Interfete grafice

Introducere

In acest laborator vorbim despre interfetele grafice realizate cu Java Swing. O interfata grafica este conceptual vorbind similara cu o interfata java, adica reprezinta un “contract” prin care utilizatorul poate interactiona cu aplicatia noastra folosind mouseul si tastatura pentru a trimite informatii si monitorul pentru a primi raspunsuri.
Cele mai simple interfete grafice erau la inceput simple listari de meniuri (ca text), concept mostenit de pe vremea consolelor printate pe hartie, nu pe ecran. In timp ele au evoluat tot mai mult. In ziua de astazi, orice pixel de pe ecran poate fi accesat si controlat, deci teoretic oricine poate sa faca orice fel de interfata grafica doreste, controland direct pixelii.
Este insa greu sa lucram la nivel de pixel, si din acest motiv au aparut tehnologii care introduc concepte de nivel inalt cum ar fi “desenarea” unui text, a unei ferestre, a unui buton, a unei imagini si, de ce nu, randari a unor scene 3D.
Initial, sistemul de operare detinea toate resursele sistemului si oferea un API (Application Programming Interface) prin care programatorul putea sa ii ceara sa construiasca ferestre sau sa faca alte diverse lucruri. Acest lucru era permis de AWT, care era un API de java facut special pentru a accesa APIul sistemului. Ulterior, au fost constatate anumite probleme, mai precis sistemele de operare au inceput sa difere in capabilitati destul de mult sau chiar in comportament si atunci programele Java trebuiau construite pentru un sistem sau altul, ceea ce era practic anti-java.

Dupa AWT a aparut Swing, care este practic construit peste AWT. AWT nu se ocupa efectiv de desenarea pixelilor pe ecran (in general), ci solicita sistemului de operare “deseneaza-mi aici o fereastra care arata asa”. Swing insa foloseste destul de putine apeluri la APIul sistemului ci se ocupa el de desenarea pixelilor, astfel incat un program o sa arate similar pe orice sistem de operare. Din acest motiv avem ceva mai multe posibilitati cand e vorba de customizarea diverselor elemente grafice (codul e la noi).
Dupa Swing a aparut JavaFX care este de fapt o tehnologie moderna (cea mai moderna) pentru generarea interfetelor grafice.
Noi vom folosi Swing pentru ca este mai simplu de inteles si urmarit, si cu un minim efort o persoana care a inteles Swing poate sa treaca la JavaFX fara probleme.

Componete grafice

Atunci cand lucram cu o interfata grafica, asa cum am spus mai sus, nu vrem sa desenam direct pixeli, ci vrem sa desenam elemente complexe, elemente pe care le numim componente.
Componenta de baza intr-o aplicatie Swing este JFrame, care reprezinta o fereastra a sistemului de operare. Voi reveni cu detalii.
Aceasta componenta, JFrame, este numita si un Container deoarece poate sa contina in interiorul ei alte componente. (**Obs: explicatiile de aici sunt simplificate pentru a putea exemplifica concepte! Informatii complete primiti la curs!** de exemplu, exista o diferenta intre un container generic si un top-level container, cum este JFrame).
Alte componente sunt de exemplu, JTextArea, JTextField, JButton etc. Puteti vedea o lista de componente si modul in care puteti sa le folositi aici:
https://docs.oracle.com/javase/tutorial/uiswing/components/componentlist.html

Va rog sa ignorati Appleturile, deoarece nu mai sunt utilizate.

Desigur, am dori ca userul sa poata interactiona cu aceste componente, iar Swing ne ofera o multitudine de moduri in care sa putem defini modul in care se poate interactiona cu ele, in functie de tipul componentei.

Pentru a construi o ierarhie (componente in componente) putem folosi componenta JPanel, care este un Container la randul ei.

Toate aceste componente trebuie sa apara pe ecran, pana la urma asta este ceea ce dorim, si metoda responsabila cu asta este metoda “public void paint(Graphics graphics)”. De cele mai multe ori nu vrem sa modificam modul in care desenam o componenta, asta este si ideea adica sa nu ne batem noi capul cu desenarea, dar daca vrem desenare “custom” nu avem de ales. De exemplu, daca vrem ca un buton sa nu arate ca un buton ci sa fie o imagine cu o bordura ciudata in forma neregulata, trebuie sa construim o subclasa a lui JButton si sa redefinim in ea metoda paint, care sa deseneze butonul.
Puteti vedea detalii suplimentare tot pe siteul Oracle:
https://docs.oracle.com/javase/tutorial/uiswing/painting/index.html

Layout managers

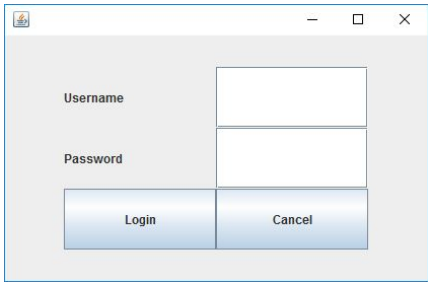
Asezarea componentelor intr-un container este un element foarte important. Sa spunem ca avem o fereastra si in ea vrem sa facem un simplu formular de login. Cum pozitionam etichetele pentru cele doua campuri, username si parola? Unde punem butonul Login? Cat de mare e acest buton? Cum se modifica pozitia si dimensiunea acestor componente in functie de redimensionarea ferestrei?

Varianta ce mai simpla ar fi sa aruncam pur si simplu niste componente in container, cu niste dimensiuni si pozitii fixe astfel incat sa arate bine fereastra. Dar daca userul redimensioneaza fereastra, trebuie sa procesam acest eveniment si sa reactionam la el, adica sa redimensionam noi componentele, sa le repozitionam etc. Desigur, treaba este foarte complexa chiar si pentru aceasta interfata simpla, si poate deveni mult mai complexa pentru o interfata incarcata de componente.

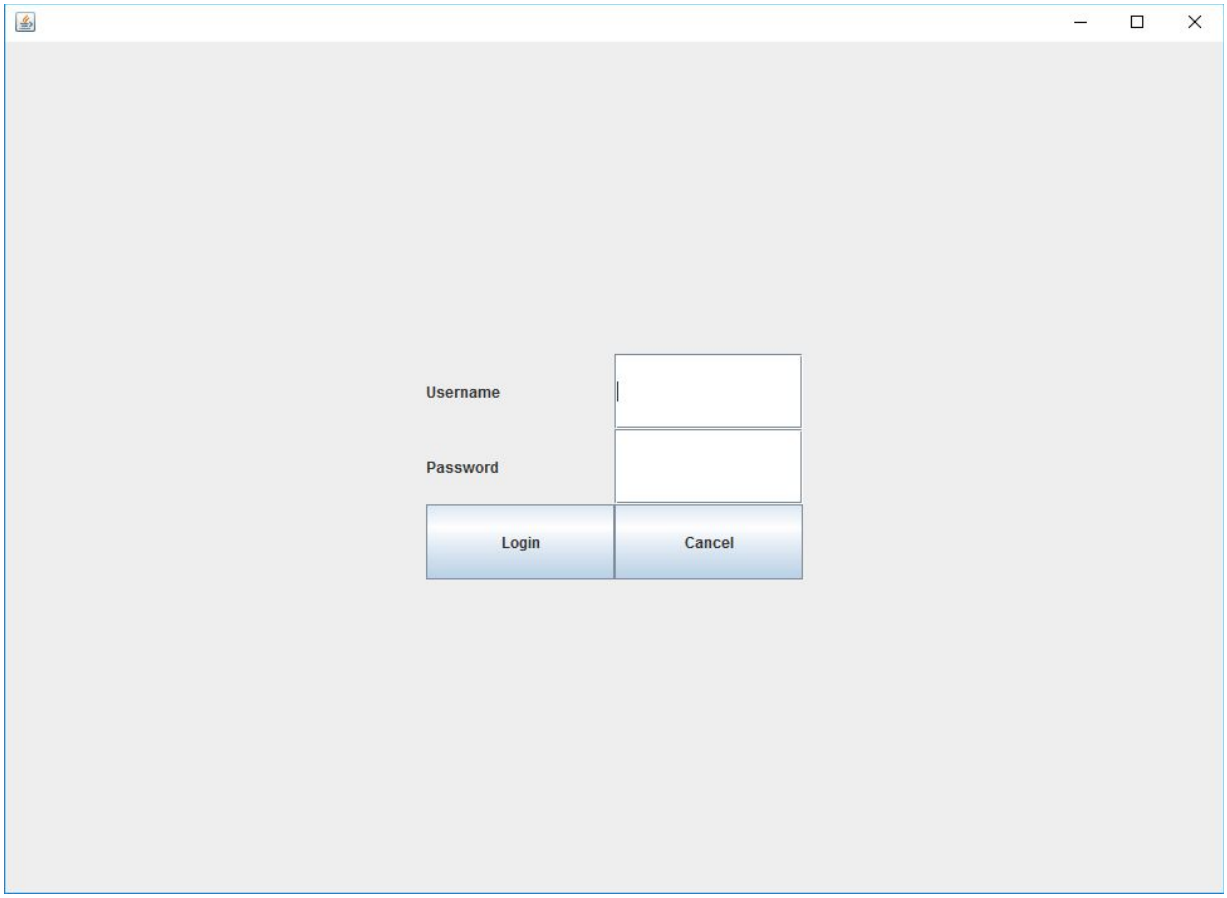
Un layout manager este o runtina care ne rezolva exact aceasta problema. Tot ce facem noi este sa facem un plan al modului in care componentele noastre vrem sa raspunda la resize, sa alegem un layout manager corespunzator si sa il instruim cum sa plaseze componentele in spatiul alocat. Unul din exercitiile foarte importante pe care le vom face este sa construim un Layout, adica sa facem un plan de pozitionare a componentelor “pe hartie” inainte de a scrie cod. Dar pentru a face acest lucru, trebuie sa stim ce instrumente avem la dispozitie, adica ce LayoutManagers exista. Iata aici lista completa:
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
Din aceasta lista, cel mai puternic LayoutManager este GridBagLayout.

Exercitiu 1:

Faceti o aplicatie care deschide o fereastra de login si arata asa:



Componentele din interior sunt mereu centrate in fereastra si de dimensiune fixa.



Pentru a construi fereastra initiala, construim o noua instanta a unui JFrame.
Apoi trebuie sa construim un nou JPanel care sa fie setat ca “contentPane” pentru JFrame.
Pentru a seta un layout manager pentru acest JPanel folosing metoda setLayout pe acest JPanel cu un argument de tipul corespunzator (de exemplu: new BorderLayout()).
Pentru etichete vom folosi new JLabel(“text aici”);
Pentru campul de username vom folosi new JTextField();
Pentru campul de password vom folosi new JButton();
Pentru butoane vom folosi new JButton(“text pe buton”);
Pentru a adauga o componenta la un container folosim numeContainer.add(componenta) sau numeContainer.add(componenta, reguliDePlasare) in functie de layout managerul folosit.

Action listeners

Majoritatea componentelor suporta interactiunea utilizatorului cu ele. Interactiunea porneste de la un simplu “mouse over” pana la click, drag etc.
Vom discuta despre cea mai simpla interactiune, un click pe un buton, dar va invit sa cautati pentru fiecare componenta interactiunile posibile pe pagina lor de documentatie.
https://docs.oracle.com/javase/tutorial/uiswing/events/eventsandcomponents.html

Pentru a raspunde la click pe un buton trebuie sa setam un ActionListener pentru butonul respectiv.
ActionListener este o interfata cu o singura metoda void numita actionPerformed, cu un parametru de tipul ActionEvent care de ofera detalii despre eveniment.
Codul din aceasta metoda va fi executat cand se face click pe buton.

Toti listenerii folosesc o logica similara: implementarea lor defineste o bucata de cod care se executa cand userul interactioneaza intr-un anumit mod cu componenta la care sunt adaugati.

De cele mai multe ori, listenerii sunt construiti ca o clasa anonima, dar asta nu este o regula.

Exercitiu 2:

Modificati exemplu de mai sus astfel incat la click pe butonul login sa dispara toate componentele si sa apara in locul lor doar un text cu “Bine ai venit XXXXX” unde XXXXX este numele utilizatorului introdus in campul username.
Pentru asta va trebui sa eliminam din panoul central toate componentele precedente si sa adaugam o componenta noua.
Pentru a elimina toate componentele folosim metoda removeAll().
Dupa ce modificam continutul unui panel (sau al oricarui container) trebuie sa chemam metodele “revalidate”, pentru ca layout managerul sa repozitioneze componentele, si “repaint” pentru a redesena pixelii din zona componentei.

Exercitiu 3:

Faceti ca la click pe Cancel sa se inchida programul.

Exercitiu 4:

Implementati interfata grafica din laboratorul 5
https://docs.google.com/document/d/1zmWRKaocTRtmlmZkFcQsABXFzPfNqqJbvr0nT3X26yI/edit?usp=sharing

Exercitiu 5:

Implementati interfata grafica din laboratorul 4
https://docs.google.com/document/d/18kcjbhrt1FsneMZ_zyHRWMK3Xk5pKSTjfxmQZkKkZxc/edit?usp=sharing

O ultima observatie…

Pentru cei care se intreaba de ce facem Java Swing in loc de Java FX, vreau sa fac urmatoare observatie. Majoritatea studentilor nu sunt pregatiti pentru a sari direct la JavaFX. Swing, desi este practic inutil pentru ca nu se mai foloseste, este doar un punct intermediar la care se poate ajunge usor si este lesne de inteles. Odata ce ati inteles cum se gandeste un Layout, JavaFX devinde un simplu exercitiu de adaptare la o librarie externa.

Cu alte cuvinte daca vreti sa va suiti pe acoperisul unei case (sa invatati cea mai recenta tehnologie) puteti sa incercati sa sariti direct si daca reusiti castigati timp, daca nu, s-ar putea sa va ia o durere de cap si eventual sa renuntati cu totul. In schimb, daca veti pune o cutie langa casa si va veti sui pe ea (Java Swing) ca punct intermediar, veti pierde un pic mai mult timp, dar sigur veti putea sa va suiti si pe acoperis. Datorita progresului in tehnologie din ziua de azi, casele (tehnologiile) incep sa fie tot mai inalte (complexe) si de cele mai multe ori trebuie sa apelati la puncte intermediare (tehnologii mai vechi) pentru a va atinge scopul.