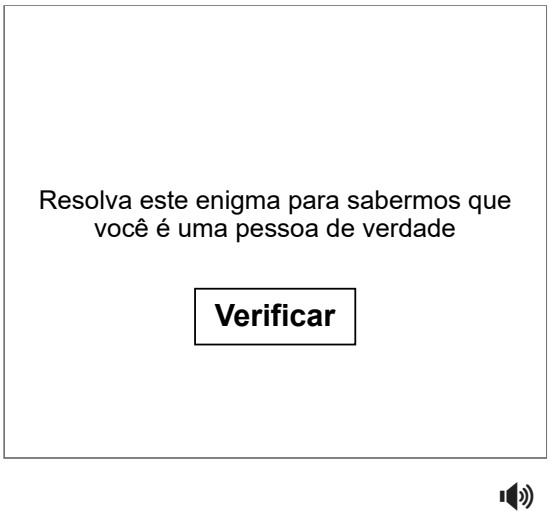
Exercício – Modelar e realizar scaffold do seu jogo

10 minutos

Verifique sua conta



Um jogo não é diferente do *sistema de faturas* mencionado em uma unidade anterior. Ele ainda tem as mesmas partes no mundo da OOP (programação orientada a objeto). Essas partes são objetos, dados e comportamento. Assim como fizemos com o sistema de faturas, você pode *modelar* um jogo como pedra, papel e tesoura descrevendo primeiro o domínio. Em seguida, tente listar o que é o quê. A modelagem do jogo é o que você vai fazer em seguida. Você também escreverá um código que poderá ser criado posteriormente.

Analisar pedra, papel e tesoura para a modelagem OOP

Este exercício é um exercício de modelagem. Você receberá uma descrição do domínio do problema. Em seguida, deverá retirar as palavras-chave importantes do texto e organizá-las em uma tabela.

Dica

Se você precisar de uma revisão sobre como modelar, confira a seção "Localizar objetos, dados e comportamento" **na segunda unidade**. Lembre-se de que a modelagem começa com perguntas: *quem interage com quem* ou *quem faz o quê com quem*.

Descrição do problema

Pedra, papel e tesoura é um jogo com dois participantes. O jogo tem rodadas. Em cada rodada, um participante escolhe um símbolo de pedra, papel ou tesoura, e o outro participante faz o mesmo. O vencedor da rodada é determinado pela comparação dos símbolos escolhidos. As regras do jogo estabelecem que pedra ganha de tesoura, tesoura vence (corta) papel e papel vence (embrulha) pedra. O vencedor da rodada recebe um ponto. O jogo continua pela quantidade de rodadas que os participantes combinarem. O vencedor é o participante com o maior número de pontos.

Modelar o jogo

1. Copie e cole o texto anterior em um documento. Realce as palavras-chave importantes colocando-as em negrito ou itálico.

   > **Dica**
   >
   > Dedique alguns minutos tentando sublinhar o que você considera palavras-chave importantes. Depois de concluir, role para baixo até o texto que realça as partes importantes.

   Este é o texto da descrição do problema que realça:

   Pedra, papel e tesoura é um *jogo* com dois participantes. O jogo tem *rodadas*. Em cada rodada, um *participante* escolhe um *símbolo* de *pedra*, *papel* ou *tesoura*, e o outro *participante* faz o mesmo. O *vencedor* da rodada é determinado pela *comparação* dos símbolos escolhidos. As *regras* do jogo estabelecem que pedra ganha de tesoura, tesoura vence (corta) papel e papel vence (embrulha) pedra. O vencedor da rodada recebe um *ponto*. O jogo continua pela quantidade de rodadas que os participantes combinarem. O vencedor é o participante com o maior número de pontos.

2. Em seguida, crie uma tabela com as colunas `Phase`, `Actor`, `Behavior` e `Data`. Organize as palavras realçadas onde você acha que elas devem ser colocadas.

   > **Dica**
   >
   > Dedique alguns minutos pensando nisso e role para baixo depois de pensar um pouco.

   Esta é a aparência da tabela resultante:

   Expandir a tabela

Fase	Ator	Comportamento	Dados
Entrada	Participante	Escolhe um símbolo	Símbolo salvo como *escolha* em Participant(choice)

Fase	Ator	Comportamento	Dados
Processando	GameRound	Compara a escolha tendo em mente as regras do jogo	*Resultado* inspecionado
Processando	GameRound	Recebe pontos com base no valor do resultado	*Pontos* adicionados ao Participant(point) vencedor
Processando	Game	Verificar resposta continuar	A resposta é true, continuar; caso contrário, sair
Saída	Game	Nova crédito de rodada do jogo ou fim do jogo	

Criar classes e estado

A tabela anterior informa a história de como o jogo progride por diferentes fases. Ao se concentrar nas duas colunas `Behavior` e `Data`, você pode *realizar scaffold* de algum código inicial como base para criação do jogo.

1. Copie o código abaixo no terminal e pressione `Enter` para criar um arquivo `rock-paper-scissor.py` e abrir o editor:

   Bash

   ```
   touch rock-paper-scissor.py
   code .
   ```

   > **Dica**
   >
   > Use o menu contextual (clique com o botão direito do mouse) para colar o código no terminal.

2. Dê a ele o seguinte conteúdo e salve o arquivo (`CTRL + S` ou `Command + S` no macOS):

   Python

   ```
   class Participant:
   class GameRound:
   class Game:
   ```

   > **Dica**
   >
   > Use `Ctrl+V` para colar o código no editor.

Você tem as classes necessárias criadas para seu jogo. Em seguida, você precisa pensar em quais dados você tem e em qual classe colocá-los.

3. Continue trabalhando com o mesmo arquivo, expanda o código e salve o arquivo:

Python

```
class Participant:
    def __init__(self):
        self.points = 0
        self.choice = ""

class GameRound:

class Game:
    def __init__(self):
        self.endGame = False
        self.participant = Participant()
        self.secondParticipant = Participant()
```

Dica

- Você pode arrastar o separador entre o editor e o terminal para ajustar o espaço disponível.
- Você pode fechar o editor do menu `...`.
- Você pode digitar `code .` no terminal para reabrir o editor.

Demos à classe `Participant` os atributos `points` e `choice` conforme indicado pela primeira e terceira linhas da tabela.

Demos ao `Game` o campo `endGame` por causa da quarta linha. Além disso, a classe `Game` tem dois participantes, `participant` e `secondParticipant`. Havia duas funções que uma variável em um objeto poderia ter. As funções podem ser um estado, como o andar de um elevador, ou um atributo descritivo. Os atributos `points` e `choice` são variáveis de estado neste contexto. Os participantes na classe `Game` são atributos descritivos porque um jogo *tem* participantes.

Parabéns! Você adicionou classes ao jogo e criou dados – atributos que você atribuiu às classes criadas. Neste ponto, você tem um bom código inicial. Ele não faz muita coisa ainda porque precisa de comportamento. Você adicionará o comportamento na próxima unidade do exercício.

Unidade seguinte: Adicionar comportamento com métodos

Continuar >

Azure Cloud Shell