**RELATÓRIO PROVA FINAL**

**Ednaldo Rodrigo Brito da Silva Júnior**

**Paulo Henrique**

**Carlos Vinicios Nogueira Costa**

**Gustavo Henrique Soares Feitosa**

**RESUMO**

O trabalho tem como objetivo executar um programa usando conhecimentos adquiridos na matéria estrutura de dados. O programa é um jogo de estratégia baseado em escolhas, usando a linguagem Java, mostrando a jornada do personagem Maxwell passando por cidades e realizando missões específicas em algumas destas, além da interação do personagem principal com o mercador. O trajeto deste personagem começa na cidade de Ubud e termina em Nargumun, onde ele pode se tornar lorde ou servo, dependendo de quantas moedas de transporte possua. Chegando nessa última cidade, o jogo é solucionado. O objetivo principal é executar todo o código, bem como os requisitos e condições apresentados na proposta do trabalho.

**Palavras-chave**: Array. Maxwell. Ubud. Nargumun. Java.

**ABSTRACT**

The objective of this work is to execute a program using knowledge acquired in data structure. The program is a choice-based strategy game, using the Java language, showing Maxwell's journey through cities and performing specific missions in some of them, in addition to the main character's interaction with the merchant. This character's path starts in the city of Ubud and ends in Nargumun, where he can become a lord or a servant, depending on how many transport coins he has. Arriving in this last city, the game is solved. The main objective is to execute all the code, as well as the requirements and conditions presented in the work proposal.

**Keywords**: Array. Maxwell. Ubud. Nargumun. Java.

**1.INTRODUÇÃO**

O projeto foi desenvolvido em equipe, sendo cada membro responsável por determinada parte do projeto, usando linguagem de programação Java, tendo como referências aulas presenciais, discussões em grupos e análises em sites específicos a respeito de estrutura de dados bem como códigos no geral. A importância desse projeto é demonstrar os conceitos de estrutura de dados, tais como arranjos, listas e hashs em convergência com os conhecimentos práticos na linguagem de programação Java, executados no jogo proposto pelo projeto acadêmico.

**2. O CÓDIGO**

O código define uma classe chamada "JogoTransporteJóia" que representa o jogo. Ele declara múltiplas variáveis ​​de membros privados, incluindo vários objetos de cidade (Cidade), variáveis ​​para moedas (moedasTransporte) e limite de poder de joia (limiarJóia), um ArrayList para missões (missaoLista), sinalizadores para missão ativa (missaoAtiva) e cidade da missão (cidadeMissao).

O construtor inicializa as cidades e seus relacionamentos vizinhos.

Os métodos "criarCidades" e "criarVizinhas" são responsáveis ​​por criar as cidades e estabelecer as relações de vizinhança entre elas.

O método "realizarMovimento" trata da movimentação do jogador entre as cidades. Ele recebe a entrada do usuário para as cidades de origem e destino, verifica se o movimento é válido e executa ações específicas com base na cidade de destino. Também permite ao jogador realizar missões em determinadas cidades.

O método "realizarTroca" trata da troca de moedas por poder de joia na cidade de destino. As regras de troca dependem do número de moedas e do número de cidades vizinhas.

Os métodos "realizarMissaoVunese", "realizarMissaoDefalsia" e "realizarMissaoKalb" lidam com missões específicas nas cidades Vunese, Defalsia e Kalb, respectivamente. Esses métodos verificam se o jogador tem moedas suficientes para aceitar a missão e fornecem recompensas em moedas se a missão for aceita.

A parte principal do código cria uma instância da classe "JogoTransporteJóia" e inicia o jogo.

**3. O PROJETO**

O projeto é composto por 3 alunos, Paulo Henrique, Carlos Vinicios, Ednaldo Júnior e Gustavo.

Paulo Henrique fez todas as declarações das variáveis privadas referentes às cidades, criou os construtores para criar as cidades e criar as cidades vizinhas. Logo após, fez os métodos para criar as cidades e os seus vizinhos através de array list.

Carlos Vinicios criou o método para realizar o movimento de Maxwell, verificou se as cidades de origem e destino são válidas, se há uma missão ativa e se o destino corresponde a uma missão específica. Analisou também se o jogador ficou sem moedas de transporte e se excedeu o limiar de joia.

Além disso, criou os métodos para realizar as missões em Vunese, Defalsia e Kalb, exibindo a missão disponível e perguntando ao jogador se aceita ou não. Caso aceite recebe uma recompensa.

Ednaldo ficou com o método realizarTroca() da troca de moedas de transporte por limiar na joia. Dependendo da quantidade de moedas e vizinhas da cidade de origem, diferentes trocas são oferecidas ao jogador.

Utilizou método obterGanhoDePoder(Cidade cidade), que retorna o ganho de poder associado a uma determinada cidade.

Fez uso do método getCidadeByName(String nome), que retorna uma instância da classe Cidade com base no nome fornecido.

Finalmente, Gustavo usou o método iniciar jogo(), que é responsável por iniciar o jogo.

**4. REALIZAÇÃO DO CÓDIGO**

Realizamos o código em conjunto. Quando necessário, realizamos reuniões para discutir o que deveria ser mudado ou melhorado no código.

No início, utilizamos o eclipse para fazer a codificação, mas com o passar do tempo trocamos para o framework Intellij. Para realizar consultas sobre erros, utilizávamos o google ou chatgpt.

**5. JAVA UTILS**

ArrayList: A variável "missaoLista" é uma instância da classe ArrayList<Missao>. Ela é utilizada para armazenar uma lista de missões.

List: A variável "vizinhas" das instâncias da classe Cidade é uma interface List<Cidade>. Ela é utilizada para armazenar as cidades vizinhas de cada cidade.

Arrays.asList: É uma função que retorna uma lista (List<T>) imutável, com base em uma matriz de objetos especificada. É utilizada para inicializar as listas de vizinhas de cada cidade na função criarVizinhas().

HashMap: A classe Cidade tem um atributo nome que serve como chave em um HashMap para buscar uma cidade pelo nome usando o método getCidadeByName.

**6. RESULTADOS**

Os resultados do projeto foram satisfatórios, pois nenhum dos integrantes do grupo havia codificado um game anteriormente, o que certamente não é algo trivial.

O transporte da joia é feito normalmente. Realizamos vários testes para chegar em Nagumun e testamos as missões que não obtivemos o resultado que esperávamos.

**7. CONCLUSÃO**

O projeto foi um grande aprendizado para o grupo, pois, apesar da complexidade para fazer o jogo, fortificamos a nossa base na estrutura de dados baseadas em arrayList.

Infelizmente, não conseguimos concluir com sucesso a codificação das missões. Carlos, que foi o responsável por essa parte, realizou diversos testes, mas não chegou a um código que obtivesse êxito em todos os requisitos propostos.

A implementação de uma interface gráfica, música e efeitos sonoros deixaria o jogo bem mais divertido. No futuro iremos proceder à execução desses elementos em um game.

**8. REFERÊNCIAS**

[1]

The Java Tutorials. Disponível em:

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>. Acesso em: 3 jun. 2023.

Trabalhando com arrays em Java. Disponível em:

[2]

<https://www.devmedia.com.br/trabalhando-com-arrays-em-java/25530>. Acesso em: 3 jun. 2023.

[3]

Arrays. Disponível em: <https://github.com/topics/arrays>. Acesso em: 3 jun. 2023.

[4]

HashMap Java: Trabalhando com Listas key-value. Disponível em: <https://www.devmedia.com.br/hashmap-java-trabalhando-com-listas-key-value/29811>. Acesso em: 3 jun. 2023.