1 - Inputs e Outputs em código

Classifique cada linha como sendo de *input*, *output*, ambas as coisas ou nenhuma das coisas.

```
1. // Monitor térmico de depósito de água quente
2. using IoT; // 'Internet of Things' com várias coisas
3. using Time;
4. // Configuração inicial
5. var sensor = IoT.ObterSensor();
6. var logger = new Logger();
7. double leitura = 0;
8. // Comunicar o início da execução
9. Console.WriteLine("A correr...");
10. // Monitorizar a temperatura do depósito
11. while(true){
12.  leitura = sensor.Ler();
13.  if (leitura > 70) throw new ExcecaoSobreaquecimento(); // Detetar perigo
14.  Console.WriteLine($"{leitura} °C");
15.  logger.Registar(leitura);
16.  Thread.Sleep(10000);
17. }
```

		Input	*Output*	*Input* e *Output*	Nem *Input*, nem *Output*
A	`// Monitor térmico de depósito de água quente`	○	○	○	◉ ✔
B	`using IoT; // 'Internet of Things' com várias coisas`	○	○	○	◉ ✔
C	`using Time;`	○	○	○	◉ ✔
D	`var sensor = IoT.ObterSensor();`	○	○	○	◉ ✔
E	`var logger = new Logger();`	○	○	○	◉ ✔
F	`double leitura = 0;`	○	○	○	◉ ✔
G	`Console.WriteLine("A correr...");`	○	◉ ✔	○	○
H	`while(true){`	○	○	○	◉ ✔
I	`leitura = sensor.Ler();`	◉ ✔	○	○	○
J	`if (leitura > 70) throw new ExcecaoSobreaquecimento();`	○	○	○	◉ ✔
K	`Console.WriteLine($"{leitura} °C");`	○	◉ ✔	○	○
L	`logger.Registar(leitura);`	○	◉ ✔	○	○
M	`Thread.Sleep(10000);`	○	○	○	◉ ✔
N	`}`	○	○	○	◉ ✔

Que aspetos de interpretação do código lhe afetaram a resposta?

A mim fiquei na duvida se as Operacoes de I/O diretas e de transferencia de dados, ou seja, apenas as linhas que explicitamente leem os dados para o programa (sensor.Ler()) ou escrevem dados para fora do programa (Console.WriteLine, logger:Registrar()) serem classificadas como input ou output.

Em suma depois de bastante refletir, a principal opcao que tomei na minha interpretacao foi de considerar input e output como a transferencia direta de dados para dentro ou para fora dos limites da aplicacao (seja via consola, ou via metodos que leem / escrevem de/para sistemas externos), distiguindo desta forma de operacoes de configuracao ou de logica interna de programa.

105 / 200 Word Limit

2 - Reescrever em MVC

Para reescrever o código de referência segundo o estilo arquitetónico MVC, indique como distribuir pelos componentes as tarefas descritas nos comentários do código.

Reescreva segundo a abordagem de Krasner & Pope (1988)

```
1. // Monitor térmico de depósito de água quente
2. using IoT; // 'Internet of Things' com várias coisas
3. using Time;
4. // Configuração inicial
5. var sensor = IoT.ObterSensor();
6. var logger = new Logger();
7. double leitura = 0;
8. // Comunicar o início da execução
9. Console.WriteLine("A correr...");
10. // Monitorizar a temperatura do depósito
11. while(true){
12.   leitura = sensor.Ler();
13.   if (leitura > 70) throw new ExcecaoSobreaquecimento(); // Detetar perigo
14.   Console.WriteLine($"{leitura} ºC");
15.   logger.Registar(leitura);
16.   Thread.Sleep(10000);
17. }
```

0,4 / 0,6

	4	8	10	13
Model	☑ ✗	☐	☐	☑ ✓
View	☐	☑ ✓	☐ ✓	☐
Controller	☐ ✓	☐	☑ ✓	☐

Reescreva segundo a abordagem de Curry & Grace (2008)

```
1. // Monitor térmico de depósito de água quente
2. using IoT; // 'Internet of Things' com várias coisas
3. using Time;
4. // Configuração inicial
5. var sensor = IoT.ObterSensor();
6. var logger = new Logger();
7. double leitura = 0;
8. // Comunicar o início da execução
9. Console.WriteLine("A correr...");
10. // Monitorizar a temperatura do depósito
11. while(true){
12.   leitura = sensor.Ler();
13.   if (leitura > 70) throw new ExcecaoSobreaquecimento(); // Detetar perigo
14.   Console.WriteLine($"{leitura} ºC");
15.   logger.Registar(leitura);
16.   Thread.Sleep(10000);
17. }
```

0,5/0,6

	4	8	10	13
Model	☑ ✗	☐	☐	☑ ✓
View	☐	☑ ✓	☑ ✓	☐
Controller	☐ ✓	☐	☐ ✓	☐

Explique as dúvidas ou dilemas com que se debateu para responder a esta secção e justifique as principais opções que tomou ao dar as suas respostas.

Para comecar, os meus dilemas centram-se em como as responsabilidades I/, controlo de fluxo e logica de negocio sao distribuidas, como a principal dinstincao entre as duas abordagens a surgir na autonomia da View para gerir as suas proprias interacoes e ciclos I/O.

O meu maior dilema foi na linha 10 (Monitorizar): Parece-me um bloco misto d I/O e logica.

- Na versao K&P optei por Controller, por ser a orquestracao central que coordena input, logica e ouput;
- Na versao C&G optei por View dada a sua maior atividade na gestao direta de interfaces (sensor como input, consola como output)

Tomar decisoes nao sao sempre faceis dai os dilemas que apresento. Tentei explicar o porque dos dilemas e porque é que tomei esta ou aquela decisao.

125 / 200 Word Limit

0,2/0,3

3 - Responsabilidades em MVC

0,4/0,5

Suponha que pretende adaptar o código do monitor térmico, para que antes de apresentar no ecrã a nova leitura e a registar, verifique se a temperatura lida é igual à anterior.
Se a nova temperatura for igual à anterior, a leitura não deve ser apresentada ao utilizador, nem registada novamente.

Com base nos princípios de **separação de responsabilidades**, **coesão forte** e tendo em conta a **manutenção futura do código**, selecione todas as propostas de distribuição de responsabilidades entre componentes MVC que são coerentes com esses princípios.

Tenha em consideração quer a variante de Krasner & Pope, quer a de Curry & Grace: se uma resposta for válida para qualquer uma delas, deve ser considerada correta.

- [ ] O **Controller** obtém a leitura atual do sensor, pede ao **Model** a leitura anterior chamando um método `UltimoRegisto()` , compara as duas e decide se pede ou não ao **Model** para a registar e à **View** para a apresentar.
- [x] O **Model**, que tem acesso interno a todas as leituras registadas, disponibiliza um método `RegistarSeNova(leitura)` , que apenas regista uma leitura se for diferente da anterior e, caso o seja, lança um evento para notificar que tal ocorreu. O **Controller**, anteriormente, associou esse evento a um delegado da **View**. Então o **Controller** chama `RegistarSeNova` com a leitura nova.
- [ ] O **Controller** guarda internamente o valor da última leitura, depois de a ter enviado ao **Model** para registo. Quando faz uma nova leitura, o **Controller** compara-a primeiro com esse valor que tem guardado, para decidir se pede ao **Model** para a registar e à **View** para a apresentar, ou se ignora a leitura.
- [ ] A **View** obtém todas as leituras e decide se as apresenta ou não, e se as comunica ao **Controller** ou não, com base numa variável local da última leitura mostrada.
- [ ] A **View** obtém todas as leituras e comunica-as ao **Controller**, tenham elas os valores que tiverem. O **Controller** chama o método `RegistarSeNova(leitura)` do Model.
- [ ] O **Model** e a **View** recebem todas as leituras do **Controller** e decidem internamente se as registam (Model) ou apresentam (View) ou não.

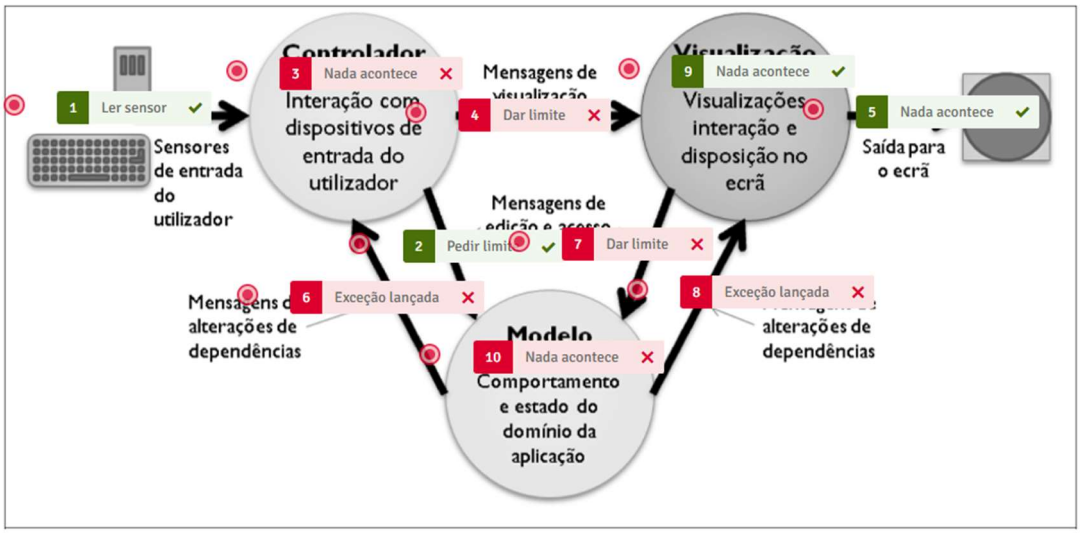
4 - MVC com exceções

0,2/0,5

Considere que o código do monitor térmico foi reorganizado segundo a variante do estilo MVC representado na imagem.
Agora, a temperatura de sobreaquecimento não está fixa em 70 °C: é uma configuração do utilizador, guardada no programa.

Imagine que ocorreu a seguinte situação: foi detetada uma temperatura superior ao limite configurado, levando ao lançamento da exceção.

Com base nesta variante do MVC, selecione nas listas pendentes junto a cada seta da imagem o tipo de ação que ela representa nesta situação.

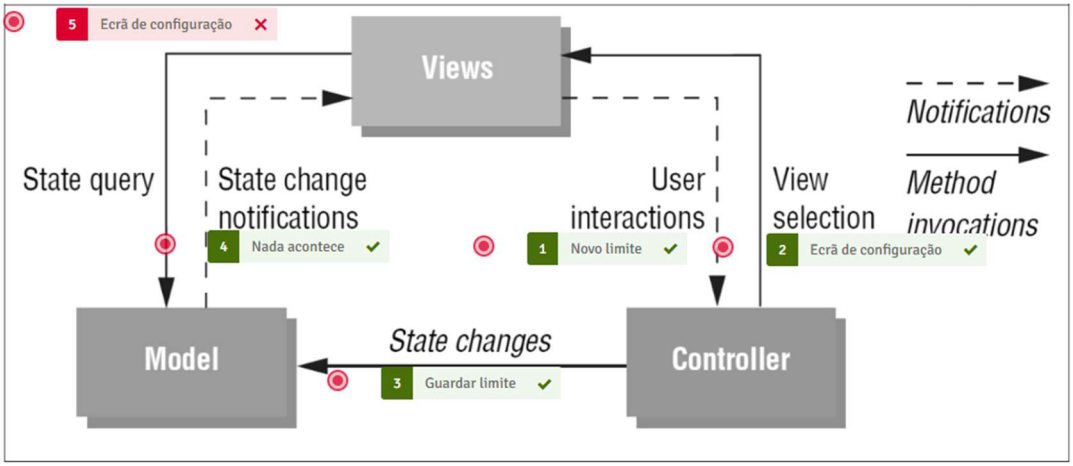


Correct answers:

3 Exceção lançada | 4 Nada acontece | 6 Dar limite | 7 Nada acontece | 8 Nada acontece | 10 Ler limite

Suponha que o utilizador pretende alterar o limite da temperatura de sobreaquecimento para um novo valor.
Com base na variante de MVC representada na imagem, selecione, nas listas pendentes, o tipo de ação que ela representa nesta situação.

0,5/0,6



Correct answers:

5 Nada acontece

5 - Equipa e projeto

Indique uma decisão concreta, não trivial, tomada pela sua equipa ao longo do projeto, que tenha tido impacto na separação de responsabilidades entre Model, View e Controller, **sem ser a escolha entre uma das duas variantes de MVC.**
Explique por que motivo essa decisão não foi trivial, ou seja, porque é que foi um dilema entre alternativas (e quais eram as alternativas).
Por fim, explique se acha, em retrospetiva, que foi a melhor decisão ou o que teria mudado.

3/3

Penso que houveram decisoes que tivemos de ter atencao redobrada quando as escolhemos em grupo. Algumas das vezes depois de tomadas, tivemos de voltar atras e rediscutir e mudar de opiniao.

Gostaria de partilhar convosco a decisao chave de - Comunicacao do Controller-View Padrao Observer (Eventos).

A decisao que tomamos foi de utilizar os eventos para o BotController notificar o DiscordView (ex. apos gerar relatorios) passando o caminho do ficheiro.

E entao porquê do dilema nesta parte - tinhamos dois caminhos possiveis:

1. Controller chamar directamente metodos da View - view.enviar_relatorio() --> criaria a meu ver um acoplamento forte (violando assim a separacao MVC);
2. A View ficar a "perguntar" ao Controller - Polling - complexidade e ineficiencia (pelo menos era o que achava);

Olhando para tras, acrededitо que usar o padrao Observe (Eventos) foi a melhor decisao pois garantiu baixo acoplamento, mantendo a separacao de responsabilidades (Controller gera, View apresenta) e assim facilitou a extensibilidade (outros componentes podem subscrever os mesmo eventos).

Em suma, talvez se tivessemos padronizado ainda mais a forma como os eventos sao definidos e registrados (usando envent bus por ex) ficaria ainda melhor o nosso projeto neste aspeto, mas a essencia do padrao Observer, quanto a mim foi fundamental e funcionou.

199 / 200 Word Limit

Durante o projeto, cada equipa contou com membros com diferentes responsabilidades.

Descreva uma situação concreta em que tenham tido de discutir e tomar decisões ativamente, **com a sua participação ativa**, sobre o funcionamento da colaboração da equipa, já depois de terem dividido as responsabilidades entre vós. (Ou seja, não na fase inicial de formação da equipa, mas já depois com o projeto em andamento.)

Explique:

- qual foi a situação ou problema enfrentado;
- qual foi a sua intervenção específica nessa situação;
- como é que essa discussão evoluiu e contribuiu para avançarem (ou resolverem o problema).

2/2

Durante este projeto desfiador, foram imensas as discussoes que tivemos e muitas dores de cabeca para decidir o que era melhor para o projeto.

Uma situacao que me lembro aconteceu ja depois da divisao de responsabilidades, surgiu a questao de como o BotController deveria entregar os relatorios/graficos gerados a DiscordView para serem enviados ao Discord.

Eu apesar de nao ter percebido ao inicio o problema e depois de pesquisar noutro trabalho que tinha participado, procurei e propus a utilizacao do padrao Observer(eventos). O BotController deveria "disparar" eventos (relatorio_gerado, grafico_gerado) como o caminho do ficheiro e a DiscordView deveria subscrever-se a estes eventos.

Depois de muito termos discutido em equipa, esta concordou que esta abordagem promovia um certo baixo acoplamento e separacao de responsabilidades. Com esta tomada de decisao, o problema da comunicacao mantento a manutencao e extensibilidade do codigo foram alinhados com os principios de qualidade. Ou seja, esta decisao permitiu que a logica da geracao de ficheiros permanecesse no Controller e a logica de interacao com o Discord permanecesse na View, sem que uma camada dependesse da implementacao da outra.

181 / 250 Word Limit

Indique uma parte do código da aplicação desenvolvida pela sua equipa com a qual esteja especialmente familiarizado/a. Algo não trivial, como um método ou algoritmo complexo, uma classe com subtilezas, um processo de comunicação entre componentes de que se orgulhe etc.

Descreva:

- qual o seu papel nessa parte do código (especificação/acompanhamento, desenvolvimento, verificação);
- como está implementada (quais as principais decisões tomadas na sua concretização);
- que dificuldades houve para a concretizar (problemas, hesitações, erros etc.).

3/3

Uma parte do codigo que me lembro assim mais foi o subsitema de geracao de relatorios e graficos -- !relatorio, !gerar_comandos. Eu tinha de analisar como tester se tudo estes relatorios ou graficos estavam a sair correctamente quando pedidos.

Ele ficou assim implementado:

- Dados: Botcrotller obtem os dados brutos do Consultmodel e processa-os com pandas
- Comunicacao: Botcontroller notifica a Dicrodview via eventos, passando os caminhos dos ficheiros para serem enviados para o Discord.

As minhas dificulades (do grupo tambe):

- Escolha de biblioteca PDF: hesitamos e tivemos o desafio em unir o uso de weasyprint e reportlab;
- Robustez dos dados aonde tentamos assegurar que os dados fossem o mais correcto possiveis com pandas;
- Tratamento de erros: para este tipo de funcionalidades o tratamento generico de Exception como o comando_seguro pode nao ter sido muito benefico, poderiamos ter utilizados execpcoes personalisafas como ReportGenerationError - para fornecer feedback mais precisos para os utilizador em casa de falha.

Foi um super desafio trabalhar neste projeto.

159 / 200 Word Limit