Relatório Final

Prova 2 Estrutura de Dados

Lígia Karolinne Araújo Sousa
Ianna Melissa Osório Sousa
Maria Eduarda Freitas
Ângela Riedel Siqueira

1.0) Resumo

O projeto tem como objetivo criar um jogo de simulação de logística para que os membros da equipe implementem os conhecimentos adquiridos em sala de aula. Esse jogo envolve o viajante Maxwell que vive na cidade e de Ubud, que foi encarregado de transportar uma jóia do poder de sua cidade até a cidade de Nargumun, sem ser alocado como servo. Para que toda essa história fosse executável no console, utilizou-se conceitos de programação orientada a objetos e diversas estruturas de dados, como: arraylist, lista, fila e grafo. Esses recursos possibilitam criar a logística de transporte do personagem entre as cidades e assim concluir a missão.

Por fim, o código gerado permite o jogador sair de Ubud e chegar até Nargumun, enquanto transporta a jóia e recebe prêmios pelas missões aceitas e realizadas.

2.0)Introdução

O projeto foi proposto pelo professor orientador da disciplina Estrutura de Dados como uma forma de avaliação dos conhecimentos adquiridos em sala. Por isso, foi estimulada a criação de um jogo na linguagem de desenvolvimento Java para que os conhecimentos de programação orientado a objetos e, lógicas de programação e estrutura de dados fossem aplicados a um desafio prático.

O jogo tem como protagonista o personagem Maxwell, um viajante que vive na cidade de Ubud. Ele foi encarregado de transportar uma jóia do poder de sua cidade até a cidade de Nargumun. Para cada trajeto que Maxwell percorre, a jóia ganha mais poder e a quantidade de moedas que ele tem sofre alterações de acordo com as suas decisões, o que torna seu transporte cada vez mais difícil. Para compreender essas informações do jogo e seus requisitos, a equipe de desenvolvimento realizou uma reunião para esclarecer o projeto e dividir as responsabilidades entre os integrantes

3.0) Descrição do projeto e da equipe

Neste projeto será criado um jogo com cenários, requisitos,componentes e funcionalidades específicas. Abaixo, será listado todos esses tópicos que compõem

o projeto desenvolvido.

3.1)Cenários

O jogo tem como protagonista o personagem Maxwell, um viajante que vive na cidade de Ubud. Ele foi encarregado de transportar uma jóia do poder de sua cidade até a cidade de Nargumun. Para cada trajeto que Maxwell percorre, a jóia ganha mais poder e a quantidade de moedas que ele tem sofre alterações de acordo com as suas decisões, o que torna seu transporte cada vez mais difícil.

3.2)Componentes e Requisitos

- Personagem: se chama Maxwell e é um viajante residente na cidade de Ubud. Ele será o jogador responsável por transportar a jóia.
- Mercador: Sempre que Maxwell chegar numa cidade, o mercador daquela cidade o aborda e faz três perguntas: 1. Quantas moedas de transporte você tem?; 2. De onde você vem, e para onde você vai?; 3. Deseja trocar suas moedas por limiar na jóia?
- Limiar da Jóia: O limiar de poder da jóia pode ser aumentado ou diminuído, de acordo com a realização das missões.Ele funciona da seguinte forma:Quando o limiar da jóia aumenta, quer dizer que Maxwell consegue carregar a jóia por mais tempo, e quando o limiar da jóia diminui, quer dizer que Maxwell consegue carregar a jóia por menos tempo.
- Poder da jóia:Sempre que a jóia ficar com menos do que zero de poder, instantaneamente ela fica com 0. Além disso, sempre que a jóia ficar com mais do que 7 de poder, Maxwell morre.
- Diálogo com o mercador: em toda cidade que o jogador chegar, haverá um mercador que fará as seguintes perguntas:Quantas moedas de transporte você tem? De onde você vem, e para onde você vai? Deseja trocar suas moedas por limiar na jóia?
- Missões: o jogador ganha moedas e aumenta/diminui o limiar da jóia, assim ajudando a chegar em Nargum, que é o destino final de Maxwell **3.3)**

Funcionalidades e estrutura do código

ATIVIDADE	INTEGRANTE	OBJETIVO	DETALHES

Relatório Final	Lígia	Documentar as etapas e atividades executadas ao longo do processo de desenvolvimento . Ele descreve as metodologias, técnicas e ferramentas utilizadas; Garantir que todos os membros da equipe compreenderam os requisitos e o funcionamento do Jogo;	Detalhes sobre a implementação do software, destacando as principais funcionalidades e recursos desenvolvidos.
Compreensão e Análise dos requisitos	Ângela Riedel; Ianna Melissa; Lígia Karolinne; Maria Eduarda;		Nos post-it, foram detalhados: missões, o que o Maxwell pode ou não fazer; diferença entre limiar da jóia e poder da jóia;
Classe MissaoDefalsia	Lígia Karolinne	Envolver tudo relacionado à realização da missão em Defalsia, como: prêmio por aceitar e finalizar a missão, além da descrição específica dessa missão. Além da atualização dos bens do personagem à medida que ele participa da missão.	**Atributos**: nome, descrição, statusConcluida, aceitou, finalizou, emAndamento, premioMoedaAceitar , PremioLimiar, PremioMoedaFinaliz ar; **Métodos:** oferecerMissao(Personagem maxwell); popularMissao(); podeRealizarMissao(Persoangem

			maxwell);
			finalizarMissao();
			atualizarBensAceitar Missao();
			Métodos getters e setters;

Documentação	Ianna Melissa Ângela Riedel	Detalhar e comentar o código fonte, explicando a lógica e as estruturas utilizadas para criar o programa do jogo	

Classe MissaoKIng do mKalb	Lígia Karolinne	Envolver tudo relacionado à realização da missão em Kingdom Kalb, como: prêmio por aceitar e finalizar a missão, além da descrição específica dessa missão. Além da atualização dos bens(ganho de moedas e aumento do limiar da jóia) do personagem à medida que ele participa da missão.	**Atributos**:
			nome,
			descrição,
			statusConcluida,
			aceitou,
			finalizou,
			emAndamento,
			premioMoedaAceitar ,
			PremioLimiar,
			PremioMoedaFinaliz ar;
			
			Métodos:
			oferecerMissao(Pers onagem maxwell);
			popularMissao();
			podeRealizarMissao(Persoangem maxwell);
			finalizarMissao();
			atualizarBensAceitar Missao();
			Métodos getters e setters;

Classe MissaoVunese	Lígia Karolinne	Envolver tudo relacionado à realização da missão em Vunese, como: prêmio por aceitar e	**Atributos**: nome, descrição, statusConcluida, aceitou, finalizou, emAndamento, premioMoedaAceit

		finalizar a missão, além	ar ,

		da descrição específica dessa missão. Além da atualização dos bens(ganho de moedas e diminuição do limiar da joia do personagem à medida que ele participa da missão	PremioLimiar, PremioMoedaFinaliz ar;
			
			Métodos:
			oferecerMissao(Pers onagem maxwell);
			popularMissao();
			podeRealizarMissao(Persoangem maxwell);
			finalizarMissao();
			atualizarBensAceitar Missao();
			Métodos getters e setters;

Classe Personagem	Ianna Melissa	Representar o personagem do jogo, além de criar toda uma estrutura que posisbilita a criação de objetos do tipo Personagem , com determinadas características e funcionalidade .	**Atributos:** moedas; poderJoia; limiarJoia; estaVivo; cidadeAtual; **Métodos:** getters e setters; checaStatusPersona gem(); → Retorna se o atributo estaVivo é verdadeiro(personag em vivo) ou falso(jogador morto)

Classe DialogoMercad or	Ângela Riedel	Criar os atributos e métodos necessários para criar a dinamicidade no diálogo entre o mercador e o jogador. Criar método responsável por atualizar os bens de acordo com as repostas que o jogador	**Atributos:** moedasMaxwell, moedas, distancia, moedasAtual, querTrocar; **Métodos:** atualizarBens()--> esse método é feito de acordo com as 3 perguntas que o mercador faz ao jogador, assim que

		fornece ao mercador	ele chega em uma cidade nova.

Classe Transporte	Maria Eduarda Freitas	Criar a lógica da logística de transporte, a partir de métodos que especificam o diálogo que precisa ter para o transporte acontecer, além da atualização dos bens do jogador durante a viagem.	Instância-se as classes: mapa, MIssaoKingdomKalb MissaoDefalsia, MissaoVunese; Métodos: iniciarTransporte() → inicia-se o transporte perguntando para onde o jogador quer ir e puxando o nome das cidades viizinhas; realizaTransporte()-- > engloba todos os métodos necessários para o transporte acontecer; atualizaBens()--> o limiar da joia e o número de moedas é atualizado de acordo com a cidade que o personagem está; exibedialogo()--> tem o diálogo detalhado entre o jogador e o mercado e , para isso alguns métodos da classe DialogoMercador são chamados; faz desenho()--> cria um desenho que ilustra a viagem do jogador de uma

			cidade para outra; aguardar() → serve para parar a execução do programa por alguns segundos;

Classe Mapa	Maria Eduarda Freitas	Criar a ligação das cidades através da implementação de listas. Criar estrutura de dados para implementar as fronteiras do mapa. Utiliza-se fila	Cria-se a lista vizinhanca;
			Métodos:
			addCidade, addVizinho, calcularDistancia() → calcula a distância entre duas cidades num mapa;
			getVizinhos();
			getCidades()--> cria-se uma lista cidades e o método a retorna com todas as cidades contidas no mapa vizinhanca;
			cria-se uma fila para fazer a busca no mapa “vizinhanca” para descobrir a distância entre duas cidades ;
			printMapa();
			popularMapa();
			saoVizinhas()→verifi ca se as cidades atual e de destino são vizinhas;

Classe Jogo	Maria Eduarda Freitas		Objetos: Mapa, Transporte, Personagem. Métodos: construtor classe Jogo; exibeMenu() → exibe o menu principal com as
			opções de ver todas as cidades, realizar transporte e sair do jogo; statusJogo() → verifica se o Jogo tá ativo e se o personagem está vivo; iniciar() → enquanto o jog estiver ativo,o menu principal é exibido e o status do jogo é atualizado;
Classe Main	Maria Eduarda Freitas		classes Mapa, Transporte e Jogo; O método iniciar() da classe Jogo é chamado para iniciar o jogo.

4.0)Metodologia

Iniciar o jogo Criação de instâncias das

Para a realização do projeto, a equipe de desenvolvimento utilizou a metodologia em cascata, ou seja, a análise, a implementação, os testes e a documentação foram

feitas em sequência. Por exemplo, primeiro buscou-se entender os requisitos do jogo, em segundo foram desenvolvidas as classes mais básicas; em terceiro, as demais foram feitas e por último, o relatório foi realizado quando o código já estava finalizado.

Além disso, o grupo também utilizou na criação do código: uma IDE(Integrated Development Environment) chamada Eclipse para escrever, executar e testar o código; o GitHub foi utilizado para compartilhar o projeto e trabalhar nele de forma colaborativa.

5.0)Estrutura de Dados

No decorrer da criação do código do jogo, foram utilizadas duas importantes estruturas de dados: tabela de hash e fila, através do uso de listas.A primeira foi implementada usando um HashMap(armazena pares chave-valor, onde cada chave - cidade - é única e mapeada para um valor correspondente - cidades vizinhas.) em Java com o objetivo de armazenar as cidades e suas respectivas vizinhanças, além de calcular a distância entre duas cidades no mapa. Observe a implementação dessa estrutura nas Figuras 1 e 2.

```
public class Mapa {
    private Map<String, List<String>> vizinhanca;

    public Mapa() {
        vizinhanca = new HashMap<>();
        popularMapa();
    }

    public void addCidade(String cidade) {
        vizinhanca.put(cidade, new ArrayList<>());
    }


    public void addVizinho(String cidade, String vizinho){
        List<String> vizinhos = vizinhanca.get(cidade);
        //Se Aymar League é vizinha de Bun, Bun é vizinha de
Aymar League
        vizinhos.add(vizinho);
    }

    public int calcularDistancia(String cidadeOrigem, String
cidadeDestino) {
        if (!vizinhanca.containsKey(cidadeOrigem) ||
!vizinhanca.containsKey(cidadeDestino)) {
```

```
            // Verificar se as cidades existem no mapa
            return -1;
        }

        if (cidadeOrigem.equals(cidadeDestino)) {
            // As cidades são iguais, distância é zero
            return 0;
        }
```

Figura 2: implementação de um hashmap.

A segunda estrutura de dados, a fila, é implementada na classe Mapa e tem como objetivo fazer uma busca em largura na estrutura de dados “vizinhanca” para encontrar a menor distância entre duas cidades: cidadeOrigem e cidadeDestino. É válido ressaltar que a distância entre as cidades é salva no mapa “vizinhanca”. Ela é utilizada para percorrer o grafo em largura, visitando primeiro as cidades mais próximas da cidadeOrigem. Observe a implementação deste código nas imagens abaixo.

```
Set<String> visitadas = new HashSet<>();
        Queue<String> fila = new LinkedList<>();
        Map<String, Integer> distancia = new HashMap<>();

        fila.offer(cidadeOrigem);
        distancia.put(cidadeOrigem, 0);

        while (!fila.isEmpty()) {
            String cidadeAtual = fila.poll();
            visitadas.add(cidadeAtual);

            List<String> vizinhas = vizinhanca.get(cidadeAtual);

            for (String vizinha : vizinhas) {
                if (!visitadas.contains(vizinha)) {
                    fila.offer(vizinha);
                    distancia.put(vizinha, distancia.get(cidadeAtual) +
1);

                    if (vizinha.equals(cidadeDestino)) {
                        return distancia.get(vizinha);
                    }
                }
            }
        }
```

6.0)Resultados

Durante a construção do código, foram feitos vários testes de pequenas partes do código para saber se estavam funcionando corretamente. Ademais, para garantir uma melhor qualidade do jogo desenvolvido, foram realizados mais testes após a finalização da versão final do programa. Em decorrência disso, constatamos que o código funciona de acordo com o planejado e que os métodos são executados no devido momento e as respostas no console estão adequadas. Logo,a equipe desenvolvedora considera o desempenho do programa como muito bom e compatível com os recursos e conhecimentos utilizados.

7.0)Conclusões

Com o fim do projeto, a equipe entra em consenso sobre ter construído um jogo bom com os recursos e conhecimentos que foram oferecidos.Entretanto, inicialmente foi difícil definir como seria desenvolvida a estrutura do código, quais classes seriam criadas e o que elas fariam e como fariam. Por isso, a experiência de criar um sistema do zero é um grande aprendizado para os integrantes da equipe e serviu para mostrar as habilidades que cada um ainda precisa aperfeiçoar.

Além disso, ao analisar o processo como todo, é nítido que o uso de estrutura de dados, como fila e grafo, facilitam a construção da lógica e economizam tempo na implementação do jogo. Logo,alguns assuntos mostrados na disciplina de Estrutura de Dados foram bem aplicados ao projeto e isso mostrou aos alunos a importância de se aprender tais recursos para melhorar a prática de codificação de cada um.

8.0)Referências

Goodrich, Michael T. Estrutura de Dados e Algoritmos em Java. 4ª ed. Porto Alegre: Bookman, 2014.