UNIVERSIDADE FEDERAL DA PARAIBA
CENTRO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANNA MYLLENNE ARAÚJO DOS SANTOS
ARTHUR OLIVEIRA RIBEIRO
LUTERO LIMA GOULART
MICHEL DINIZ MEDEIROS

Quiz

Disponível em: https://github.com/Alquimas/ProjetoJava

João Pessoa
2023

Introdução

O projeto proposto consiste no desenvolvimento de um quiz para dois jogadores. O objetivo do jogo é testar o conhecimento dos jogadores sobre o conteúdo estudado na disciplina de Java, por meio de 5 perguntas com diferentes níveis de dificuldade.

Cada pergunta será apresentada com cinco alternativas de respostas, e os jogadores deverão selecionar a opção correta. A ordem de resposta será alternada entre os jogadores, começando pelo jogador 1 e, em seguida, o jogador 2.

A pontuação dos jogadores será determinada pelo número de acertos em cada pergunta, sendo que a pontuação varia de acordo com o nível de dificuldade da pergunta. Perguntas fáceis terão peso de 40 pontos, perguntas de nível intermediário terão peso de 60 pontos, e perguntas difíceis valerão 100 pontos. Em caso de resposta incorreta, o jogador perderá 10, 20 ou 30 pontos, dependendo do nível de dificuldade da pergunta.

Após as 5 rodadas de perguntas para cada jogador, o programa exibirá um resultado final em forma de tabela, contendo o nome de cada jogador e sua pontuação. Além disso, o programa informará quem foi o vencedor ou se houve empate. Ao finalizar o jogo, será perguntado se os jogadores desejam jogar novamente.

Com a implementação desse jogo de quiz em Java, os jogadores poderão testar seus conhecimentos sobre a disciplina, de forma divertida e interativa.

Modelagem do problema

A modelagem proposta para a solução do problema consiste na utilização de várias classes que representam os diferentes elementos do jogo. Os aspectos de orientação a objetos são trabalhados na definição das classes, seus atributos e métodos, bem como nos relacionamentos entre elas.

Classe Jogador:

- Encapsulamento (atributos privados com métodos getter e setter).
- Abstração (representa um jogador do quiz).

A classe Jogador representa um jogador do quiz. Ela armazena o nome do jogador (nomeJ), os pontos acumulados (pontos) e o número de vitórias (wins). A classe possui métodos para obter e definir os valores dos atributos, bem como métodos para incrementar os pontos e as vitórias do jogador.

Classe Pair:

- Encapsulamento (atributos privados com métodos getter e setter).

- Abstração (representa um par de valores).
- Estado (armazena dois valores: x e y).

A classe Pair representa um par de valores. Ela possui dois atributos privados: x e y, que podem ser acessados através dos métodos getter e setter. A classe encapsula os valores em um único objeto, permitindo que sejam passados e manipulados como uma unidade coesa. A classe Pair é utilizada no contexto do jogo para representar uma combinação de dificuldade (x) e número de questões (y) associados a uma determinada pergunta.

Classe BancoDados:

- Encapsulamento (atributos privados com métodos getter e setter).
- Abstração (representa uma coleção de questões do quiz).
- Estado (armazena as questões em uma estrutura de dados).
- Comportamento (métodos para adicionar, remover e obter questões).

A classe BancoDados representa uma coleção de questões do quiz. Ela possui um atributo privado que armazena as questões em uma estrutura de dados, como uma lista, um array ou uma outra estrutura adequada.

Classe ModeloQuestao:

- Encapsulamento (atributos privados com métodos getter e setter).
- Abstração (representa um modelo de questão do quiz).

A classe ModeloQuestao representa um modelo de questão do quiz.

Classe entraNome:

- Encapsulamento (atributos e métodos privados).
- Abstração (representa a entrada do nome de um jogador no quiz).

A classe entraNome é responsável por obter a entrada do nome de um jogador. Ela possui métodos privados que realizam a interação com o usuário para capturar o nome. O objetivo é garantir que o nome seja obtido corretamente, seguindo alguma regra de entrada pré-definida.

Classe entraNumero:

- Encapsulamento (atributos e métodos privados).
- Abstração (representa a entrada de um número no quiz).

A classe entraNumero é responsável por obter a entrada de um número durante o quiz. O objetivo é garantir que o número seja capturado corretamente e esteja dentro de um intervalo válido, se necessário.

Classe entraOpcao:

- Encapsulamento (atributos e métodos privados).

- Abstração (representa a entrada de uma opção no quiz).

A classe entraOpcao é responsável por obter a entrada de uma opção durante o quiz. O objetivo é garantir que a opção seja capturada corretamente e esteja dentro de um intervalo válido, se necessário.

Classe opcao:

- Encapsulamento (atributos privados com métodos getter e setter).
- Abstração (representa uma opção do quiz).

A classe opção representa uma opção do quiz.

Classe scan:

- Encapsulamento (métodos estáticos privados).
- Abstração (representa a entrada de dados pelo usuário).

A classe scan encapsula a lógica de entrada de dados pelo usuário. Essa classe oferece um meio abstrato para realizar a entrada de dados, facilitando a utilização em outras partes do programa.

Classe geraArray:

- Abstração (representa a geração de um array aleatório).

A classe geraArray é responsável por gerar um array com valores aleatórios. A classe utiliza um algoritmo para gerar um array com valores únicos dentro desse intervalo.

Classe geraStack:

- Abstração (representa a geração de uma pilha de pares).

A classe geraStack é responsável por gerar uma pilha de pares (classe Pair) aleatórios. Ela utiliza um conjunto (Set) para garantir que os pares gerados sejam únicos.

Classe getLoss:

- Herança (implementa a interface Pontuacao).
- Polimorfismo (sobrescreve o método getPoints da interface).

A classe getLoss implementa a lógica de pontuação quando o jogador erra uma questão. A classe sobrescreve o método getPoints da interface Pontuacao para retornar os pontos atualizados com base na dificuldade.

Classe getWin:

- Herança (implementa a interface Pontuacao).
- Polimorfismo (sobrescreve o método getPoints da interface).

A classe getWin implementa a lógica de pontuação quando o jogador acerta uma questão. A classe sobrescreve o método getPoints da interface Pontuacao para retornar os pontos atualizados com base na dificuldade.

Interface Pontuacao:

- Abstração (define um contrato para as classes de pontuação).

A interface Pontuacao define o contrato para as classes getWin e getLoss, que implementam a lógica de pontuação do jogo.

Classe quiz:

- Composição (utilização de outras classes).
- Polimorfismo (utilização de interfaces).

A classe quiz é responsável por controlar a lógica do jogo do quiz. Ela possui o método jogarQuiz, que recebe dois objetos Jogador como parâmetros e realiza as etapas necessarias.

Ferramentas utilizadas

Durante o desenvolvimento do projeto foram utilizadas duas IDE's, o Visual Studio Code e o IntelliJ IDEA também foi utilizado o bloco de notas. Foram utilizadas diferentes softwares para que as necessidades e gostos pessoais fossem preservados.

Para facilitar o gerenciamento do código-fonte e a colaboração entre os membros da equipe, foi utilizado o sistema de controle de versão GitHub.

No projeto, organizamos nosso código em pacotes para melhorar a modularidade e organização. A estrutura de pacotes ficou da seguinte forma:

No pacote **"entrada"**, estão as classes responsáveis pela entrada de dados no sistema, permitindo a leitura de nomes, números e opções escolhidas pelo usuário. Essas classes facilitam a interação com o usuário por meio do console.

No pacote **"aux_class"**, estão localizadas classes auxiliares que são utilizadas em outros componentes do sistema. Essas classes fornecem funcionalidades específicas, como a geração de números aleatórios e a manipulação de pares de valores.

O pacote **"funcoes"** abriga classes que implementam funções específicas do sistema. Ele inclui a geração de pilhas de questões aleatórias e a execução do quiz. Essas classes são responsáveis por gerenciar a lógica do jogo.

No pacote **"backbone"**, encontram-se as classes que constituem o núcleo do sistema. Elas são responsáveis pela modelagem das questões e pelo acesso ao banco de dados simulado. Essas classes definem a estrutura e o comportamento das questões do quiz.

Por fim, no pacote **"pontuacao"**, estão as classes responsáveis pelo cálculo da pontuação dos jogadores com base em suas respostas corretas ou incorretas. Essas classes determinam como os pontos são atribuídos aos jogadores durante o jogo.

A convenção utilizada pelo time de desenvolvimento segue algumas práticas comuns de nomenclatura em Java, como nomes de classes começando com letra maiúscula, nomes de métodos e variáveis em camelcase, e uso de comentários para explicar o propósito e a lógica dos trechos de código.

Resultados e considerações finais

Conseguimos implementar com sucesso as principais funcionalidades do jogo permitindo que os jogadores respondam a perguntas e acumulem pontos com base em suas respostas.

Durante o desenvolvimento do projeto tivemos alguns desafios sendo o mais expressivo a dificuldade de definir uma lógica em que fosse possível inserir todos os conceitos de programação orientada a objeto no problema escolhido.

Em relação à aprendizagem da linguagem Java e do paradigma de programação orientada a objetos, consideramos que tivemos uma experiência enriquecedora. Foi possível aplicar os conceitos aprendidos em sala de aula na prática, desenvolvendo um projeto real.

Como considerações finais, gostaríamos de fornecer feedback sobre a disciplina. Acreditamos que o conteúdo abordado foi bastante relevante e bem estruturado. As atividades práticas e o projeto foram fundamentais para consolidar o aprendizado.

Sugerimos que as aulas iniciais que apresentam a linguagem Java sejam mais sucintas e sejam ministradas de forma mais rápida para que os alunos tenham mais tempo para praticar a codificação já no modelo orientado a objetos; também sugerimos que tenham aulas para auxiliar os alunos no aprendizado de uma codificação mais limpa.