Lizandra Ottmann de Pádua

Thaísa Alves Andrade

**Relátorio de Projeto**

– Campo Minado –

Goiania, 01 de Dezembro de 2013

**Padrões Aplicados: Strategy, Observer e Iterator**

**Padrão Strategy**

O padrão Strategy é usado para “definir uma família de algoritmos, encapsular cada uma delas e torná-las intercambiáveis. Strategy permite que o algoritmo varie independentemente dos clientes que o utilizam” (como definido no livro do GoF). Em outras palavras, Strategy nos permite configurar uma classe com um de vários comportamentos, utilizando o conceito de OO chamado de composição.

Nesse caso, o Strategy foi aplicado para melhorar a forma como as bombas são distribuídos, uma vez que existem atualmente duas formas diferentes de faze-lo: Aleatória e Por Arquivo (e futuramente podem ser necessárias adicionar outras formas de distribuição).

O padrão sugere que algoritmos parecidos (distribuição de minas) sejam separados  de quem os utiliza (Minas). Para encapsulá-las criamos uma interface (StrategyMinas). Foram implementadas então, as estratégias concretas (SorteiaArquivo e SorteiaAleatorio).

Dessa forma, o cliente que for utilizar as Minas, poderá configurá-lo com a distribuiação desejada. Caso seja necessário incluir uma nova forma distribuição basta implementar a estratégia ou reutilizar outra.

**Padrão Iterator**

O Padrão de Projeto Iterator tem como objetivo encapsular uma iteração.  O Padrão Iterator é definido como: “O Padrão Iterator fornece uma maneira de acessar sequencialmente os elementos de um objeto agregado sem expor a sua representação subjacente”, ou seja, sem a necessidade de saber se ele está usando Matriz ou ArrayList. No nosso caso, Matriz.

Nesse caso, o Iterator foi aplicado no contexto de exibição de todas as Minas do Jogo. Quando o jogador clicar em uma mina e perder o jogo todas as demais devem ser exibida, o mesmo deve ocorrer quando ele ganhar as partidas, são duas situações onde o tabuleiro deve ser varrido em busca da localização de todas as minas.

Optamos por aplicar o Iterator Interno, que segue a idéia de uma classe abstrata implementa o método de percurso e realiza uma operação com cada um dos elementos do conjunto de dados.

  Para começar, foi implementada a interface iterator que define apenas o “esqueleto” que a classe agregada (MinasIterator) deve implementar. Para encapsularmos uma iteração para a Busca das Minas criamos a classe MinasIterator que é o iterador que contém a Matriz do Campo, buscada a partir da classe ObjetoJogo.

O Padrão Iterator fica responsável por toda a tarefa de iteração desse contexto, retirando assim do agregado a responsabilidade de ter loops de busca, dessa forma ele simplifica a interface e a sua implementação deixando a responsabilidade onde deveria estar.

**Padrão Observer**

“Definir uma dependência um para muitos entre objetos, de maneira que quando um objeto muda de estado todos os seus dependentes são notificados e atualizados automaticamente.” (GAMMA, Erich et al. Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos).

Criamos uma classe chamada Objetojogo para facilitar o trafego dos dados pelas classes. Assim, a classe ObjetoJogo usa o conceito de  abstract podendo ter somente uma única instância. Cria-se o objeto da classe(ObjetoJogo) na classe Game que repassa esse objeto a todos que precisarão usar os dados. Sendo assim, ao passar o objeto para alguma classe, a mesma deve criar um variável de classe que recebe esse dado, manipula o mesmo e depois seta o valor alterado no  objeto novamente.

No padrão observer, deve ter em cada classe um método que fique observando se esse objeto sofreria alguma modificação e caso tenha, o mesmo deve alterar esse dado na sua classe de origem. Fizemos melhor, criamos um objeto compartilhado e uma classe que fica observando esse objeto, e quando tiver alteração, o mesmo mostra os novos dados do campo de forma que qualquer que seja a classe que utilize os dados desse objeto, caso esses dados forem alterados, os dados das demais classes serão atualizados instantaneamente.