UNIVERSIDADE AbERTA www.uab.pt

Ministério da Ciência, Tecnologia e Ensino Superior

Resolução / Resumo de correção do exame

U.C. 21179

Laboratório de Desenvolvimento de Software

6 de julho de 2018

1.ª Parte (4 valores)

1.

a) Linhas do código de referência onde há operações de *input*: 13. (0,5 valores)

b) Linhas do código de referência onde há operações de *output*: 13. (0,5 valores)

Observações: a nível do *input*, admitia-se como corretas as respostas 8 e 29 (linhas chamadas em consequência de *input* noutra parte do programa); "Nenhuma" (pois embora ShowDialog provoque *input*, isso pode não ser percetível a quem não tenha experiência de programação nesta API); 24/25 (pois embora não haja *input* na 24, pode-se considerar que esta o origina nas linhas das reticências).
A nível do *output*, considera-se parcialmente certas as respostas (além da linha 13) 24/25 e 31, por motivos idênticos aos relatados para o *input*.

2. a) Segundo a abordagem de Krasner & Pope (1988): M:03, C:07, C:10, C:15, V: 18. V:28 (0,3 valores)

b) Segundo a abordagem de Curry & Grace (2008): M:03, C:07, V:10, C:15, V: 18. V:28 (0,3 valores)

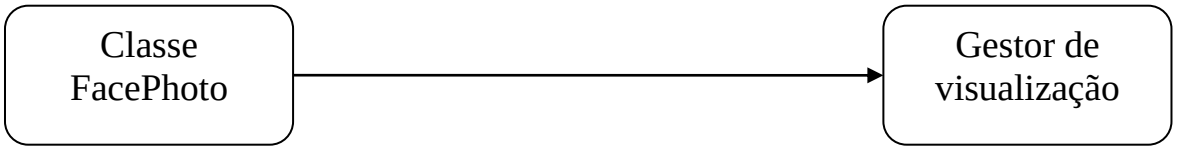
c) Dúvidas ou dilemas com que se debateu para responder às alíneas a) e b) e justificações (0,4 valores): Espera-se dos alunos a identificação de aspetos do código cuja classificação como M, V ou C possa ser alvo de argumentação lógica a favor de mais do que uma destas componentes, apresentando esses dilemas e fundamentando as opções que tomaram. Apresentam-se aqui alguns exemplos de aspetos que poderiam mencionar como oferecendo dúvidas:

- na abordagem de Krasner & Pope, embora o input do bloco "Obter do utilizador…" corresponda ao Controller, sendo admissível nesse processo pequenas ações de output exclusivamente para obter esse input, tal poderia ser um dilema: apresentar uma caixa de diálogo inteira não deveria ser responsabilidade da View? Expor este dilema leva a considerar a opção V:10 como correta na alínea 2a;
- na mesma abordagem, a tarefa "Mostrar a imagem" corresponde à View. Contudo, como é necessário ler a imagem de disco para a poder mostrar, isso pode originar o dilema de considerar-se que este bloco de código tem partes da View e do Controller. Se esse dilema for exposto, considera-se a opção C:18 correta na alínea 2a;
- na abordagem de Curry & Grace, o input é processado na View, mas a reação a ele, originando nova View, ocorre no Controller, pelo que a secção "Reage ao clique" pertence ao Controller. Contudo, podia surgir o dilema de considerar que a reação imediata seja na View e só depois transite para o Controller, optando o aluno por indicar V:08 e explicar este dilema, considerando-se neste caso essa opção como correta na 2b.

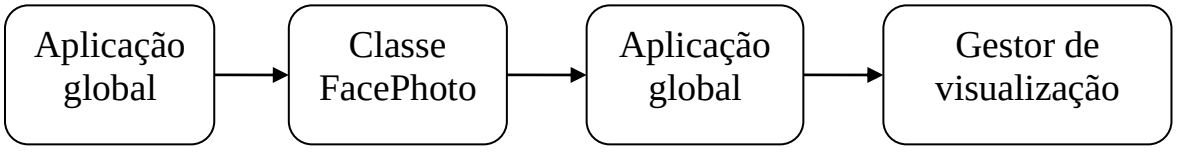
3. (1 valor, equitativamente distribuído pelos três casos)

i:

ii:



iii:



4. a) (0,4 valores) Não existe qualquer falha, erro ou falta, pois trata-se apenas de um comportamento e estado previsto, parte do fluxo normal de execução do programa.

b) (0,3 valores)O aspeto essencial da resposta é dar um exemplo que demonstre a compreensão do significado do conceito de falha. São aceitáveis exemplos que sejam baseados em conceitos gerais de

programação, ainda que o comportamento exato na plataforma .Net possa ser diferente. Por exemplo, pode não haver memória para criar o objeto referenciado por openDlg na linha 11, o que na plataforma .Net origina nessa linha uma falha, mas poderia considerar-se, à imagem de outros ambientes, que a falha só ocorresse na linha 12, quando se tentasse aceder a uma propriedade ou atributo do objeto. O relevante é deixar claro que a falha é o acontecimento instantâneo (programa termina abruptamente, "crasha", etc.) e não o estado de erro (estado inconsistente do computador) nem a ausência de código (falta) para lidar com ele. Outros exemplos poderiam ser usados.

c) (0,3 valores) O aspeto essencial da resposta é dar um exemplo que demonstre a compreensão do significado do conceito de falta. Por exemplo, recorrendo ao mesmo exemplo da alínea b), considera-se correto indicar como "falta" tanto a ausência de código para verificação do resultado do new como a não inclusão do new dentro de um bloco try.

2.ª Parte (8 Valores)

5. Tomando como base a sua resposta à pergunta 2, opte por uma das variantes do estilo arquitetónico MVC. Reescreva o código no estilo arquitetónico MVC, considerando o seguinte:

   a) Indique a variante escolhida por si para responder a esta pergunta e desenhe o diagrama MVC correspondente, **substituindo os rótulos genéricos que encontra nos materiais de referência com os aspetos específicos deste caso**.

      Usar os diagramas de referência da última página do enunciado e alterar o texto. Por exemplo, se a opção for pelo estilo de Krasner & Pope, solucionado em 2a) como: "M:03, C:07, C:10, C:15, V: 18. V:28" então a bola "M" deveria ter o número 03,

a bola C os 07, 10 e 15 e a V os restantes. Os rótulos genéricos da comunicação e controlo também deviam ser substituídos. Há várias soluções admissíveis, sendo o essencial que se respeite a ligação entre módulos. Por exemplo, para "Mensagens de visualização" (ligação Controller-View) admitir-se-ia "Mostrar a imagem" (pois há uma transição controller->view no código). Para a ligação View->Model, podia-se indicar "parte do FacePhoto_MouseMove" ou "consulta de faceDescriptions", porque para mostrar a descrição de uma face na View é preciso consultá-la no Model. E assim sucessivamente.

b) Defina as classes que pertencem aos componentes Model, View e Controller. Não utilize eventos, delegados, interfaces, nem exceções, apenas os atributos e métodos habituais.

Deve-se usar a divisão da resposta 2a) ou 2b). Há várias soluções válidas. Por exemplo, na classe Model poder-se-ia ter os atributos faces e faceDescriptions, bem como os métodos para o Controller solicitar a sua alteração e para a View consultar o valor atual. O Controller, no caso 2a), teria os atributos ligados ao input, como openDlg e o método de obter do utilizador o ficheiro a analisar. Já no caso 2b) estes elementos estariam na View. Também na View estariam os métodos de mostrar a imagem e a descrição. O Controller pode ter um método "Lançar a View" e outro "Iniciar o Model", entre outras soluções.

c) Escreva o código de um método `Controller.Main();` que controle o funcionamento global da aplicação.

O método Controller.Main depende da resposta à alínea anterior, mas o essencial é que demonstre compreensão do funcionamento do estilo MVC escolhido, não da arquitetura .Net. Por exemplo, para o estilo de Krasner & Pope:

```
Controller.Main (){
  model = new Model();
  view = new View(model);
  Uri ficheiro;
  int opcao = -1;
  int ImagemEscolhida = -1;
  while (opcao!=0){
      bool opcao = MostrarOpcoesEMostrarRato();
      if(opcao==1){ //Escolher ficheiro de imagem
        ficheiro=ObterFicheiroAnalisar();
        model.GuardarFacesEDescricoes(Processar(ficheiro));
        view.MostrarImagem(ficheiro);
      }
      if(opcao==2) //Mostrar descrição
        view.MostrarDescricao(ImagemEscolhida);
  }
  exit(); /*Para ser correto em .Net deveria (e poderia) ser
uma das soluções adequadas do .Net, como Application.Exit ou
Environment.Exit, mas não é a plataforma .Net que está em
avaliação, nem estas soluções faziam parte do conteúdo da
unidade curricular, pelo que qualquer solução que explicite a
intenção do programador seria considerada correta.*/
}
```

Para o estilo de Curry&Grace bastaria que as tarefas de input (MostrarOpcoesEMostrarRato, ObterFicheiroAnalisar) fossem feitas no componente View. Isto poderia ser como métodos da classe única View ou como classes ViewCentral, ViewEscolha, ViewLeituraFicheiros. A abordagem de resolução é livre.

3.ª Parte (8 Valores)

6. Tomando como base a sua resposta à pergunta 5, reescreva o código para obter independência de componentes e separação de interesses, da seguinte forma:

a) Defina e utilize eventos e delegados para reportar alterações nos módulos.

b) Defina e utilize interfaces para que a comunicação de dados ocorra com menos dependências entre módulos.

c) Defina exceções e escreva o código que as lança em caso de erro. Escreva igualmente o código que as apanha, respeitando as responsabilidades de cada módulo no MVC.

A resposta depende da pergunta anterior e tem muitas soluções possíveis. O essencial é demonstrar domínio da aplicação das três técnicas.

a) Por exemplo, em vez do Controller chamar model.GuardarFacesEDescricoes(Processar(ficheiro)); e depois view.MostrarImagem(ficheiro), pode definir um evento "FicheiroProcessado", que será lançado quando terminar o método "ProcessarFicheiro" e associar a esse evento, como delegados, os dois métodos: GuardarFacesEDescricoes, do Model; e MostrarImagem, da View.

b) Em vez de passar um endereço de ficheiro à View, que apenas tem de mostrar o conteúdo mas implica dar à View responsabilidades de o abrir e de o fechar, poder-se-ia definir uma interface iImagem que permite apenas obter os dados a mostrar (por exemplo, com um único método DadosDaImagem). O controller ficaria com a responsabilidade de fazer a leitura do ficheiro no método Processar (não era necessário implementá-lo) e passaria à View, como resultado do Processar um objeto ImagemLida que implementava iImagem.

c) Qualquer aspeto que possa eventualmente gerar um erro pode ser escolhido, mas de preferência dentro da View ou do Model, para que a decisão de como reagir ao erro tenha de ocorrer no Controller. Por exemplo, pode-se assumir que o "MostrarDescricao" da View pode gerar um erro (por exemplo, não conseguir criar o tipo de letra ou outra situação do género), fazer-se um try à volta dele no Controller e no catch da exceção estar a apanhar uma exceção personalizada (por ex. ExcecaoVistaDesconfigurada). Dentro desse catch pode-se depois fazer qualquer tipo de reação, por exemplo, chamar um método da View para simplificar o ecrã (view.DefineEcraSimples) ou para o ler em voz alta (view.DefineDescricaoAudio) ou outra solução e lançar novamente o MostrarDescricao. Ou mesmo mudando o funcionamento para simplesmente mandar a view alertar o utilizador para o problema: view.IndicarErro("Não foi possível mostrar a descrição, reinicie o programa.").