Poo Laboratoare[1]

Contents

[1] http://www.google.ro/

Laborator10

Programare Orientată pe Obiecte: Laborator 10

1 Dezvoltarea unei aplicații cu interfață grafică

1.1 Overview

În cazul programelor pe care le-ați făcut până acum, toate mesajele și răspunsurile apăreau ca linii de text succesive, care defilau pe ecran (ecranul era folosit în mod text). Un astfel de stil de comunicare nu este atractiv pentru utilizatori, motiv pentru care se preferă dialogul prin interfețe grafice sau **GUI (Graphical User Interface)** (ecranul este folosit în mod grafic).

Componentele unei interfețe grafice se numesc elemente de control (**widgets**) și pot fi activate de către utilizator cu ajutorul mouse-ului sau al tastaturii. Activarea unui element de control determină un anumit răspuns din partea programului, răspuns concretizat prin execuția unei anumite operații. Cele mai răspândite interfețe grafice sunt cele bazate pe ferestre, adică fiecărei aplicații active la un moment dat i se alocă o zonă dreptunghiulară pe ecran, numită fereastră. În interiorul ferestrei respective, se vor găsi toate elementele de control necesare dialogului utilizator-aplicație, precum și informațiile afișate de aplicație.

Pentru ca o aplicație care folosește interfața grafică să poată funcționa, este necesară existența unei platforme sau server grafic care să determine intrarea monitorului în regim grafic și care să ofere funcțiile primitive necesare, între altele, pentru desenarea componentelor GUI și pentru receptarea semnalelor generate de mouse și tastatură. În sprijinul programatorilor au fost create suporturi software care permit ca o aplicație grafică să poată fi construită din "piese" standardizate, gata create.

1.2 Componente

De regulă, un suport software pune la dispoziția programatorului următoarele elemente:

- **o bibliotecă de elemente de control** (bare de defilare, chenare, butoane etc.). Această bibliotecă eliberează programatorul de sarcina de a desena și colora de fiecare dată toate elementele GUI ale unei aplicații
- **un manager de ferestre**: acesta este cel care impune cum "arată" elementele GUI, dictează modul în care este asamblată și echipată o fereastră, precum și modul în care va fi manipulată. Mediul Windows are încorporat un astfel de manager de ferestre, care definește aspectul general al unei ferestre Windows. Mediul Linux este mai configurabil din acest punct de vedere, permițând utilizatorului o gamă mai variată de manageri de ferestre: **GNOME**, **KDE**, **FVWM**, etc.
- **o bibliotecă de funcții sau clase predefinite**. Această bibliotecă permite construirea interfeței grafice, pornind de la fereastra aplicației, conform standardului impus de managerul de ferestre instalat pe mașina pe care rulează aplicația. Limbajul Java 2 oferă pachetele predefinite java.awt și javax.swing, împreună cu o serie de subpachete ale acestora, care încorporează clasele necesare pentru a construi aplicații grafice.

În cele ce urmează veți dezvolta o aplicație grafică completă pentru a observa cât de ușor se poate face acest lucru folosind facilitățile puse la dispoziție de **Swing**.

2 Probleme de laborator

Observație! Sa va folosi aplicația **Calculator** din sistemul MS-Windows (**Start-Accessories-Calculator**), inclusiv operațiile cu memoria (tastele care conțin litera *M*).

2.1 Problema 1

(2.5 puncte) Să se dezvolte treptat un program care să realizeze principalele funcţii ale aplicaţiei **Calculator** cu următoarele simplificări:

- Nu se folosesc butoane pentru cifre zecimale, pentru semne şi punct zecimal (numerele se pot introduce numai prin taste)
- Nu se folosesc butoane pentru funcţii şi nici butonul **Backspace**
- Nu se foloseşte butonul **CE** (doar butonul **C** pentru ştergere)

Pentru aceasta se va realiza mai întâi aşezarea obiectelor vizuale în fereastră. Se vor folosi panouri intermediare pentru următoarea aşezare:

- linia 1: câmpul text (de lungime 20)
- linia 2: butonul **M** (dezactivat), butoanele **C** şi =
- linia 3: butoanele /, *, -, +
- linia 4: butoanele **MC**, **MR**, **MS**, **M+**

Butonul **M** îşi schimbă inscripţia (iniţial patru spaţii) atunci când memoria conţine o valoare nenulă (inscripţia devine **M**). Dimensiunea ferestrei principale este (280, 280). Dezactivarea butonului se face cu metoda **setEnabled(false)**.

2.2 Problema 2

(2.5 puncte) Să se trateze evenimentele de la butoanele + (adunare), - (scădere), * (înmulţire), / (împărţire), = (calcul şi afişare rezultat) şi **C** (şterge număr afişat şi rezultat). Pentru toate cele patru butoane cu operaţii aritmetice se va defini un singur ascultător, care memorează codul operaţiei. Operaţia este efectuată la acţionarea butonului =. După orice acţionare de buton focalizarea revine pe câmpul text (se va folosi metoda **requestFocus()**).

Alte indicaţii:

- la împărţire prin zero se afişează în câmpul text mesajul *Divide by Zero*
- ştergerea câmpului text se face la acţionarea butoanelor +, -, *, /
- conversia de la **String** la **float** se face cu metoda **Float.parseFloat()**

2.3 Problema 3

(2.5 puncte) Să se adauge tratarea de evenimente de la butoanele **MS** (pune rezultat în memorie), **MR** (readuce din memorie în rezultat parţial şi afişează), **MC** (şterge memorie), **M+** (adună rezultat parţial la numărul din memorie).

2.4 Problema 4

(2.5 puncte) Să se adauge programului posibilitatea ca ştergerea câmpului text să nu se facă decât la introducerea primei cifre din noul numar (numărul introdus rămâne afişat şi după acţionarea unui buton de operaţie aritmetică). În acest scop trebuie tratat evenimentul de tasta acţionată într-un ascultător de tip **KeyListener** la câmpul text. Se poate extinde clasa **KeyAdapter** şi redefini metoda **keyTyped()**.