Programação Orientada a Objetos
Padrões de Projeto
Game of Life

Rodrigo Bonifácio

6 de fevereiro de 2013

Game of Life

Proposto pelo matemático John Conway (Princenton University) em 1970, esse não corresponde a um jogo típico ...

Game of Life

Proposto pelo matemático John Conway (Princenton University) em 1970, esse não corresponde a um jogo típico . . .

- Não existem jogadores
- Não existem vencedores ou perdedores

Uma vez que as "peças" são posicionadas, as regras determinam tudo que acontecerá a seguir.

Regras

As células, dispostas em um "tabuleiro" em forma de grade bidimensional, podem estar vivas ou mortas. Uma célula viva é indicada por uma marca na posição específica do tabuleiro.

Regras

As células, dispostas em um "tabuleiro" em forma de grade bidimensional, podem estar vivas ou mortas. Uma célula viva é indicada por uma marca na posição específica do tabuleiro. Uma nova geração de células depende da vizinhança de cada célula específica (cada célula possui no máximo 8 células vizinhas).

Regras

As células, dispostas em um "tabuleiro" em forma de grade bidimensional, podem estar vivas ou mortas. Uma célula viva é indicada por uma marca na posição específica do tabuleiro. Uma nova geração de células depende da vizinhança de cada célula específica (cada célula possui no máximo 8 células vizinhas).

- Uma célula morta com exatamente três células vizinhas vivas se torna uma célula viva (nascimento).
- Uma célula viva com duas ou três células vizinhas vivas permanece viva (sobrevive).
- Em todos os outros casos, a célula morre ou continua morta (superpopulação ou solidão).

Implementação

Em 2011.1, disponibilizei uma implementação que me tomou algo em torno de 6 horas da noite de um sábado

Implementação

Em 2011.1, disponibilizei uma implementação que me tomou algo em torno de 6 horas da noite de um sábado, escutando Ramones e tomando vinho.

Implementação

Em 2011.1, disponibilizei uma implementação que me tomou algo em torno de 6 horas da noite de um sábado, escutando Ramones e tomando vinho.

Ou seja, a implementação contém várias falhas de design

Implementação

Em 2011.1, disponibilizei uma implementação que me tomou algo em torno de 6 horas da noite de um sábado, escutando Ramones e tomando vinho.

Ou seja, a implementação contém várias falhas de design (propositais).

- Vocês precisam melhorar essa implementação
- Usando como guia alguns padrões de projeto

Implementação

Em 2011.1, disponibilizei uma implementação que me tomou algo em torno de 6 horas da noite de um sábado, escutando Ramones e tomando vinho.

Ou seja, a implementação contém várias falhas de design (propositais).

- Vocês precisam melhorar essa implementação
- Usando como guia alguns padrões de projeto

Usaremos a abordagem *Problem Based Learning* para discutirmos sobre padões de projeto.

Falha 01

A decomposição *Model View Controller* da implementação não está adequada.

Falha 01

A decomposição *Model View Controller* da implementação não está adequada. **Desafio:** Reestruturar de acordo com o padrão MVC confome discutido por:

- Yen Ping Ping Shan, *An event-driven model-view-controller framework for Smalltalk*, OOPSLA '89.
- Glenn E. Krasner and Stephen T. Pope, *A cookbook for using the model-view controller user interface paradigm in Smalltalk-80*, Journal of Object-Oriented Programming, Volume 1 Issue 3, Aug./Sept. 1988

Falha 02

Ainda associada a decomposição MVC seguida na implementação atual, as responsabilidades das classes não estão bem distribuídas. Ou seja, os componentes possuem um elevado grau de acoplamento. **Desafio:** Usar os padrões de projeto *Mediator* e *Facade* para reduzir o acoplamento entre as classes, de tal forma que possamos centralizar as colaborações entre os objetos.

Falha 03

Existe um forte acoplamento entre a classe *GameOfLife* e a classe *Statistics*. Basicamente, a instância da classe *Statistics* precisa ser notificada quando uma célula é morta ou ressuscitada. Mas outros objetos poderiam ter interesse neste tipo de notificação, associada a mudança de estado do jogo.

Falha 03

Existe um forte acoplamento entre a classe *GameOfLife* e a classe *Statistics*. Basicamente, a instância da classe *Statistics* precisa ser notificada quando uma célula é morta ou ressuscitada. Mas outros objetos poderiam ter interesse neste tipo de notificação, associada a mudança de estado do jogo. **Desafio:** Usar o padrão de projeto *Observer* para diminuir esse acoplamento, permitindo que instãncias de outras classes possam se registrar nesse tipo de evento.

Falha 04

As regras do jogo (derivação de uma nova geração) estão definidas no método *nextGeneration()* da classe *GameOfLife*. Por outro lado, existem diferentes estratégias de derivação de uma nova geração. Por exemplo, a variação *HighLife* sugere que uma célula morta seja ressuscitada caso tenha seis células vizinhas vivas[1].

[1] Existem várias outras estratégias: http://www.mirekw.com/ca/rullex_life.html

Falha 04

As regras do jogo (derivação de uma nova geração) estão definidas no método *nextGeneration()* da classe *GameOfLife*. Por outro lado, existem diferentes estratégias de derivação de uma nova geração. Por exemplo, a variação *HighLife* sugere que uma célula morta seja ressuscitada caso tenha seis células vizinhas vivas[1].
Desafio: Usar os padrões de projeto *Strategy* e *Template Method* para modularizar as regras do jogo, de tal forma que possamos ter diferentes estratégias de implementação.

[1] Existem várias outras estratégias: http://www.mirekw.com/ca/rullex_life.html

Falha 05

A transição de estado de uma célula está implementada na classe *GameOfLife*, basicamente porque temos apenas dois estados para células (viva / morta). Por outro lado, existem variações do jogo *GameOfLife* que suportam mais estados. Por exemplo, na variação *ImigrationGame*, uma célula ressuscitada tem as mesmas características genéticas das células mais frequentes em relação as suas vizinhas.

Falha 05

A transição de estado de uma célula está implementada na classe *GameOfLife*, basicamente porque temos apenas dois estados para células (viva / morta). Por outro lado, existem variações do jogo *GameOfLife* que suportam mais estados. Por exemplo, na variação *ImigrationGame*, uma célula ressuscitada tem as mesmas características genéticas das células mais frequentes em relação as suas vizinhas. **Desafio:** Usar o padrão *State* para controlar a transição de estados das células, bem como evoluir a aplicação para considera dois tipos de células vivas “o” e “x”.

Falha 06

A implementação não oferece meios para retornar a uma geração anterior. Por outro lado, isso é útil quando desejamos comparar as evoluções entre duas gerações de forma mais precisa. A idéia é implementar essa nova *feature*, de tal forma que o usuário possa retornar *n-gerações*, onde $1 <= n <= 5$. **Sugestão:** Usar o padrão de projeto *Memento*, de tal forma que o estado de um objeto seja encapsulado, facilitando operações de *undo*.

Falha 07

A interface gráfica atual é bastante rudimentar, por outro lado, facilita a portabilidade. Gostaríamos de ter uma implementação do *GameOfLife* usando alguma interface gráfica, baseada em frameworks como GTK+, WX, ou QT. Esses frameworks utilizam padrões de projeto como *Observer* e *Command* para notificar e tratar eventos. **Desafio:** Implementar uma interface gráfica utilizando algum framework existente, mas isso de forma flexível, coexistindo com a interface baseada em *shell*.

Temo 7 falhas e 8 desafios

Objetivo desse trabalho

1. Implemente quatro desafios da sua escolha

Objetivo desse trabalho

1. Implemente quatro desafios da sua escolha
2. . . . mas note que a sua escolha precisa contemplar ou a falha 5 ou a falha 6 (ou ambas), e o seu grupo precisa implementar pelo menos três variações das regras do jogo.
3. A escolha do seu grupo deve ser enviada até o dia 12/02/2013; e cada grupo apresentará as soluções para as falhas escolhidas (nos dias 19 e 21 de fevereiro)

Esse trabalho corresponde a 60% da nota da prova

O que tem que fazer:

1. Reestruturar o código da aplicação base *GameOfLife*, introduzindo as melhorias sugeridas.
2. Apresentar o padrão de projeto
   - Todos os alunos do grupo devem se envolver tanto na implementação quanto nas apresentações.