Nome: Gustavo Molino Teixeira Alves RA: 247144
Nome: Rodrigo Botelho Zuiani RA: 245244

O grupo decidiu por uma estrutura contendo 11 classes(AppWumpus, Buraco, Caverna, Componente, Controle, Heroi, Montador, Ouro, Sala, Toolkit e Wumpus).
A classe Componente define métodos gerais para todos os componentes utilizando herança e sobrecarga de métodos (Heroi, Buraco, Ouro e Wumpus). Desse modo, na classe Montador, responsável por verificar se a caverna é válida e montá-la, utiliza-se amarração pois, declara-se apenas um Componente “comp_atual”, que é instanciado baseando-se em qual dos componentes é necessário naquele momento. Nota-se que assim, podemos adicionar outros componentes para melhorar o jogo utilizando códigos já escritos em componentes.
código:

```
Componente comp_atual = null;
if(!tipoComp.equals("_")) {
    if (tipoComp.equals("W")) {
        n_wumpus++;
        comp_atual = new Wumpus(x, y, 'W');
    }
    else if (tipoComp.equals("B")) {
        n_buracos++;
        comp_atual = new Buraco(x, y, 'B');
    }
    else if (tipoComp.equals("O")) {
        n_ouro++;
        comp_atual = new Ouro(x, y, 'O');
    }
    else {
        n_heroi++;
        if(x != 0 || y != 0){
            System.out.println("Caverna invalida");
            System.exit(0);
        }
        comp_atual = new Heroi(x, y, 'P');
        controle.conectaHeroi(comp_atual);
    }
    comp_atual.conectaCaverna(caverna);
    if(!comp_atual.solicitaSala()){
        System.out.println("Caverna invalida");
        return false;
    }
    comp_atual.geraEfeito();
```

Na classe caverna, cria-se uma array de salas, que irão representar cada uma das salas da caverna e seus componentes internos, que são criados no montador e conectados às salas pela caverna. Além de criar uma matriz que representa o efeito visual que aparecerá no console para o usuário.

Na classe Sala, utiliza-se polimorfismo, pois declara-se uma array de componentes, para receber todos os componentes que podem estar presentes naquela sala ao mesmo tempo, como o Ouro e o Herói ou o Herói e o Wumpus. e posteriormente utilizamos os métodos dos componentes especificamente.

Código:

```
private Componente[] compSala;
```

A classe controle é responsável pela movimentação do herói e pela interação deste com os itens da caverna, além de controlar a interface do herói, com o score, nome, etc. Ela declara um Componente herói, que será cedido pelo montador e instanciado na classe herói e verificará, pela caverna, o que há em cada sala e realiza toda a interação com o herói. Como lutar contra o wumpus, obter o ouro e sentir a brisa e o fedor.

Código ações do herói:

```
public void acao(char acao) {
    int xDestino = heroi.coordenadaX;
    int yDestino = heroi.coordenadaY;
    if (acao == 'w') {
        yDestino--;
        Movimenta(xDestino, yDestino);
    } else if (acao == 's') {
        yDestino++;
        Movimenta(xDestino, yDestino);
    } else if (acao == 'd') {
        xDestino++;
        Movimenta(xDestino, yDestino);
    } else if (acao == 'a') {
        xDestino--;
        Movimenta(xDestino, yDestino);
    } else if (acao == 'k') {
        System.out.println("Você equipou sua flecha, boa sorte!!");
        heroi.equipaFlecha();
    } else if (acao == 'c') {
        if (heroi.caverna.getComponenteSala(xDestino, yDestino) ==
'O') {
            System.out.println("Parabéns aventureiro, você recolheu
o ouro. Agora fuja desta caverna!!");
            heroi.capturaOuro();
        }
```

código de interação com o Wumpus e com a brisa e fedor:

Duelo contra wumpus:

```
if (heroi.caverna.getComponenteSala(xDestino, yDestino) == 'W') {
    boolean matou = false;
    if (heroi.getflechaEquipada()) {
        matou = heroi.usaFlecha();
        alteraPontuacao(-100);
    }
    if (matou) {
        alteraPontuacao(500);
```

```
        System.out.println("Incrível aventureiro, você matou o
monstro Wumpus!!");

    } else {
        alteraPontuacao(-1000);
        status = 'L';
    }
}
```

Brisa e fedor:

```
if(heroi.caverna.brisa(xDestino, yDestino)){
    System.out.println("Você sente uma brisa nesta sala!!");
}
if(heroi.caverna.fedor(xDestino, yDestino)){
    System.out.println("Você sente um fedor nesta sala!!");
```