Programando um jogo de RPG

Criação do roteiro:

1. Defina 2 personagens possíveis para o jogo. Defina o nome, idade, a história do personagem, um poder especial para cada e uma característica para ambos que possa ser representada por um número, como a quantidade de vida, energia, dinheiro, entre outros. O poder especial deve ser capaz de alterar também o valor dessa característica.

- Característica: Vida, Mana e dano

**Personagem 1:**

- Nome: **Rodan**

- Tipo: **Mago**

- Poder especial:

- **Tempestade de flechas**, aumenta a porcentagem de danos no obstáculo (custo: 30% de mana)

- **Auto Regeneração**, aumenta a porcentagem de vida do personagem (custo: 50% de mana)

- História:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

- Valor inicial da característica:

- Vida -> 70

- Mana -> 0

**Personagem 2:**

- Nome: **Dofer**

- Tipo: **Cavaleiro**

- Poder especial:

- **Espada flamejante**, aumenta a porcentagem de danos no obstáculo (custo: 65% de mana)

- **Armadura impenetrável,** personagem não recebe dano por 2 turnos mas força de ataque diminui (custo: 80% de mana)

História: Respeitado cavaleiro da Guarda Real de Javacity, Dofer, estava mais um dia na guarda das muralhas da cidade quando, com um som ensurdecedor, uma luz verde em forma de garra, rapidamente se materializando, emana das sombras e mata seu irmão e parceiro de armas Mikelos. Ao vê-lo Dofer utiliza seu famoso ataque espada flamejante e parte para cima da garra que, para sua surpresa, superou sua extrema velocidade e ataca-o antes que pudesse sequer arranhá-la, o cavaleiro caiu paralisado e, fechando os olhos, viu por alguns instantes a garra voltando para as sombras e espairecer lentamente. Semanas após o ocorrido, Dofer encontra-se no hospital e, ao abrir seus olhos, começa a sentir uma dor latente em seu peito, ele olha para baixo e vê um buraco onde seu coração deveria estar. Dofer gritou e gritou até que o médico de plantão chega tentando acalmá-lo e explicá-lo um pouco mais sobre o seu estado ao chegar no hospital. O médico contou que em seu peito havia um buraco de aproximadamente um palmo de largura, sob intensas hemorragias, seu coração estava batendo exposto, quase como se quisesse desprender-se do corpo do cavaleiro. Além disso, em seu coração havia uma substância verde, acídica, corroendo sua carne, que o doutor nunca vira antes, mas que suspeitava ser algum tipo de magia negra. Infelizmente, a única solução para salvá-lo foi retirar seu coração infectado e realizar um tratamento intensivo de materialização de mana — energia mágica que todos os seres humanos compartilham, ainda que em níveis diferentes, e que pode ser transferida para outro corpo —, com os magos renomados da cidade tentando um procedimento arriscado de criar um proto-coração com sua já existente mana e com a energia doada pelos humildes habitantes de Javacity, mas apesar do êxito no tratamento, o médico o avisou que ele nunca mais teria o mesmo vigor, as mesmas forças e a mesma mana em seu corpo que antes. Desde esse dia, Dofer treinou ininterruptamente por dois anos, em dias quentes e noites chuvosas, ultrapassando a fadiga de cada músculo em seu corpo para partir em busca do assassino de seu irmão. Após cinco anos de uma jornada incansável por respostas, o cavaleiro ouviu boatos de que ao Sul de Javacity, nos subterrâneos da Floresta Sombria de Sonfejer habitava um mago negro que realizava experimentos que emanavam longínquas e estranhas luzes verdes, o herói sentiu sua espinha arrepiar, ele já sabia o que fazer.

- Valor inicial da característica:

* Vida -> 100
* Mana -> 0

2. Crie uma introdução para o jogo. Diga onde o/a personagem está e qual o objetivo dele/a no jogo. Escolha somente um objetivo.

**- Local:** Javacity

**- Objetivo:** Derrotar o mago negro Napda e seus subordinados.

**- Itens** (quais itens estão presentes nesse local que poderiam integrar o inventário do/da personagem?):

- Poção de mana e Poção de vida;

- Armas diversas com atributos diversos;

- Poção para aumentar força de ataque comum

**- Contexto** (como seu personagem foi para esse local? Por que ele gostaria de alcançar esse objetivo?):

- Ele nasceu em Javacity

- Dofer: estava treinando e inicia sua jornada de lá

- Rodan: […]

- Dofer: para vingar a morte de seu irmão

- Rodan: […]

3. Agora vamos criar obstáculos para o/a personagem. Pode ser algum inimigo, alguma dificuldade, enfim, qualquer coisa que o/a personagem tenha que se superar aos poucos para alcançar o objetivo. A interação com esses obstáculos deve diminuir ou aumentar o valor da característica definida para o/a personagem. Crie 3 obstáculos distintos:

- Obstáculo 1:  **Napda** (Vilão)

- O que o personagem deve fazer para superar o obstáculo?

**Matá-lo**

- Como esse obstáculo aumenta ou diminui o valor da característica definida para o/a personagem?

**Causando dano a sua vida**

- Obstáculo 2: **Flegoman** (vilão)

- O que o personagem deve fazer para superar o obstáculo?

**Matá-lo**

- Como esse obstáculo aumenta ou diminui o valor da característica definida para

o/a personagem?

**Causando dano a vida do personagem**

- Obstáculo 3: **Dataretan (vilão)**

- O que o personagem deve fazer para superar o obstáculo?

**Matá-lo**

- Como esse obstáculo aumenta ou diminui o valor da característica definida para

o/a personagem?

**Causando dano a vida do personagem**

**Caminho 1:** o/a personagem pode ou não superar os obstáculos! Se o valor da característica passar de determinado limiar (pode ser máximo ou mínimo), o/a personagem não superou os obstáculos e perdeu o jogo. Alguns exemplos:

a. Em uma batalha entre o/a personagem e um inimigo, uma característica de “vida” pode ir diminuindo. Os diferentes inimigos podem ter diferentes poderes, por isso causam danos diferentes na vida do/da personagem (como no exemplo do Bomberman que fizemos em sala).

Escreva algumas palavras contando o que aconteceu:

Game Over:

O (nome do obstáculo) te matou

**Caminho 2:** Superado os obstáculos o/a personagem está livre para cumprir o

objetivo. Escreva algumas palavras de conclusão do objetivo e finalização do jogo.

Parabéns, (herói escolhido) salvou todos os cidadãos de Java City, Você é um herói

Roteiro preparado! Vamos programar?

Programando o jogo (leia tudo antes de implementar!):

Criando os/as personagens:

1. Crie uma interface chamada Personagem. Ele deverá conter os métodos obrigatórios para manipulação de um personagem do jogo. São eles pelo menos: a. contarHistória(): para imprimir na tela a história do/da personagem b. poderEspecial(): para aplicar o poder especial definido para o/a personagem. c. adicionarItem(): para adicionar um item ao inventário do/da personagem. As assinaturas desses métodos não precisam ser as descritas acima, elas podem ter um ou mais parâmetros conforme a necessidade do seu jogo. É também permitido adicionar mais métodos na interface conforme a necessidade do seu jogo.

2. Agora implemente as classes necessárias para criação dos dois Personagens. As boas práticas de programação nos dizem que deveríamos ter uma classe mãe para os métodos e atributos comuns e classes filhas para implementar métodos e atributos especializados de cada personagem. Não esqueça de colocar o modificar final nas classes que não poderão mais ser herdadas (personagem final). Ambos os personagens deveriam ter pelo menos os atributos nome, idade, a característica escolhida no roteiro e um inventário para armazenar itens. Escolha os tipos apropriados para cada atributo. Dou a dica de que o inventário precisa ser alguma classe do grupo de Coleções. Não esqueça de implementar os métodos nas classes apropriadas. Ou seja, os métodos utilizados da mesma maneira por ambos os Personagens deveriam ser implementados na classe mãe e os métodos específicos deveriam ser implementados nas classes filhas e estarem na classe mãe como

abstratos. Não esqueça de dizer que a classe mãe implementa a interface correspondente.

3. Antes da implementação, **desenhe um esquemático em UML** com a interface, a classe mãe, as classes filhas e os modificadores de acesso apropriados. Utilize o modificador de acesso final nas classes que não terão mais herdeiros.

Criando os obstáculos:

4. Do mesmo modo que criamos uma interface, uma classe mãe para o/a personagem e classes filhas para especificar as características dos/das personagens, vamos fazer o mesmo com os obstáculos. Crie uma interface Obstáculo (ou um nome mais apropriado se preferir) com um método obrigatório:

a. interageComPersonagem(Personagem personagem): esse método receberá como argumento um objeto Personagem e implementará a interação que o obstáculo terá com o personagem. Esse método imprime uma mensagem na tela explicando o que está fazendo e aumenta ou diminui a característica definida para o personagem. Por exemplo, se o obstáculo for um inimigo em uma luta, esse método será responsável por diminuir a vida do personagem na quantidade de unidades escolhida, e assim por diante. Outros exemplos: i. Em uma batalha entre o/a personagem e um inimigo, uma

característica de “vida” pode ir diminuindo. Os diferentes inimigos

podem ter diferentes poderes, por isso causam danos diferentes na

vida do/da personagem (como no exemplo do Bomberman que

fizemos em sala).

ii. O/A personagem pode tomar poções que aumentam ou diminuem sua vida em diferentes proporções.

iii. O/A personagem pode gastar uma característica “dinheiro” a cada rodada até que ele não tenha mais dinheiro para gastar.

As assinaturas desse método precisa necessariamente ter como argumento um objeto da classe Personagem, mas podem ter mais argumentos se isso fizer sentido para o seu jogo. É também permitido adicionar mais métodos na interface conforme a necessidade do seu jogo.

5. Agora, implemente a classe mãe e classes filhas para criação dos três Obstáculos. Não esqueça de colocar os atributos e métodos comuns na classe mãe e os específicos nas classes filhas. Se o obstáculo for um inimigo a ser derrotado, um atributo pode ser a “vida” desse inimigo. Se o obstáculo for algo a ser comprado, o atributo pode ser o “preço” que esse algo custa, por exemplo. Não esqueça de dizer que a classe mãe implementa a interface correspondente. Não esqueça de colocar o modificar final nas classes que não poderão mais ser herdadas (obstáculo final).

6. Antes da implementação, **desenhe um esquemático em UML** com a interface, a classe mãe, as classes filhas e os modificadores de acesso apropriados. Utilize o modificador de acesso final nas classes que não terão mais herdeiros.

Implementando o jogo:

7. Com as classes base criadas, podemos programar o jogo! No início do jogo, o usuário do programa deve escolher o/a personagem com o/a qual vai jogar. Instancie cada um dos/das personagens e exiba na tela uma mensagem de boas-vindas, o nome, característica, idade e história dos/das personagens **utilizando o método ContarHistoria()** e pergunte ao usuário com qual personagem ele/a quer jogar. Esse/a será o/a personagem principal do jogo.

8. Uma vez escolhido o/a personagem, apresente na tela a história elaborada para a introdução, apresentando o local, objetivo e contexto. Nessa parte do jogo, o/a personagem pode adicionar alguns itens no inventário. Exiba quais itens estão disponíveis e o jogador pode escolher quais gostaria de adicionar no inventário **usando o método adicionarItem().**

9. Pronto! Agora chegamos no ápice do jogo. Programe interações entre o/a personagem e os obstáculos, onde cada interação aumenta ou diminui o valor da característica do/da personagem pensado anteriormente. **Intercale os métodos PoderEspecial()** do/da personagem quando o/a personagem estiver jogando **e InterageComPersonagem()** quando for a vez do obstáculo. **Dicas: estruturas de repetição são muito úteis para programar essa parte. Estruturas condicionais também podem te ajudar.**

10. Finalize o jogo. O/A personagem superou os obstáculos? Exiba a mensagem apropriada para cada possível final.

Dicas:

1. Números aleatórios são úteis para simular jogadas de dados e comportamento de inimigos e obstáculos durante o jogo. Eles deixam o seu jogo mais dinâmico e cada vez que o usuário jogá-lo pode acabar tendo um resultado diferente.

2. Não esqueça de tratar possíveis exceções no seu jogo. É recomendado o tratamento de exceção toda vez que o usuário der alguma entrada, garantindo que o jogo não irá ser fechado caso o usuário coloque uma entrada que provoque um erro.

3. Esse post faz uma explicação simples e bacana do que estamos fazendo neste projeto. Ele usa a linguagem de programação Java e está em inglês, mas você pode traduzir os diferentes trechos da página utilizando o Google Translate: https://medium.com/@franksands/why-is-object-oriented-programming-useful-with-an -rpg-example-javascript-and-java-670e2d1c5505