

## Pergunta 1

0,25 em 0,25 pontos



Para que se consiga gerar um painel com 20 partes de tamanhos iguais, de forma que cada parte receba, apenas, um componente, deve-se utilizar o seguinte comando:

Resposta Selecionada: ☒ b. `painel.setLayout(new GridLayout(4, 5)).`

- Respostas:
- ☐ a. `painel.setLayout(BorderLayout(20)).`
  - ☒ b. `painel.setLayout(new GridLayout(4, 5)).`
  - ☐ c. `painel.setLayout(20).`
  - ☐ d. `painel.setLayout(new FlowLayout(5, 4)).`
  - ☐ e. `painel.setLayout(new CardLayout(20)).`

Comentário da resposta: Resposta: B

Comentário: um painel é um objeto do tipo "*Panel*", que é um *container*, ou seja, um objeto que pode receber outros componentes. O painel possui um *layout* padrão (que é o *FlowLayout*), que não divide o painel em partes. Assim, o tipo do *layout* deste painel pode ser redefinido, com o método "`setLayout(new TipoLayout())`". O *GridLayout* é um *layout* que divide o painel em partes iguais, dividindo-o em linhas e colunas definidos nos parâmetros da sua instância (números inteiros), de forma que o primeiro valor do parâmetro define a quantidade de linhas e o segundo valor define a quantidade de colunas. No caso do item cuja opção é o "*GridLayout*", o painel está sendo dividido em 4 linhas e 5 colunas, formando, ao todo, 20 partes de tamanhos, exatamente, iguais. Com os outros *layouts* não é possível dividir o painel em 20 partes. O *BorderLayout* divide em 5 partes (norte, sul, leste, oeste e central). Já o *FlowLayout* e o *CardLayout* não dividem o painel em partes distintas.

## Pergunta 2

0,25 em 0,25 pontos



Em linguagem Java, ao trabalharmos com *menus*, podemos afirmar que, se trabalharmos com a biblioteca do Swing:

- I. Um *JMenu* é um componente que permite gerar novas opções de *menus*, e que pode ser adicionado a um *JMenuBar* ou, também, a um outro *JMenu*;
- II. Um *JMenuItem* é um componente que, ao ser selecionado, realiza uma ação no sistema;
- III. A Barra de *Menu* (*JMenuBar*) horizontal, que fica na parte superior da janela, é adicionada a ela através do método "`add`" da própria janela: `this.add(...)` ...
- IV. Tanto o *JMenuBar* quanto o *JMenu* podem receber os componentes de *menus* através do método "`add`".

Assinale a alternativa correta:

Resposta Selecionada: ☒ d. Apenas as alternativas I e II estão corretas.

- Respostas:
- ☐ a. Apenas as alternativas II, III e IV estão corretas.
  - ☐ b. Apenas a alternativa III está correta.
  - ☐ c. Apenas as alternativas I, II e IV estão corretas.
  - ☒ d. Apenas as alternativas I e II estão corretas.
  - ☐ e. Todas as alternativas estão corretas.

Comentário da resposta: Resposta: D

Comentário: existem vários "componentes de *menu*", disponíveis para se trabalhar na linguagem Java, utilizando-se da biblioteca do Swing, como, por exemplo, o *JMenuBar*, o *JMenu*, o *JMenuItem*, entre outros que possuem utilidades específicas, quando estamos tratando de elementos de *menus*. A afirmação I explica, corretamente, a utilidade do componente *JMenu* que, em uma janela, pode ser acessado a partir da barra de *menus*, ou a partir de outro menu, dependendo de onde foi localizado no programa. A afirmação II explica, corretamente, a utilidade do componente *JMenuItem* que, em uma janela, é acionado a partir de um *menu*, e que não mostra novas opções de *menu*, mas sim, realiza alguma ação no programa (como, por exemplo, o item de *menu* "*Sair*" que, normalmente, existe para fechar e finalizar o programa).

## Pergunta 3

0,25 em 0,25 pontos



O que é correto afirmar sobre os gerenciadores de *layout* da biblioteca java.awt?

Resposta  
Selecionada:

☒ e. São classes que tem por objetivo gerenciar a disposição e o arranjo dos componentes em uma interface gráfica.

Respostas:

a. São compostos pelas interfaces WindowListener, KeyListener, ActionListener e outros.

b. São interfaces que necessitam ser implementadas na nova classe.

c. Permitem, somente, o gerenciamento dos componentes da biblioteca AWT.

d.

São classes que aceitam qualquer quantidade de componentes, determinando, assim, os tamanhos destes componentes.

☒ e. São classes que tem por objetivo gerenciar a disposição e o arranjo dos componentes em uma interface gráfica.

Comentário da  
resposta:

Resposta: E

Comentário: os "Gerenciadores de *Layout*" são os componentes que definem o *layout* de um *container* como o painel ou a janela. Desta forma, esses elementos são classes da biblioteca do AWT, como, por exemplo: o BorderLayout, o GridLayout, o FlowLayout, entre outros e determinam a forma com que os componentes são distribuídos pelo *container* (como aparecerão na tela). Para cada um deles, existem regras a serem seguidas, a fim de que funcione corretamente.

#### Pergunta 4

0,25 em 0,25 pontos



A linguagem JAVA é uma linguagem do Paradigma Orientado a Objetos, a qual tem esses objetos como a base de seu funcionamento. Neste sentido, o desenvolvimento de sistemas em JAVA se baseia na criação das classes, os seus atributos e seus métodos. Uma interface gráfica GUI (*Graphical User Interface*) possui bibliotecas de classes que permitem a construção de sistemas *desktop* em que há a interação com o usuário, de forma que a todo *container*, define-se um *layout* específico que vai determinar como os componentes estarão nele dispostos. Cada tipo de *layout* é uma classe específica da biblioteca do Java, que possui características próprias, e que definem, inclusive, como os componentes podem ser adicionados à tela. Quanto às características de *layouts*, podemos afirmar que:

I. O BorderLayout permite que insiramos os componentes através dos métodos setBounds(...), setLocation(...) ou setSize(...), cada um em bordas específicas;

II. O GridLayout divide o *container* em partes iguais, dispostas em linhas e colunas;

III. O CardLayout divide a tela em 5 partes (norte, sul, leste, oeste e centro), como se fossem cartas, cada uma disposta ao lado da outra;

IV. Quando definimos o *layout* como nulo, cada componente deve ser adicionado levando-se em consideração a posição do canto superior esquerdo do componente, o seu comprimento e a sua altura.

Estão corretas, apenas, as afirmações:

Resposta Selecionada:

☒ a. II e IV.

Respostas:

☒ a. II e IV.

b. I e II.

c. II e III.

d. I, II e III.

e. II, III e IV.

Comentário da  
resposta:

Resposta: A

Comentário: a afirmação I está incorreta pois o BorderLayout é um tipo de *layout* que divide a área em 5 partes (norte, sul, leste, oeste e central), sendo que, em cada parte, só é possível inserir um componente que pode até ser um *container*. Além disto, os métodos elencados na afirmação são métodos que definem o tamanho e a localização do componente, mas que só são utilizados em casos específicos, como para o *layout* nulo.

A afirmação II explica, corretamente, a utilidade do GridLayout.

A afirmação III está incorreta, pois o CardLayout é um tipo de *layout* que não divide a área do *container* em partes, mas sim, define que naquele *container* poderão ser mostrados vários painéis diferentes, cada um a seu tempo, sendo possível trocar de painel (trocar de tela) a partir do acionamento de algum evento específico.

A afirmação IV explica, corretamente, a utilidade de um *layout* nulo, já que, neste caso, a área do *container* não é dividida em partes, mas possibilita-se que cada componente seja colocado em qualquer localidade daquela área.

#### Pergunta 5

0,25 em 0,25 pontos



Para a construção da janela da imagem a seguir, foram utilizados vários componentes pertencentes à biblioteca Swing. Analise a imagem a seguir:

Fonte: autoria própria.

Assinale a alternativa em que todos os componentes nela listada estão presentes nesta janela:

Resposta Selecionada: ☒ d. JFrame, JRadioButton, JScrollPane, JMenuBar, JButton, JTable, JLabel, JMenuItem, JTextField, JMenu.

- Respostas:
- a. JRadioButton, JPanel, JMenuBar, JButton, JTable, JPopupMenu, JLabel, JTextField, JMenuItem, JComboBox.
  - b. JScrollPane, JMenuBar, JFrame, JButton, JTable, JLabel, JMenu, JTextArea, JCheckBox, JMenuItem.
  - c. JRadioButton, JScrollPane, JButton, DefaultTableModel, JLabel, JMenu, JPasswordField, JPanel, JChoice.
  - ☒ d. JFrame, JRadioButton, JScrollPane, JMenuBar, JButton, JTable, JLabel, JMenuItem, JTextField, JMenu.
  - e. JFrame, JCheckBox, JPanel, JButton, JTable, JPopupMenu, JList, JMenu, JTextArea, JMenuItem, JComboBox.

Comentário da resposta: Resposta: D  
Comentário: de acordo com a imagem da questão e os elementos listados nas alternativas disponíveis, não é possível identificar a presença dos seguintes componentes: JPopupMenu, JChoice, ... componentes esses que aparecem (um ou outro) nas outras alternativas.  
Observações:  
- O JFrame é a própria janela;  
- O JRadioButton são as opções de escolaridade do formulário;  
- O JScrollPane é o painel em que está inserido a tabela, já que, na imagem, aparece a barra de rolagem, na parte da tabela;  
- O JMenuBar está presente, já que se observam as opções de *menus* a serem selecionadas na imagem da janela;  
- O JButton são os botões "Gravar" e "Cancelar";  
- O JTable é a tabela montada na janela;  
- O JLabel são os rótulos que explicam cada campo do formulário (como o texto: nome, e-mail, curso etc.);  
- O JMenuItem pode ser observado como sendo o item de *menu* "Sair" que aparece na imagem;  
- O JTextField são os campos passíveis de serem preenchidos;  
- O JMenu pode ser observado como sendo o *menu* "Arquivo" que apresenta nova opção de *menu* (como a opção "Sair").

## Pergunta 6

0,25 em 0,25 pontos



Uma janela padrão, geralmente, possui *menus* que nos permitem selecionar uma opção dentre várias outras disponíveis, com a finalidade de facilitar a navegação por entre as diversas tarefas e ações que um sistema pode oferecer. As diversas classes utilizadas na construção de *menus* pertencem à biblioteca padrão do Java. Uma dessas classes permite que se crie um elemento que, ao ser selecionado (acionado na tela), oferece outras opções de *menus*. Essa classe é a classe:

Resposta Selecionada: ☒ c. JMenu.

- Respostas:
- a. JMenuOptions.
  - b. JMenuItem.
  - ☒ c. JMenu.
  - d. JMenuContext.
  - e. JMenuSelect.

Comentário da resposta: Resposta: C  
Comentário: todos os componentes de janela são classes de bibliotecas pré-instaladas com o próprio Java na sua instalação. (Sendo assim, para a construção de *menus* interativos na janela, as classes dessas bibliotecas que definem os elementos de *menus* que quando selecionadas na janela, mostram novas opções de *menu* (sem gerar alguma ação específica), são as classes:  
Menu – da biblioteca do AWT;  
JMenu – da biblioteca do Swing.

## Pergunta 7

0,25 em 0,25 pontos



Algumas classes de interfaces gráficas (GUI) são consideradas *containers*, e podem agrupar e disponibilizar um conjunto de componentes, assim como agrupar outros *containers*. Qual dos itens a seguir contém, na sua totalidade, componentes tanto da biblioteca AWT quanto da biblioteca Swing, que podem ser considerados como *containers*, para a disponibilização de componentes em uma interface gráfica?

Resposta Selecionada: ☒ b. Panel/JPanel/Frame/JFrame.

Respostas:

- a. Label/JLabel/Frame/JFrame.
- ☒ b. Panel/JPanel/Frame/JFrame.
- c. List/JList/Checkbox/JCheckBox.
- d. TextField/JTextField/TextArea/JTextArea.
- e. Container/JContainer/Component/JComponent.

Comentário da Resposta: B

resposta: Comentário: um *container* é um componente em que adicionamos vários outros componentes, para que possam coexistir em uma mesma área da janela. Desta forma, trabalha-se com dois *containers* que representam a janela, e eventuais áreas desta janela (que são os painéis) que contém outros componentes. A janela é construída a partir da classe Frame ou JFrame (dependendo da biblioteca utilizada). Já o painel é construído a partir das classes Panel ou JPanel.

## Pergunta 8

0,25 em 0,25 pontos



Ao posicionarmos os componentes na tela (quando estamos trabalhando com *layout* nulo, qual é o comando que, ao mesmo tempo, posiciona e, também, define o tamanho do componente?

Resposta Selecionada: ☒ a. componente.setBounds(...).

Respostas:

- ☒ a. componente.setBounds(...).
- b. componente.setSize(...).
- c. componente.setPosition(...).
- d. componente.setDouble(...).
- e. componente.setLocation(...).

Comentário da Resposta: A

resposta:

Comentário: quando trabalhamos com o *layout* nulo em uma janela, cada componente desta janela deve ser posicionado, assim como ter o seu tamanho determinado, via codificação enquanto se está montando a tela. Desta forma, podemos utilizar, para isso, alguns comandos (métodos), sendo que cada um tem a sua particularidade. Imagine que o objeto "componente" represente algum componente a ser posicionado na tela (como um botão, por exemplo). O comando "componente.setSize(hor, ver)", com dois parâmetros ("hor" e "ver") que são números inteiros, define, apenas, o tamanho do componente, onde o primeiro parâmetro "hor" define o tamanho horizontal do componente (em *pixels*), e o segundo parâmetro "ver" define o tamanho vertical do componente (em *pixels*). O comando componente.setLocation(x, y), com dois parâmetros ("x" e "y") que são números inteiros, define, apenas, a localização do componente no *container*, onde os parâmetros "x" e "y" definem (em *pixels*) a posição do canto superior esquerdo do componente, levando-se em consideração que a posição (0,0) é o canto superior esquerdo do *container* onde aquele componente foi inserido). Desta forma, existe um comando (um método) que permite inserir em seus parâmetros, tanto a posição quanto o tamanho do componente, sendo este comando: componente.setBounds(x, y, hor, ver) – com a mesma explicação dos parâmetros descrita para os comandos anteriores.

## Pergunta 9

0,25 em 0,25 pontos



Analise o programa a seguir. Este programa pretende montar, na totalidade, a janela que aparece na imagem a seguir (já que lhe faltam alguns comandos):

```
import javax.swing.*;

public class Janela extends JFrame{
    public JMenuBar mb = new JMenuBar();
    public JMenu m1 = new JMenu("Arquivo");
    public JMenuItem m2 = new JMenuItem("Novo");
    public JMenuItem m3 = new JMenuItem("Sair");
    public JMenu m4 = new JMenu("Editar");
    public JMenuItem m5 = new JMenuItem("Copiar");
    public JMenuItem m6 = new JMenuItem("Colar");
    public JMenu m7 = new JMenu("Ajuda");
    public JMenuItem m8 = new JMenuItem("Conteúdo");
```

```

public static void main(String[] args) {
    Janela jan = new Janela();
}
public Janela() {
    this.setBounds(100, 100, 300, 200);
    mb.add(m1);
    mb.add(m4);
    mb.add(m7);

    // ??? comandos que faltam ???

    this.setJMenuBar(mb);
    this.setVisible(true);
}
}

```



Fonte: autoria própria.

Qual dos itens a seguir mostra o conjunto de comandos que completam o programa anterior de forma a montar, corretamente, o quadro de *menus*, da janela da imagem anterior?

Resposta Selecionada: ☒ e. m1.add(m2); m1.add(m3); m4.add(m5); m4.add(m6); m7.add(m8).

Respostas:

- a. m7.add(m2); m7.add(m3); m4.add(m5); m4.add(m6); m1.add(m8).
- b. mb.add(m2); mb.add(m3); mb.add(m5); mb.add(m6); mb.add(m8).
- c. m1.add(m2); m1.add(m3); m1.add(m5); m1.add(m6); m1.add(m8).
- d. m1.add(m2); m2.add(m3); m3.add(m5); m4.add(m6); m5.add(m8).
- ☒ e. m1.add(m2); m1.add(m3); m4.add(m5); m4.add(m6); m7.add(m8).

Comentário da resposta:

Resposta: E

Comentário: para a montagem dos *menus* de uma janela, a ordem com que se adiciona um componente de *menu*, ao longo da criação (codificação) do método que irá montar os *menus*, terá impacto direto na sequência de *menus* que será, efetivamente, apresentada na janela.

Sendo assim, o método "add(...)" só existirá para os elementos *Menu* ou *JMenu*, ou para os elementos *MenuBar* ou *JMenuBar*, que são componentes aos quais conseguimos adicionar outros componentes de *menu*. A ordem na qual se deve adicionar os componentes de *menu*, deve ser:

- Da esquerda para a direita (para a Barra de *Menu*);
- De cima para baixo (para um *Menu*).

De acordo com o código do programa apresentado no enunciado, apenas, os elementos da Barra de *Menu* haviam sido, inicialmente, adicionados, que são os *menus*: "Arquivo" (que é o objeto m1), "Editar" (que é o objeto m4) e "Ajuda" (que é o objeto m7), de forma que os seus *submenus* não haviam sido adicionados aos *menus*. Sendo assim, de acordo com a imagem da janela:

- Para o *menu* "Arquivo", deve-se adicionar os elementos "novo" (que é o objeto m2) e "sair" (que é o objeto m3) fazendo-se:  
m1.add(m2);  
m1.add(m3);  
... observe que os componentes m2 e m3 são adicionados ao componente m1.

- Para o *menu* "Editar", deve-se adicionar os elementos "copiar" (que é o objeto m5) e "colar" (que é o objeto m6) fazendo-se:  
m4.add(m5);  
m4.add(m6);

... observe que os componentes m5 e m6 são adicionados ao componente m4.

- Para o *menu* "Ajuda", deve-se adicionar o elemento "Conteúdo" (que é o objeto m8) fazendo-se:  
m7.add(m8);

... observe que o componente m8 é adicionado ao componente m7.

E com esses acréscimos sendo feitos ao programa inicial, o quadro de *menus* será criado de acordo com o modelo que aparece na imagem do enunciado.

## Pergunta 10

0,25 em 0,25 pontos



Com relação à criação de tabelas, como os componentes de uma janela, se a quantidade de linhas ultrapassar o espaço disponível de acordo com o tamanho da janela, como podemos fazer para que apareça o recurso de "Barra de Rolagem" a fim de permitir a visualização de todo o conteúdo da tabela (já que esta é uma tabela muito grande)?

Resposta

Selecionada:

- ☒ c. Habilita a visualização da barra de rolagem com o método showScrollBar() existente na classe Table.

Respostas:

a. Insere-se a barra de rolagem a partir do método `insertScrollBar()` existente na classe `Table`.

b.

Cria-se a tabela com todo o seu conteúdo e, por fim, define-se que o objeto da classe `Table` (que representa a tabela) existirá em um `JScrollPane`, bastando instanciá-lo tendo o objeto da classe `Table` como parâmetro.

☒ c. Habilita a visualização da barra de rolagem com o método `showScrollBar()` existente na classe `Table`.

d. Adiciona-se, juntamente com o objeto da classe `Table`, um objeto da classe `ScrollBar`.

e.

Não é possível mostrar a barra de rolagem, de forma que, para se determinar o tamanho da janela, deve-se prever a existência de uma tabela grande.

Comentário  
da resposta:

Resposta: C

Comentário: com a linguagem Java, na geração de tabelas em janelas de programas para o *desktop*, para que a barra de rolagem apareça com a tabela, utilizamos um recurso próprio do Java, que é utilizar um elemento que equivale a um painel com barras de rolagens, dentro do qual será inserida a tabela. Desta forma, instancia-se este “painel com a barra de rolagem” (cujo tipo é a classe do componente `JScrollPane`), instanciando-a e colocando-a como parâmetro da instância o objeto `JTable`, criado e já preenchido com os dados a serem mostrados na tabela. Assim, a tabela aparecerá dentro deste painel com as barras de rolagem.

← OK