

Note di rilascio del simulatore CalcBin, versione 1.0

Architettura del software

Il progetto (JAVA) è caratterizzato dai file CalcBinApp.java, Calcoli.java, CalcBinView.java, CalcBinSimulatore.java , Simulatore.java, jTextFieldLimit.java, jTextFieldLimitResto.java, CalcBinAboutBox.java.

Nel file CalcBinApp.java troviamo il “main” che inizializza i due oggetti rispettivamente della classe simulatore e calcoli, il metodo startup() nel quale inizializziamo l’istanza dell’oggetto della classe CalcbinView, passandolo come parametro del metodo show() ereditato dalla classe SingleFrameApplication (della quale CalcBinApp è un’estensione), e due metodi public showSimulatore() e showHome() .

Nella “public class Calcoli” troviamo gli attributi e variabili contenenti i numeri acquisiti in run time per gestire le operazioni della calcolatrice.

I metodi più importanti riguardano le quattro operazioni in binario (AddizioneBin, SottrazioneBin, MoltiplicazioneBin, DivisioneBin) e le conversioni, da decimale a binario (dec2bin) e da binario a decimale (bin2dec).

Abbiamo dovuto creare dei metodi appositi per ogni operazione poichè,sono necessari per la visualizzazione passo passo delle varie operazioni. Inoltre, i metodi “risultato” e “visualizza” permettono di visualizzare il risultato nella TextBox della calcolatrice.

Il file CalcBinView.java gestisce la GUI con tutti i vari pulsanti della calcolatrice e l'incolonnamento. Attraverso l'IDE (NetBeans 7.0) è possibile distinguere la sezione “design”, dove è possibile operare graficamente sull'interfaccia attraverso semplici spostamenti delle

componenti grafiche(JButton, JTextBox, JLabel), e la sezione del codice.

Le variabili (private) e le librerie sono tutte dichiarate e importate direttamente da NetBeans e non devono essere modificate. Sono state applicate le opportune modifiche ai vari “Jbutton” , mentre il metodo aggiornaJLabel () viene utilizzato per gestire la visualizzazione delle operazioni in colonna, e viene richiamato dal metodo StepByStepActionPerformed() che permette di mostrare l’incolonnamento un passo alla volta. Il relativo JButton viene reso disponibile (visibile) solo dopo aver effettuato un’operazione, ed è gestito dai metodi dei bottoni delle operazioni e dell'uguale (buttonugualeActionPerformed, buttonpiùActionPerformed, buttonperActionPerformed, buttonmenoActionPerformed, buttondivisoActionPerformed). Il jBTeoria permette, mediante il relativo actionPerformed(), di aprire il pdf con le nozioni di teoria che il progetto si propone di esplicitare. Il jBSimulatore(ActionPerformed()) permette di collegarsi al CalcBinSimulatore richiamando il metodo showSimulatore() della classe CalcbinApp.

Il file CalcBinSimulatore.java rende un servizio al file Simulatore.java implementando un nuovo JPanel nel quale è possibile esercitarsi producendo operazioni random tramite l’apposito jButton1 (il cui testo è stato rinominato “genera operazione”).E’ possibile fissare una operazione per volta mediante l’apposito JradioButton . Il tutto viene gestito nel metodo jButton1ActionPerformed, settando o utilizzando, di volta in volta, le stringhe (primo, secondo, risultato) che identificano gli operandi e il risultato dell’operazione e sono definite nel file principale Simulatore.java . Tramite lo stesso metodo vengono rese visibili le opportune JTextfield (“jtTerzo”) nelle quali l’utente tenterà di inserire il risultato corretto. In alternativa, è possibile fissare una operazione per volta mediante l’apposito

JradioButton . Il jButtonSoluzioneActionPerformed() permette di visualizzare la soluzione dell’operazione mediante l’apposito pulsante. Il tasto jBVerifica permette di verificare l’esercizio svolto. Il

`JBHomeActionPerformed()` permette di tornare alla View principale richiamando il metodo, della classe `CalcBinApp`, `showHome()`, che richiama il pannello direttamente da `CalcBinView` (`getPanel()`). Le classi `jTextFieldLimit` e `jTextFieldLimitResto` estendono l'oggetto `textBox` ed effettuano controlli sull'input inserito.

Tecniche di programmazione

Abbiamo programmato utilizzando il linguaggio di programmazione, orientato agli oggetti, java. Per l'interfaccia grafica (GUI) ci siamo appoggiati al framework `java.swing`.

Collaudi

Il programma è stato testato su Windows 7/8 ma non dovrebbero esserci problemi nel caso si utilizzino sistemi Unix/Linux, Mac o versione di Windows antecedenti alla 7, purchè sia installato Java sulla macchina.

Idee per ulteriori sviluppi

Tra le possibili migliorie apportabili al software:

Aumentare il range delle possibili divisioni, visualizzabili tramite lo step by step.

Implementazione del reverse step by step.

Implementazione di operazioni con parentesi.

Miglioramenti apportabili alla modalità "simulatore":

- possibilità di inserimento dei risultati parziali inerenti alla moltiplicazione e divisione tramite l'aggiunta di ulteriori `jTextBox`

- aggiunta di pulsanti che agevolino l'utente nell'inserimento dei risultati parziali andando a compilare in automatico le righe contenenti solo bit 0 nel caso della moltiplicazione.
- aggiunta di pulsanti che agevolino l'utente nell'inserimento del divisore nelle sottrazioni parziali, nel caso delle divisioni.