INSTITUTO FEDERAL DO ESPIRITO SANTO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

RELATÓRIO DE PROJETO

XADREZ

DIEGO PASTI

Serra, 8 de junho de 2014

**Relatório de Projeto**

Esse documento demonstra os principais conceitos e tecnologias utilizadas no projeto Xadrez, ministrada na disciplina de Programação Orientada a Objetos II. Abaixo segue uma breve explicação de como o projeto foi estruturado.

Padrão Arquitetural Modelo-Visão-Controle

Na fase de projeto, o sistema foi dividido seguindo dois estilos arquiteturais sendo eles o estilo Partição e Camadas. Por opção e circunstancias (de tempo e até da estrutura herdada) foi decidido manter apenas uma partição e organizar a estrutura interna em camadas.

Para estrutura das camadas foi utilizado o padrão Modelo-Visão-Controle e suas subdivisões, sendo elas: Camada de Interação Humana, Camada de Controle de Interface, Camada de Domínio do Problema, Camada de Gestão de Tarefas e Camada de Gestão de Dados. Essa estrutura permite que o projeto seja organizado de acordo com os tipos de tarefas, que basicamente são:

|  |  |  |
| --- | --- | --- |
| Camada | Subcamada | Descrição |
| Modelo | Gestão de Tarefas | Camada responsável pelo comportamento das entidades do projeto (ex. casos de usos do sistema). |
| Domínio do Problema | Camada responsável por representar todas as entidades envolvidas no projeto. |
| Gestão de Dados | Camada responsável por gerenciar a manipulação da base de dados. |
| Visão | Interação Humana | Camada responsável por reunir as classes que manipulam a interface. |
| Controle | Controle de Interação | Camada Responsável por ser uma ponte entre a comunicação dos modelos e as visões. |

Conceitos utilizados

Todo o projeto foi desenvolvido utilizando a linguagem Java e essencialmente a orientação à objetos, seguindo os princípios de programação voltada à interface, priorizando o uso de composições a heranças. Nenhuma variável global foi utilizada, os padrões de nomenclatura da linguagem Java foram mantidos e boas práticas de codificação seguindo o padrão clean code também foram seguidos. O resultado foi códigos mais fáceis de cuidar e manter agregando valor ao produto final. Recursos específicos da linguagem tais como Generics (na utilização de listas) e Reflection(na utilização de acesso a arquivos de imagem do projeto) também foram utilizados.

Projeto

A versão atual do sistema não apresenta nenhum problema relacionado a compilação. Foi detectado um erro em tempo de execução em um caso muito especifico e relativamente raro, mas que a solução (até simples por sinal) ainda não pode ser implementada. A lógica do projeto está bem coesa, contudo pontos específicos como tipo de movimento da peça podem ser aprimoradas (e serão) e funcionalidades como desistir, pedir empate e o cadastro dos jogadores estão incompletas. Os diagramas do sistema estão aproximadamente 70% a 80% atualizado, um numero significativo de modificações foram feitas nos últimos dias e tais modificações ainda não foram reproduzidas nos modelos.

Todos os tipos de movimentos de peças xeque e xeque-mate foram implementados, jogadas específicas como El Passant, promoção, roque menor e maior não foram implementadas.

Devido um conjunto de fatores a ferramenta Maven não pode ser configurada e utilizada no projeto, com isso, toda e qualquer solução ou técnica que a exigisse também não pode ser implementada, dentre elas o armazenamento via Hypersonic dos dados e a análise de qualidade de código via Sonar.

O projeto contabiliza a pontuação dos jogadores em cada partida (apesar do placar ter sido suprimido nessa versão – pelo menos em sua forma como era em versões passadas) o que não está implementado ainda é o registro de vitórias, derrotas e empates para os jogadores nem o ranking de pontuação dos mesmos.

O projeto pode ser jogado por 1 ou 2 pessoas, a lógica para as jogadas do computador(Zeus) são simples mas funciona satisfatoriamente (possíveis modificações estão previstas para sua melhoria) contudo não é possível interromper o jogo e continuar posteriormente, a lógica para implementação desta tarefa não foi realizada já o Chat para conversação dos jogadores foi realizada mas não há como (ainda) persistir as mensagens trocadas.

Não foram utilizados testes unitários para controle de qualidade. Os padrões de projeto utilizados são descritos na próxima sessão.

Padrões de Projeto

O projeto foi estruturado seguindo alguns padrões de projeto criativo e adicionalmente alguns padrões comportamentais. Abaixo segue uma breve explicação da utilização de cada um no projeto:

* **Fabrica Abstrata**: Foi utilizado para a instanciação do controle de peças (brancas e pretas) permitindo que a instanciação de cada peça feita seja chamada em apenas um ponto dentro do código, e devido a interface utilizada e os conceitos de polimorfismo podemos alterar uma fabrica de peças pretas ou brancas sem termos que alterar os métodos como são invocados.
* **Protótipo**: Foi utilizado nos slots (casa) do tabuleiro, apesar da sua utilização ter sido parcialmente suprimida devido a não implementação manual da clonagem de atributos complexo como listas. Com ela iríamos instanciar apenas uma vez o slot, e o utilizaríamos como modelo para clonagem dos demais slots que é muito mais eficiente.
* **Observador**: O padrão comportamental observador foi utilizado para notificarmos alterações na matriz de peças do tabuleiro. Para tal cada peça implementa uma interface observável e recebe ao ser instanciada o objeto que irá monitorar e notificar as mudanças do tabuleiro.
* **Estratégia**: O padrão comportamental estratégia foi utilizado para definirmos a lógica de movimentação das peças, visto que este era a grande e praticamente única diferença funcional das peças, dessa forma a metodologia de movimentação de cada peça é um atributo (que pode inclusive, ser alterada em tempo de execução) e implementa uma interface Tipo de Movimento que permite que todos os movimentos sejam invocados da mesma forma. Também foi utilizado para definir a inteligência e a dificuldade do qual o jogador Zeus (maquina) executa, sendo a inteligência as estratégias de jogo (jogadas ofensiva, defensivas e padrão) que ele deve considerar em cada escolha e a dificuldade será analise de jogada com estudo de possíveis jogadas a frente.

Conclusão

Abaixo resumo principais aspectos descritos acima descritos em uma tabela que segue abaixo:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

A utilização dos conceitos de orientação a objetos, desenvolvimento voltado à interface, utilização de composição ao invés de heranças, os padrões de projeto e arquitetural, controle de versão, qualidade de código e modelagem permitiram que o projeto pudesse ser desenvolvido com um foco em melhorar organização e manutenabilidade do código alem do próprio desempenho. A utilização destes conceitos impacta muito no que diz respeito à complexidade da tarefa de desenvolver, mas impacta para melhor visto que não se trata apenas de desenvolver e sim desenvolver com qualidade. O resultado é um produto de alto padrão de qualidade extremamente diferenciado no mercado.

- Nenhuma variável global foi utilizada.

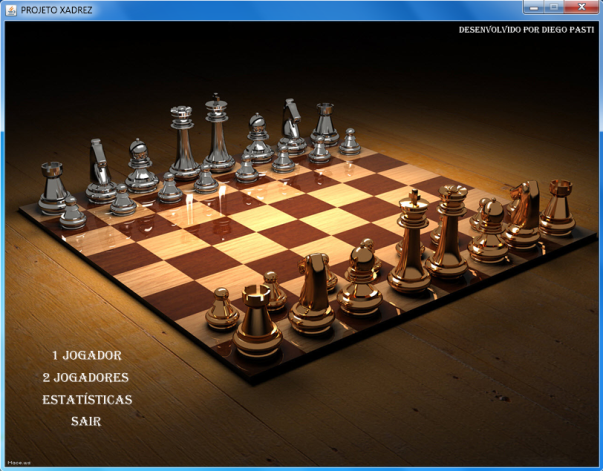
- Utilizam em sua maioria as regras de clean code.

- Utiliza a orientação a objetos.

- Utiliza o padrão fabrica, o padrão protótipo foi utilizado para representar clonar os Slots do tabuleiro, mas não ficou bom.

- Utiliza interface gráfica que melhora muito a jogabilidade.

- Utiliza o Padrão Estratégia no Tipo de Movimento das Peças e na Inteligência do Zeus.



Devido diversos contratempos, e dificuldades encontradas no meio do caminho, mesmo com a ajuda do professor, alguns critérios de avaliação não puderam ser implementados, abaixo relaciono algumas das pendências:

- Falta criar a tela de Desistência, Vitoria e Empate.

- Falta implementar a Promoção do Peão.

- Falta implementar o movimento de Roque Maior e Menor.

- Falta corrigi alguns detalhes nos modelos.

- Falta fazer os testes unitários

- Falta converter o projeto pelo maven

- Falta implementar o armazenamento das partidas em Banco de Dados

- Falta aumentar a abstração na utilização do Padrão estratégia nos tipos de movimento.

O Projeto não pode ser convertido usando o Maven, e isso pode dificultar a analise do projeto, contudo o projeto pode ser executado normalmente usando Eclipse. Desde já agradeço toda a ajuda e atenção.

Projeto

A versão atual do sistema não apresenta nenhum problema relacionado a compilação. Foi detectado um erro em tempo de execução em um caso muito especifico e relativamente raro, mas que ainda continua sem solução (pelo menos até a presente data).

Modelagem

O projeto foi descrito via uml por meio da ferramenta Astah Comunity. Foram criados

Procedimento de Entrega - Verificações - Projeto - O projeto está sem erros de compilação?

Procedimento de Entrega - Envio - Enviar o trabalho pelo moodle (link do projeto + relatório) até 08/06/14.

Procedimento de Entrega - Verificações - Diagrama - O modelo está de acordo com o projeto?

Procedimento de Entrega - Verificações - Projeto - O projeto está sem erros de lógica? Procedimento de Entrega - Verificações - Relatório - O Relatório está explicando os padrões utilizados?

Procedimento de Entrega - Verificações - Diagrama - O Modelo está organizado e dentro dos padrões de qualidade?

Procedimento de Entrega - Verificações - Relatório - O relatório deve conter conclusão sobre o uso dos padrões no auxilio dos padrões de projeto.

Procedimento de Entrega - Verificações - Relatório - O relatório deve conter explicação da utilização da arquitetura mvc.

Procedimento de Entrega - Verificações - Relatório - O relatório deve conter conclusão numerica apartir de dados do sonar.

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o movimento do bispo.

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o movimento do torre.

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o movimento do cavalo.

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o movimento do rei.

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o movimento da rainha.

Implementação - Requisito Funcional - Partida - Pontuação - O sistema deve registrar a vitória (vale 3 pontos).

Implementação - Requisito Funcional - Partida - Pontuação - O sistema deve registrar o empate (vale 1 ponto).

Procedimento de Entrega - Verificações - Relatório - O Relatório está mostrando as tarefas que foram implementada e as que não foram?

Implementação - Requisito Funcional - Peças - Movimentos - Especiais - O sistema deve permitir o movimento roque grande.

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o movimento do peão.

Implementação - Requisito Funcional - Peças - Movimentos - Especiais - O sistema deve permitir o movimento roque menor.

Implementação - Requisito Funcional - Peças - Movimentos - Especiais - O sistema deve permitir o movimento "en passant".

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o xeque.

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o xeque-mate.

Implementação - Requisito Funcional - Peças - Movimentos - Básicos - O sistema deve permitir o movimento do peão (inicial).

Implementação - Requisito não Funcional - O sistema deve utilizar interface gráfica swing.

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve permitir que o usuario pause o jogo para continuar depois.

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve permitir que o usuario desista.

Implementação - Requisito de Implementação - Recursos - O projeto deve utilizar generics e interfaces.

Implementação - Requisito não Funcional - O sistema deve utilizar um chat para as conversas.

Implementação - Requisito não Funcional - O sistema deve utilizar danco de dados para salvar as entidades.

Implementação - Requisito não Funcional - O sistema deve utilizar interface amigável.

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve registrar o numero de vitórias, derrotas e empates.

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve permitir o jogador alterar seus dados (apenas os seus).

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve cadastrar cada jogador (nome,email,apelido).

Implementação - Requisito Funcional - Partida - Pontuação - O sistema deve registrar a derrota (vale 0 pontos).

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve permitir que o usuario peça empate.

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve permitir que o usuario jogue em dupla.

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve permitir que usuario jogue sozinho.

Implementação - Requisito Funcional - Partida - Jogador - O sistema deve ter um ranking de jogadores.

Implementação - Requisito de Implementação - Padrões - Criátivo - O projeto deve utilizar singular (singleton).

Implementação - Requisito de Implementação - Padrões - Criátivo - O projeto deve utilizar protótipo.

Implementação - Requisito de Implementação - Padrões - Criátivo - O projeto deve utilizar método fabrica.

Implementação - Requisito de Implementação - Padrões - Criátivo - O projeto deve utilizar fábrica abstrata.

Implementação - Requisito de Implementação - Recursos - O projeto deve utilizar hypersonic com dao para manipulação de dados.

Procedimento de Entrega - Verificações - Relatório - O relatório deve conter explicação de cada padrão utilizado.

Implementação - Requisito de Implementação - Padrões - Estrutural - O projeto deve utilizar compositor ou outros (se necessário).

Implementação - Requisito de Implementação - Conceitos - O projeto deve utilizar nomenclatura padrão java.

Implementação - Requisito de Implementação - Conceitos - O projeto não deve utilizar variáveis globais.

Implementação - Requisito de Implementação - Recursos - O projeto deve utilizar o maven

Implementação - Requisito de Implementação - Recursos - O projeto deve utilizar reflection.

Implementação - Requisito de Implementação - Conceitos - O projeto deve utilizar o menor numero de classes possíveis.

Implementação - Requisito de Implementação - Conceitos - O projeto deve utilizar orientação a objetos.

Implementação - Requisito de Implementação - Padrões - Arquitetural - O projeto deve utilizar modelo-visão-controle.

Implementação - Requisito de Implementação - Recursos - O projeto deve utilizar testes unitários nas classes de negócios.

Implementação - Requisito de Implementação - Conceitos - O projeto deve utilizar clean code.

Padrão Arquitetural MVC

Na fase de projeto, o sistema foi dividido seguindo dois estilos arquiteturais sendo eles o estilo Partição e Camadas. Por opção e circunstancias (de tempo e até da estrutura herdada) foi decidido manter apenas uma partição e organizar a estrutura interna em camadas.

Para estrutura das camadas foi utilizado o padrão Modelo-Visão-Controle e suas subdivisões, sendo elas: Camada de Interação Humana, Camada de Controle de Interface, Camada de Domínio do Problema, Camada de Gestão de Tarefas e Camada de Gestão de Dados. Essa estrutura permite que o projeto seja organizado de acordo com os tipos de tarefas, que basicamente são:

Conclusão

A utilização dos conceitos de orientação a objetos, desenvolvimento voltado à interface, utilização de composição ao invés de heranças, os padrões de projeto e arquitetural, controle de versão, qualidade de código e modelagem permitiram que o projeto pudesse ser desenvolvido com um foco em melhorar organização e manutenabilidade do código alem do próprio desempenho. A utilização destes conceitos impacta muito no que diz respeito à complexidade da tarefa de desenvolver, mas impacta para melhor visto que não se trata apenas de desenvolver e sim desenvolver com qualidade. O resultado é um produto de alto padrão de qualidade extremamente diferenciado no mercado.