Interfețe grafice Swing

Swing Graphical User Interfaces

Obiective:
- implementarea intefețelor grafice Swing;

Objectives:
- implementing Swing graphical user interfaces;

Biblioteca Swing este definită în JFC (Java Foundation Classes) și oferă programatorilor componente noi de unde și posibilitatea de a crea interfeţe grafice complexe.

Swing îmbunătăţeşte pachetul AWT (Abstract Window Toolkit), nefiind complet separat de el. AWT este folosit ca şi un contact intre Swing şi sistemul de operare pe care Java este instalat.

Caracteristici principale Swing:

- Modul de prezentare şi comportamentul interfeţelor nu depind de platformă aşa cum e cazul in AWT
- Există posibilitatea de a avea componente nu neapărat rectangulare
- Accesibilitate pentru persoanele cu nevoi speciale
- Viteza de lucru este mai mare folosind convenţii logice pentru numele claselor şi ale metodelor
- Tratarea evenimentelor = similară cu mecanismele folosite in AWT

Principalele componente folosite în programarea Swing sunt prezentate în următorul tabel:

Tip componentă	**Explicaţii**	**Componente**
Containere de bază	Orice aplicaţie care deţine o interfaţă grafică are cel puţin un astfel de container. Componentele (care nu pot fi tot containere de bază) se vor adăuga acestui container. Reprezintă locul în care toate celelalte componente se vor desena.	*JFrame*, *JWindow*, *JDialog*, *JApplet*
Containere intermediare	Scopul lor este acela de a grupa mai multe componente şi de a simplifica poziţionarea, dimensiunile şi modul în care componentele conţinute vor reacţiona la modificarea dimensiunilor ferestrei de bază. Pentru această li se ataşază un Layout Manager.	*JPanel*, *JScrollPane*, *JSplitPane*, *JTabedPane*

Componente atomice	Nu au, precum celelalte tipuri de componente, rolul de a ţine alte componente, şi sunt suficiente pentru a prezenta informaţia pe care o deţin utilizatorului sau pentru a primi informaţii de la utilizatorii interfeţei grafice. Permit interacţiunea cu utilizatorul.	*JLabel*, *JButton*, *JList*, *JComboBox*, *JTextField*, *JTextArea*, *JTable*, *JTree*, *JMenu*, *JPopup*, *JMenuItem*, *JToolBar*, *JOptionPane*, *JFileChooser*, *JColorChooser*, *JSlider*, *JScrollBar*, *JProgressBar*, *JDesktopPane*, *JInternalFrame*, etc.

https://docs.oracle.com/javase/8/docs/api/javax/swing/***.html (unde *** se inlocuiește cu numele clasei căutate (*JFrame*, etc.))

Lucru individual

1. Scrieți o aplicație Java folosind biblioteca Swing în cadrul căreia sunt afișate 3 componente de tip checkbox (*JCheckBox*), 3 componente de tip radio button (*JRadioButton*) și 3 componente de tip drop-down list (*JComboBox*). Selecțiile făcute de utilizator vor fi afișate într-un câmp de tip etichetă (*JLabel*).

==========

2. Scrieți o aplicație Java ce permite prin intermediul bibliotecii Swing selectarea unui fișier din sistemul local de fișiere. Se pot selecta mai multe fișiere, iar denumirea lor alături de calea completă va fi afișată într-o componentă de tip etichetă (*JLabel*).

==========

3. Scrieți o aplicație Java – Swing ce permite selectarea unei imagini stocate local și afișarea acesteia în cadrul interfeței grafice. La apăsarea unui buton, imaginea va fi ștearsă din interfața grafică.

==========

4. Scrieți o aplicație Java ce utilizează componenta *JTabbedPane* și care permite desenarea în prima fereastră a unei figuri geometrice ale cărei caracteristici sunt definite într-o a doua fereastră - tipul figurii, dimensiuni, mod de umplere (butoane radio), culoare (triplet RGB).

==========

5. Implementați o aplicație Java bazată pe biblioteca Swing ce criptează în timp real datele introduse într-un câmp de timp *JTextArea* folosind codul lui Cesar (https://en.wikipedia.org/wiki/Caesar_cipher). Deplasarea caracterelor va fi stabilită dintr-un câmp de tip drop-down list (max. 10).

==========

6. Implementați o aplicație Java bazată pe biblioteca Swing ce simulează o agendă telefonică. Utilizatorul are opțiunea de a adăuga un nou contact cu nume, prenume și număr de telefon. Odată adăugat acest contact, el va fi disponibil într-o listă de contacte. În dreptul fiecărui contact va fi afișat un buton ce permite ștergerea sa.

==========

7. Scrieți o aplicație Java de tip Swing care în fereastra principală (*JFrame*) afișează 3 componente de selecție simplă (*JComboBox*) care afișează cifrele între 0 și 9. Combinația obținută prin selecție este comparată cu o valoare întreagă [000...999] generată aleator la pornirea programului. Fiecare nereușită este contorizată și urmată de afișarea unui mesaj (*JLabel*) care informează utilizatorul asupra relației dintre numărul format prin selecție și valoarea aleatoare generată (valoare prea mare, prea mică sau exactă). După reușită, componentele de selecție devin inactive.

==========

8. Scrieți o aplicație Java de tip Swing care permite introducerea unei parole într-un câmp de tip *JTextField*. Caracterele sunt afișate cu *. Complexitatea parolei este dată de lungime (>8 caractere) și de îndeplinirea următoarelor criterii: să aibă cel puțin o cifră, un caracter majuscul și un caracter special (altceva decât cifră sau literă). În funcție de complexitate, un pătrat desenat lângă campul de parolă este colorat cu diferite nuanțe (roșu-portocaliu-galben-verde) pe măsură ce utilizatorul scrie caracterele. Un buton care afisează un ochi permite vizualizarea în clar a parolei introduse atât timp cât este ținut apăsat.

Individual work

1. Write a Swing-based Java application which has 3 check boxes (*JCheckBox*), 3 radio buttons (*JRadioButton*) and 3 drop-down lists (*JComboBox*). A label component (*JLabel*) will display all the choices made by the user.

==========

2. Write a Swing-based Java application which enbles the user to select files from the local file system. More than one file can be selected, and the names of the files along with their path will be displayed in a label component (*JLabel*).

==========

3. Write a Java – Swing application which enables to user to select an image from the local file system and displays it in the GUI. At the press of a button, the image will be deleted.

==========

4. Write an application which uses the *JTabbedPane* component. In one of the panes a geometrical shape is drawn. Its characteristics are defined in the second pane – the type of shape, its dimensions, the fill type (radio buttons) and its color (RGB triplet).

==========

5. Write a Swing-based Java application which encrypts in real time the text entered in a *JTextArea* using Caesar's cipher (https://en.wikipedia.org/wiki/Caesar_cipher). The shift of the algorithm is set through a drop-down list (max. 10).

==========

6. Write a Swing-based Java application which simulates a phone contacts app. The user can add a new contact with its name, surname and phone number. Once the contact is added, it will be displayed in a contact list. Next to each contact there is also a button which deletes the selected contact.

==========

7. Write a Swing-based Java application which in the main window (*JFrame*) shows 3 *JComboBox*es displaying the digits from 0 to 9. The resulting combination compared with a randomly generated value between 000 and 999. The task of the user is to guess the random combination. The GUI will display an appropriate message at each try (lower, higher or equal). After the user guesses the combination, the selection components are disabled.

==========

8. Write a Swing-based Java application which enables the user to input a password in *JTextField* component. The characters are shown as *. The complexity of the password should have the following restrictions: length > 8 characters, 1 digit, 1 uppercase character and a special character (other than letters and digits). Given the complexity, a square drawn next to the password input field is colored accordingly (red – orange – yellow – green) as the user types the password. Also, a button displaying an eye image enables the user to view the actual characters from the typed password as long as the button is pressed.