro.

# 2. Concepção e Implementação

## 2.1. Estrutura de Classes

A estrutura de classes do projecto assenta em quatro classes essenciais: *Bone, Boneyard, Player e Board.*

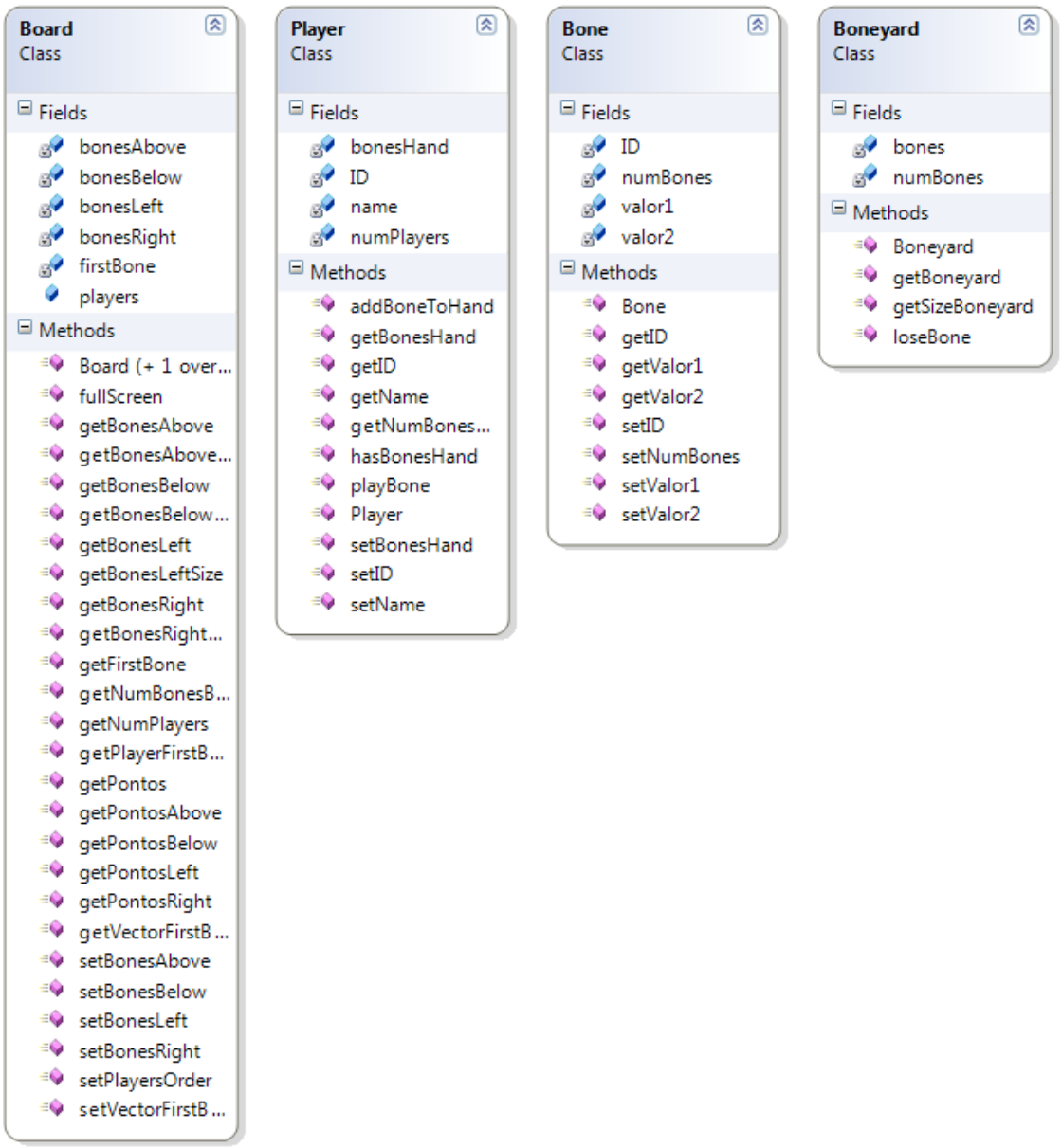
A classe *Bone* é a classe utilizada para implementar as peças, bem como as funções básicas para obter os valores de cada peça. Nesta classe foi também criado um tipo de variável *Bone* que nos facilitou todo o projecto uma vez que não tínhamos que identificar a peça pelos seus dois valores correspondentes. Uma vez que todo o jogo é baseado nestas peças, esta classe foi de uma fulcral importância para o projecto.

A classe *Boneyard* é onde foi implementado o monte das peças, criando-as todas no início de cada partida e é onde os jogadores as vão buscar sempre que necessário. É uma classe importante porque gera todas as peças do jogo, o que significa que não haveria jogo sem ela.

A classe *Player* é a classe que implementa o jogador como elemento principal do jogo. Ela cria um jogador com o seu nome, ID e as peças que lhe são atribuídas. É uma classe importante na medida em que disponibiliza ao jogo todas as informações relativas ao jogador inclusive funções que jogam a peça e a eliminam da mão do jogador.

A classe *Board* é a classe mais complexa do projecto e que envolve ao todo 26 funções que dinamizam o jogo e o tornam possível. Esta classe tem a representação gráfica do jogo e contem também todas as regras pelas quais o jogo se rege executando todas as verificações para saber se um jogador pode jogar a peça escolhida no sítio escolhido. É também responsável pela contabilização dos pontos e é sem dúvida essencial para que o jogo corra e a mais importante de todo o projecto.

Na imagem seguinte está o diagrama de classes do projecto:


Board
Class
Fields
bonesAbove
bonesBelow
bonesLeft
bonesRight
firstBone
players
Methods
Board (+ 1 over...
fullScreen
getBonesAbove
getBonesAbove...
getBonesBelow
getBonesBelow...
getBonesLeft
getBonesLeftSize
getBonesRight
getBonesRight...
getFirstBone
getNumBonesB...
getNumPlayers
getPlayerFirstB...
getPontos
getPontosAbove
getPontosBelow
getPontosLeft
getPontosRight
getVectorFirstB...
setBonesAbove
setBonesBelow
setBonesLeft
setBonesRight
setPlayersOrder
setVectorFirstB...
Player
Class
Fields
bonesHand
ID
name
numPlayers
Methods
addBoneToHand
getBonesHand
getID
getName
getNumBones...
hasBonesHand
playBone
Player
setBonesHand
setID
setName
Bone
Class
Fields
ID
numBones
valor1
valor2
Methods
Bone
getID
getValor1
getValor2
setID
setNumBones
setValor1
setValor2
Boneyard
Class
Fields
bones
numBones
Methods
Boneyard
getBoneyard
getSizeBoneyard
loseBone

## 2.2. Implementação das classes e algoritmos utilizados

Relativamente a à implementação das classes, note-se que a classe *Board* é a que tem mais trabalho realizado pois faz todo o jogo correr desde as regras à representação gráfica. Dado este facto que tornou esta classe muito complexa difícil de analisar pelo que surgiu um erro que afectou todo o projecto. Uma vez que não o conseguimos encontrar e resolver, houve funções criadas já testadas que não se podem demonstrar e portanto valorizar.

# 3. Conclusão

Finalizado o projecto, chegamos à conclusão que este projecto não foi bem-sucedido uma vez que não conseguimos alcançar os objectivos básicos que tinham como ponto essencial por o jogo executável. Para além disso, não conseguimos obter os pontos de valorização correspondentes à implementação do jogador automático.

Em relação ao que tinha sido planeado na primeira fase do projecto, o programa desenvolvido só teve melhorias, fruto de uma má preparação na sua concepção. Houve inúmeras funções criadas somente na fase final do projecto que foram essenciais para o funcionamento do programa, assim como para a implementação de regras básicas do jogo. O programa tem várias limitações que foram surgindo à medida que ia sendo desenvolvido nomeadamente na representação gráfica das peças e que por razões de tempo não puderam ser implementadas. Para além dessas limitações, existe um erro na altura do jogador colocar a peça escolhida no lado escolhido que é fatal e inutiliza praticamente todo o código criado.

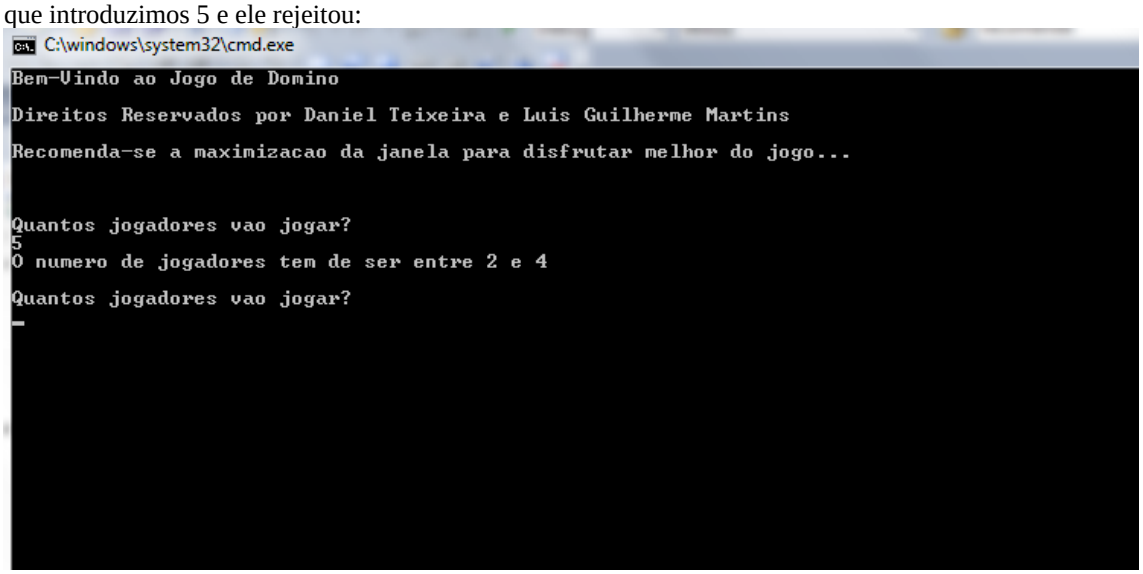
Umas das conclusões mais importantes e pertinentes que influenciou todo o projecto foi sem dúvida o tema. Depois de concluído o primeiro projecto e do que tínhamos aprendido com ele, estávamos à espera de um projecto que fosse mais interessante. A ideia de ser um jogo foi apelativa mas de Dominó foi um pouco desapontante para o grupo e fez-nos perder a motivação que tínhamos ganho com o projecto anterior. A culpa obviamente pertence inteiramente ao grupo que não se aplicou o suficiente mas se o tema do projecto fosse outro, decerto que a motivação e o empenho seriam muito maiores e os resultados seriam também muito mais positivos.

# 4. Bibliografia

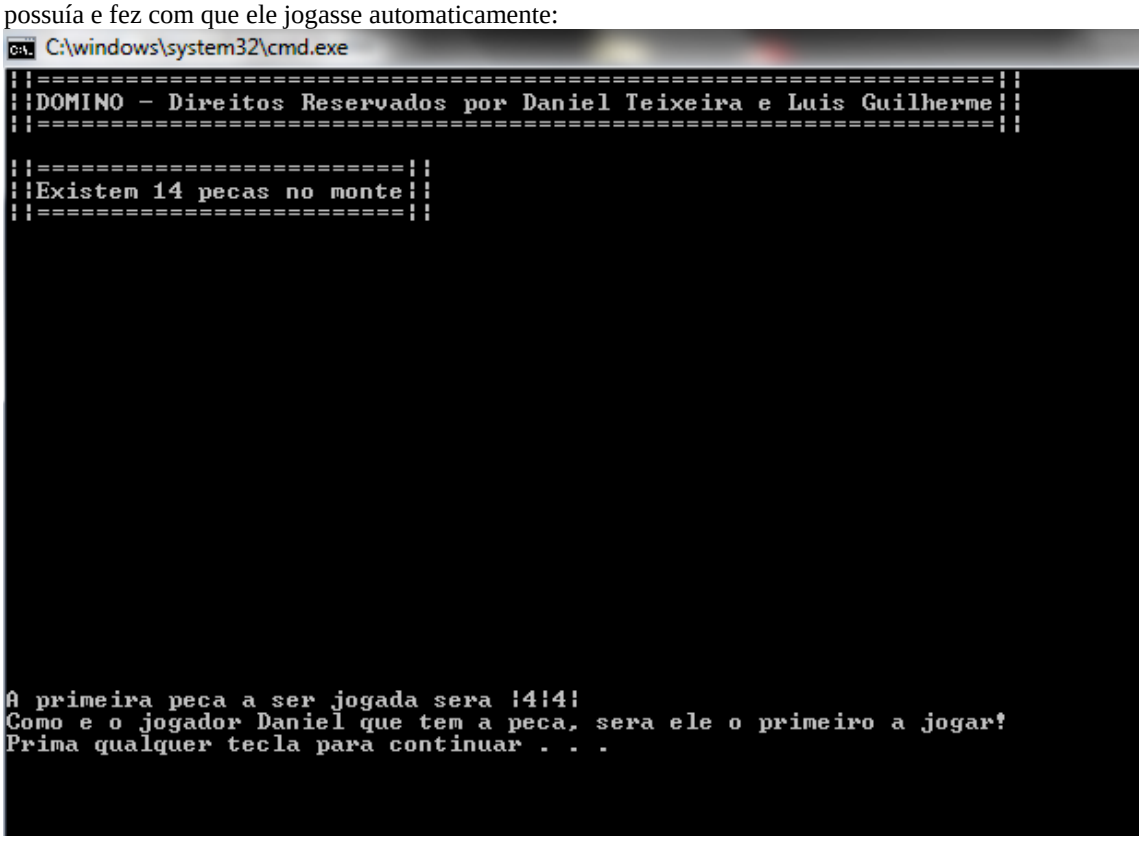

- Anónimo, "All Fives Domino Rules", 2011, http://www.domino-games.com/domino-rules/allfives-rules.html (consultado em 2011-05-17)
- P. Deitel, H. Deitel: "C++ How to Program", Pearson, 2010

# 5. Apêndices

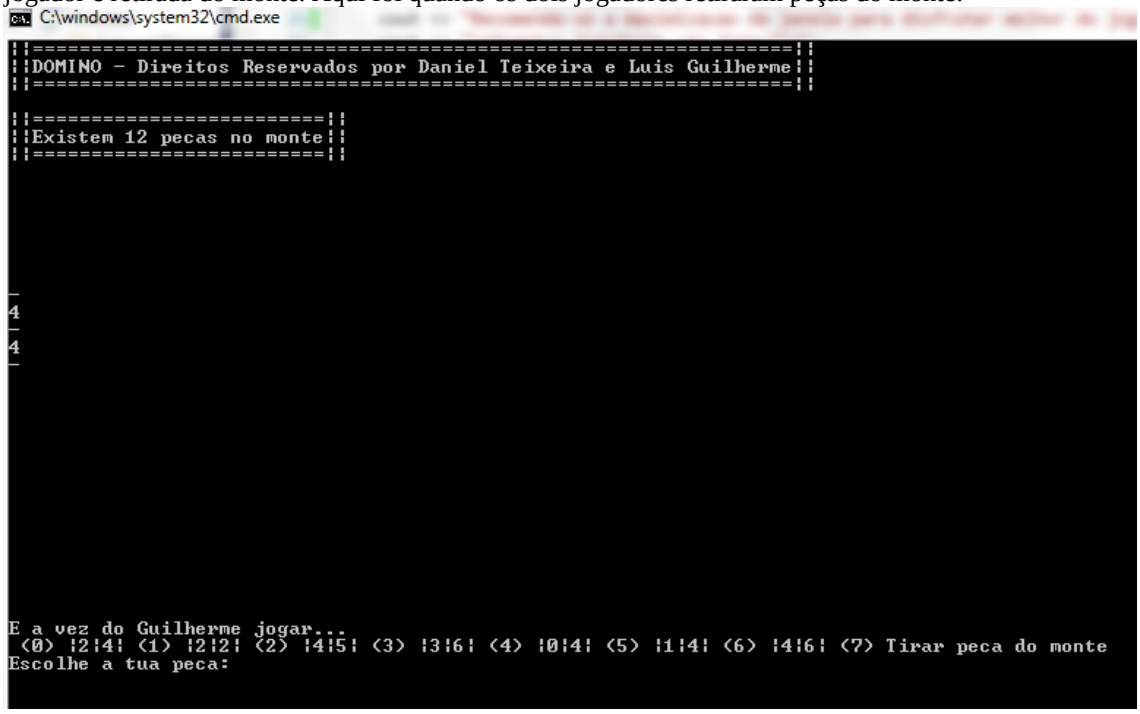
Na figura seguinte demonstra-se que o programa só aceita números de jogadores que vão de 2 a 4 sendo que introduzimos 5 e ele rejeitou:




Nesta figura vemos que o programa detectou o jogador com o double maior e qual o jogador que a possuía e fez com que ele jogasse automaticamente:



Caso seja necessário retirar uma peça do monte, basta premir a tecla 7 e uma peça é adicionada ao jogador e retirada do monte. Aqui foi quando os dois jogadores retiraram peças do monte:




Exemplo do erro que nos afecta e nos impossibilita de demonstrar o resto do jogo e funções que estavam já feitas e testadas: