LABORATÓRIO DE DESENVOLVIMENTO DE SOFTWARE | 21179 - EXAME

Data e hora de realização

27 de junho de 2023, às 10h00 de Portugal Continental

Duração da prova

120 minutos + 60 minutos de tolerância

Resolução

1.ª Parte (4 Valores)

1.

a) Indique as linhas do código de referência onde há operações de *input*.

R: 21, 27

b) Indique as linhas do código de referência onde há operações de *output.*

R: 21, 22, 27, 28, 33

Caso tenha tido dúvidas de interpretação do código que lhe afetaram a resposta, exponha-as por escrito.

R: Como não conhecemos o funcionamento interno dos objetos da classe api.Chat, não podemos descartar a possibilidade de nalgum momento ser estabelecida comunicação com o sistema externo para efeitos de configuração ou outros. Por exemplo, no CreateConversation. Foi tomada a opção de apenas identificar como *input* e *output* os casos em que ocorrem explicitamente.

2. Para reescrever o código de referência segundo o estilo arquitetónico MVC, indique como distribuir pelos componentes as tarefas descritas nos comentários desse código. Por ex., se considerar que a tarefa "`// Acrescentar ao pedido as instruções de fundo`" (linha 12) corresponde ao Controller, marque a célula C12, se acha que corresponde ao Model, marque a célula M12, para indicar que corresponde à View, marque a célula V12.

a) Segundo a abordagem de Krasner & Pope (1988).

Linha → / ↓ Componente	1	5	12	18	20	22	24	26	28	30
M		X	X							
V						X			X	X
C	X			X	X		X	X		

b) Segundo a abordagem de Curry & Grace (2008).

Linha → / ↓ Componente	1	5	12	18	20	22	24	26	28	30
M		X	X							
V				X	X	X	X	X	X	X
C	X									

c) Explique as dúvidas ou dilemas com que se debateu para responder às alíneas a) e b) e justifique as principais opções que tomou ao dar as suas respostas.

R: Para a variante de Krasner & Pope, o comentário da linha 20 é problemático porque tem input e output em conjunto: poder-se-ia ter assinalado V e C. Mas optei por encará-lo como tarefa mais alinhada com o Controller, porque implica parar o fluxo de execução do programa enquanto se aguarda o input, responsabilidade última do Controller. Idem para as linhas 26. Por fim, os pedidos do utilizador no código de exemplo são do Model (internos), mas vendo só para os comentários 18 e 24 (para reescrever) o que devia ocorrer seria um input. A resposta C (K&P) e V (C&G) advém disso, mas pode surgir como dúvida.

3. Suponha que antes de enviar a linha 21, `chat.GetResponseFromChatbotAsync();` queria verificar se tinha saldo na conta com que paga o serviço.

a) Explique como devem os componentes MVC do seu código articular-se para consultar o saldo e decidir se é suficiente ou não, agindo depois em conformidade (enviar ou não o pedido para o ChatGPT ou informar o utilizador da falta de saldo).

R: Tanto no caso K&P como C&G, estará no Controller o poder de decidir se está na altura de avançar para a tarefa da linha 21 ou não. Assim, o Controller começa por perguntar ao Model se o saldo é suficiente para o pedido. Se o saldo for suficiente, o Controller procede como nas respostas à pergunta 2: passa a tarefa à View (C&G) ou executa-a (K&P). Se o saldo for insuficiente, o Controller pedirá à View para informar o utilizador da falta de saldo.

b) Discuta se esta situação deveria ser ou não implementada com recurso a exceções em caso de falta de saldo, fundamentando a sua resposta.

R: Não, porque é uma situação perfeitamente previsível em termos do fluxo normal da lógica de negócio do programa.

4. Neste código de exemplo, as mensagens do utilizador estão simuladas, escritas diretamente no código (linhas 19 e 25). Num sistema real seriam solicitadas ao utilizador. Mas as instruções prévias e exemplos também estão escritas diretamente no código (linhas 6 a 16). Como poderia modificar este aspeto do código para melhorar a qualidade de adaptabilidade, respeitando o estilo MVC?

R: Para que este código tivesse mais adaptabilidade a mudanças de requisitos ou outras alterações no seu contexto sociotécnico, poderíamos ter as instruções prévias e exemplos como dados editáveis fora do código. Por exemplo em ficheiros ou bases de dados de configuração, ou mesmo num serviço Web. Seria preciso que o acesso a esses ficheiros/bases de dados/Web services não ocorresse no Model, mas sim envolvendo o Controller, para processar esse *input*.

2.ª Parte (8 Valores)

5. Reescreva o código de referência segundo o modelo MVC, de acordo com as seguintes alíneas. Como comentário no início do código, indique se está a usar a sua resposta 2a ou 2b como base para esta parte.

a) Para não ter de usar a biblioteca OpenAI-API-dotnet, crie uma classe `Chat` com os métodos do exemplo, mas vazios de código… Estamos apenas a evitar ter de usar a biblioteca para podermos compilar. Pode usar na linha 3 simplesmente `new Chat();`

R:
```
class Chat
{
  public static Chat CreateConversation() {}
  public void AppendSystemMessage(string msg) {}
  public void AppendUserInput(string msg) {}
  public void AppendExampleChatbotOutput(string msg) {}
  //Também se consideraria certa a linha seguinte sem a
  //programação assíncrona, ou seja se fosse apenas:
  //public string GetResponseFromChatbotAsync()
  public async Task<string> GetResponseFromChatbotAsync(){}
  //Na linha a seguir, qualquer outra coleção iterável
  //também seria válida.
  public List<ChatMessage> Messages {get;}
}
```

b) Crie classes para os componentes Controller, View e Model e dentro delas métodos (código vazio, apenas indicação dos parâmetros de entrada e tipo de retorno) correspondentes à distribuição de responsabilidades que fez na sua resposta 2a/2b

(ou seja, um método para cada linha de comentários do exemplo).

R:

```
class Controller
{//assume que chat é um objeto interno
  void CriarConversa();
  string ObterInputDoUtilizador();
  //Segundo K&P. Segundo C&G este método seria da View.

  string EnviarPedidoParaChat();
  //Segundo K&P. Segundo C&G este método seria da View
  //e nesse caso com o pedido num parâmetro.
}

class View
{
  void WriteResposta(string resposta);
  void WriteHistorico(Chat chat);
  // Segundo K&P, segundo C&G juntam-se os acima refs.
}

class Model
{
  void AcrescentarInstruçoesDeFundo(Chat chat);
  void AcrescentarExemplos(Chat chat);
}
```

c) Preencha o código dos métodos anteriores para reproduzir o exemplo com as suas classes dos componentes MVC.

R:

```
//Controller
void CriarConversa(){
  chat = new Chat();
}

//Controller K&P ou View C&G
string ObterInputDoUtilizador(){
  return Console.ReadLine();
}

//Controller K&P ou View C&G
//Se na View, "chat" tem de ser dado como parâmetro
string EnviarPedidoParaChat(){
  return await chat.GetResponseFromChatbotAsync();
}

//View
void WriteResposta(string resposta){
```

```
  Console.WriteLine(resposta);
}
void WriteHistorico(Chat chat){
  foreach (ChatMessage msg in chat.Messages){
    Console.WriteLine($"{msg.Role}: {msg.Content}");
  }
}

//Model
void AcrescentarInstruçoesDeFundo(Chat chat){
  chat.AppendSystemMessage(instruções); //atributo
}

void AcrescentarExemplos(Chat chat){
  chat.AppendUserInput(exemplos[0][0]); //atributo
  chat.AppendExampleChatbotOutput(exemplos[0][1]);
  chat.AppendUserInput(exemplos[1][0]);
  chat.AppendExampleChatbotOutput(exemplos[1][1]);
}
```

3.ª Parte (8 Valores)

6. Com a sua resposta à pergunta 5, reescreva o código para obter independência de componentes e separação de interesses, sem ter de ser código funcional. Por ex., não é necessário que a classe `Chat` que criou na alínea contacte o ChatGPT, basta que o simule (por ex. escrevendo no ecrã "Mensagem enviada: …").

a) Defina (não é preciso implementar, apenas apresentar as linhas de definição) eventos e delegados que serão usados para reportar alterações entre componentes.

R:

```
//Havia várias alternativas válidas, apenas se exemplifica.
//Por exemplo, em vez de se chamar a View com um objeto
//Chat, a View pode pedir que lhe seja enviado, isto
//permite que mais tarde se possa delegar a tarefa de gerar
//o chat a outras partes do programa.
//Na classe View:
public delegate void ForneceChat(ref Chat chat);
public event ForneceChat PrecisoDeChat;
```

b) No componente Controller, ligue os eventos dos componentes aos delegados respetivos.

R:

```
//Novamente, depende das múltiplas soluções possíveis.
//Aqui considera-se o exemplo anterior para um caso da
//View. Daria, no Controller:

void GetChat(ref Chat objetochat)
{
  objetochat=chat.Clone(); //Sem clone aceita-se também.
}

View.ForneceChat=GetChat;
```

d) Faça um `try-catch` no Controller que possa apanhar uma exceção lançada caso a resposta do Chatbot demore muito tempo (é lançada dentro do `GetResponseFromChatbotAsync`). No `catch` quer-se viabilizar uma reação prestável ao utilizador, respeitando as responsabilidades de cada componente no MVC. Não tem de implementar essa reação: basta que indique em comentário, dentro do `catch`, o que faria para o concretizar.

R:

```
//Uma exceção:
class PedidoLentoExc:Exception(){}

//O try-catch seria em redor dos "EnviarPedidoParaChat":
//que é onde está o GetResponse…
try{
    string resposta= view.EnviarPedidoParaChat();
}
catch(PedidoLentoExc)
{
    //Iria dizer que o chat está a demorar muito tempo
    //e perguntar se aguardava ou se tentava novamente
    //usaria a View e o Controller em coordenação.
}
```