Instituto Superior de Engenharia de Coimbra

Programação Avançada – 2018/19

Exame da época normal 11-06-2019

Nome: _____

Número de Aluno: _____

Duração: 2h30 *Sem consulta*

IMPORTANTE: Deve responder às perguntas 1, 2, 5 e 6 nesta folha. A pergunta 3 deve ser respondida na folha de prova principal e a pergunta 4 numa de continuação. Pode rasurar a resposta desde que esta seja perceptível.

1 - [4%] Considere a **Listagem A**. Indique com um X qual é a sequência de mensagens que este programa imprime. Escolha apenas uma.

____ quantos=0 quantos=2; a[1]=(10,3) ; a[0]=(10,3)

____ quantos=0 quantos=1; a[1]=(10,2) ; a[0]=(10,3)

____ quantos=0 quantos=0; a[1]=(10,1) ; a[0]=(10,2)

____ quantos=0 quantos=0; a[1]=(10,1) ; a[0]=(10,1)

____ quantos=0 quantos=2; a[1]=(10,3) ; a[0]=(10,4)

2 - [8%] Considere o código da **Listagem B**. O programa imprime 6 mensagens.

a) **[4%]** Indique com um X quais as 3 primeiras mensagens. Escolha apenas uma opção.

____ false true true

____ false false true

____ true true true

____ false false false

____ false true false

b) **[4%]** Indique com um X quais as 3 últimas mensagens. Escolha apenas uma opção.

____ false true true

____ true false true

____ false true false

____ false false false

____ true true false

3 - [4%] Considere a **Listagem C**. Indique com um X qual é a sequência de mensagens que este programa imprime. Escolha *apenas uma*

____ m1 f1 f3 f4 f5 m4 m5

____ m1 f1 f2 f3 f4 f5 m2 m4 m5

____ m1 f1 f3 f5 m3 m5

____ m1 f1 f3 f5 m4 m5

____ m1 f1 f3 m4 m5

____ m1 f1 f3 f5 f6 m4 m5

4 - [4%] Considere o código da **Listagem D**, onde existem 9 anomalias que impedem a sua compilação com sucesso. Identifique as linhas onde estas ocorrem.

____ ____ ____ ____ ____ ____ ____ ____ ____

5 - [25%] [ESTA PERGUNTA DEVE SER RESPONDIDA NA **FOLHA DE PROVA PRINCIPAL]** Considere o código da **Listagem E**, onde se encontra definida a classe ***SelfCheckoutCashdesk*** destinada a controlar o estado de caixas de supermercado self-service. Uma caixa self-service caracteriza-se por: (1) permitir o registo de itens através da leitura dos respectivos códigos de barra; (2) verificar, através do incremento de peso observado, se os itens vão sendo colocados no saco de compras depois de registados; (3) efectuar o pagamento das compras; e (4) aguardar pela intervenção de um funcionário sempre que surge um determinado problema. O método ***processEvent(Enums event, Info info)*** da classe ***SelfCheckoutCashdesk*** é invocado depois de ocorrer um determinado evento e a classe ***Info*** possui, entre outros, os seguintes métodos públicos: ***double getBagWeight()***, ***double getScannedItemsWeight()*** e ***boolean paymentSuceeded()***.

Pretende-se uma nova versão da classe ***SelfCheckoutCashdesk*** que seja realizada sob a forma de uma máquina de estados orientada a objectos. Considere, para o efeito, o padrão estudado nas aulas laboratoriais em que uma máquina de estados é uma instância de uma classe que, entre outros possíveis atributos, inclui o seu **estado** actual.

a) **[12,5%]** Represente o **diagrama** da máquina de estados ***SelfCheckoutCashdesk*** que idealizou, com atribuição de nomes aos estados e às transições/métodos que sejam concordantes com o código da **Listagem C**. Deve representar todas as transições que podem ocorrer entre os estados. Cada transição, incondicional ou condicional, deve ser representada por uma seta graficamente distinta.

b) **[12.5%]** Relativamente à nova versão da máquina de estados ***SelfCheckoutCashdesk***, implemente, de forma adequada, uma interface ***IStates*** e uma classe ***SateAdapter*** que permita evitar definir todos os métodos em todos os estados concretos. A interface ***IStates*** não deve possuir qualquer método designado ***processEvent***.

Programação Avançada - Exame da época normal - 11-06-2019

Nome: _____

Número de Aluno: _____

6 - [15%] Considere que pretende-se desenvolver interfaces de utilizador que permitam representar, de forma actualizada, o estado de caixas self-service encapsuladas pela classe ***SelfCheckoutCashdesk*** da **Listagem E**. Considere, igualmente, que estão disponíveis a interface e a classe definidas na **Listagem F**.

a) **[7.5%]** Indique que alterações (código e local) são necessárias na classe ***SelfCheckoutCashdesk*** ou de que maneira deve ser utilizada (criando outra classe que a refere) para que seja possível registar e actualizar automaticamente (i.e., de forma assíncrona) interfaces de utilizador (vistas) sempre que o método ***processEvent*** é invocado. Devem obrigatoriamente recorrer aos elementos disponibilizados na **Listagem F**.

_____

_____

_____

_____

_____

_____

b) **[7.5%]** A classe ***CashDeskBasicView*** (**Listagem G**) destina-se a apresentar na consola (*System.out*) uma mensagem com a situação actual da caixa self-service observada sempre que esta sofre uma mudança (e.g., "Situation changed: IDLE"). Complete o código desta classe, apresentado na listagem G, ou apresente uma solução funcionalmente equivalente.

A _____

B _____

C _____

D _____

_____

_____

7 - [15%] Pretende-se definir, considerando o padrão *Command* estudado nas aulas, a classe ***ProcessEventCommand*** (**Listagem H**) destinada a encapsular invocações ao método ***processEvent(Enums event , Info info)*** da classe ***SelfCheckoutCashdesk*** (**Listagem E**).

a) **[7.5%]** Complete o código da classe ***ProcessEventCommand***.

A _____

B _____

C _____

D _____

E _____

b) **[7.5%]** Reescreva a expressão ***cashDesk.ProcessEvent(event, info)***, sendo ***cashDesk*** uma instância da classe ***SelfCheckoutCashdesk*** (**Listagem E**), recorrendo ao comando/classe ***ProcessEventCommand*** da **Listagem H**.

8 - [25%] **[**ESTA PERGUNTA DEVE SER RESPONDIDA NUMA **FOLHA DE PROVA DE CONTINUAÇÃO]** Complete a **Listagem I**, relativa a um hipotético sistema de gestão de propostas de projectos (*projects*) e estágios (*internships*) curriculares, atendendo ao resultado esperado da execução do método ***main*** da classe ***UseFactory*** e aos seguintes pressupostos:

- Duas propostas são consideradas iguais quando possuem o mesmo identificador (atributo *id*) e pertencem ao mesmo período (ano lectivo e semestre);
- A ordenação natural de uma lista de propostas é definida pelo valor retornado pelo método ***compareTo(Proposal o)*** da interface ***Comparable<Proposal>*** (zero, negativo ou positivo caso o objecto *this* seja, respectivamente, igual, inferior ou superior ao objecto *o* em termos de ordenação). Para o efeito, deve começar por ser usado o ano lectivo em ordem crescente. Em caso de igualdade, passa a ser usado o semestre em ordem crescente. Finalmente, se a situação de igualdade se mantiver, é usado o identificador em ordem crescente;
- Os métodos ***toString*** das classes ***Project*** e ***Internship*** devem recorrer ao valor devolvido pelo método homólogo da classe de base ***Proposal***.

Escreva na folha de prova apenas o código em falta identificando a respectiva alínea, entre A e J.