Spesifikasi Tugas Besar 1
IF2210 Pemrograman Berorientasi Objek

Revisi

28 Februari 2020

2 Maret 2020

Deskripsi

Fahmi the Cabbage Merchant, 5 minutes before the disaster, circa 2020, colorized

Cabbage Merchant (yang sebenarnya bernama "Fahmi") selalu kesal karena kubisnya selalu dirusak oleh Avatar. Karena itu, ia ingin menuntut Avatar ganti rugi. Sebelum itu, dia akan menghitung berapa besar kerugiannya. Namun, ia tidak punya kalkulator karena semua kalkulatornya dibeli mahasiswa IF 2018 untuk ujian Probstat. Sebagai mahasiswa IF 2018 yang bertanggung jawab, Anda ingin membuatkan kalkulator untuk Fahmi.

Fungsionalitas

Kalkulator yang Anda buat minimal harus mampu memenuhi kebutuhan berikut:

1. Tombol 0 - 9 dan titik (.)
    a. Menuliskan angka atau titik ke layar
    b. Titik sebagai *decimal separator*
2. Tombol operator matematika
    a. Operasi matematika yang wajib difasilitasi antara lain penjumlahan (+), pengurangan (-), perkalian (x), pembagian (:), dan akar (√).
    b. Tombol pengurangan (-) juga berfungsi untuk menandakan bilangan negatif.
    c. **Bonus**: kalkulator mampu menggunakan operasi matematika lain (kurung, pemangkatan, fungsi trigonometri, dll).
3. Tombol perhitungan (=)

   a. Ketika diklik, kalkulator akan menghitung ekspresi di layar.
   b. Ekspresi di layar selalu berupa salah satu dari berikut:
      i. A ⊛ B dengan ⊛ merupakan operator matematika, A dan B merupakan bilangan real yang valid.
      ii. √C, dengan C merupakan bilangan real yang valid
   c. Bilangan real berikut juga merupakan bilangan yang valid:
      i. 2.10
      ii. -51
      iii. 0002.7000
   d. Sedangkan bilangan real berikut tidak valid:
      i. 51..3
      ii. --12
      iii. 5.1.2
   e. Mungkin saja, terjadi kesalahan perhitungan dan kesalahan sintaks. Jika demikian tampilkan error di layar beserta penyebab kesalahannya, misalkan pembagian dengan nol, akar dari bilangan negatif, dan lainnya (Anda tidak boleh hanya menuliskan "error" tanpa penjelasan penyebab kesalahannya).
   f. **Bonus**: kalkulator mampu menerima ekspresi dengan format apapun, tidak terbatas pada poin (b).
4. Tombol ans
   a. Menuliskan hasil perhitungan sebelumnya di layar.
   b. Dengan demikian, pengguna dapat melakukan perhitungan seperti berikut:
      i. Menekan tombol 2 (tertulis "2" di layar)
      ii. Menekan tombol + (tertulis "2+" di layar)
      iii. Menekan tombol 3 (tertulis "2+3" di layar)
      iv. Menekan tombol = (tertulis "5" di layar)
      v. Menekan tombol 7 (tertulis "7" di layar)
      vi. Menekan tombol x (tertulis "7x" di layar)
      vii. Menekan tombol ans (tertulis "7xans" atau "7x5 di layar)
      viii. Menekan tombol = (tertulis "35" di layar)
5. Tombol MC dan MR
   a. Tombol MC akan menyimpan nilai "saat ini" ke dalam history kalkulator
   b. Ketika tombol MR ditekan, nilai yang disimpan akan dikeluarkan ke layar
   c. Berbeda dengan kalkulator biasa, history di kalkulator Anda akan bertipe queue, sehingga mampu menyimpan beberapa nilai sekaligus
6. Tombol clear
   a. Layar akan menjadi kosong dan history perhitungan juga kosong

Panduan Pengerjaan

Kalkulator yang Anda buat harus menggunakan pemrograman berorientasi objek. Anda diperbolehkan memilih bahasa apapun, selama bahasa tersebut mampu memberikan tampilan GUI kalkulator (memiliki binding ke framework GUI) dan memfasilitasi paradigma Objek. Anda diwajibkan memanfaatkan konsep OOP di beberapa hal berikut:

- Penggunaan kelas Button untuk tiap tombol, yang memiliki method onClick untuk meng-*handle click event*
- Penggunaan kelas Ekspresi (seperti pada praktikum 4), beserta memanfaatkan polymorphism
- Penggunaan method atau operator overloading
- Penggunaan kelas Generic
- Penggunaan Exception
- Penggunaan Library queue untuk fitur memori

Selain dari contoh di atas, Anda diperbolehkan memanfaatkan konsep OOP di tempat lainnya. Jika ada keterbatasan dari segi bahasa, Anda juga boleh berkonsultasi dengan asisten.

Dalam pengerjaan, buatlah kode yang baik dengan menerapkan prinsip:

- DRY (Don't Repeat Yourself), tidak memiliki kode duplikat
- Memiliki struktur kelas yang mudah dipahami
- Dekomposisi yang baik dan implementasi yang tidak terlalu kompleks (sebuah method tidak terlalu panjang)
- Mengikuti prinsip SOL di SOLID
- Mengikuti konvensi penamaan yang baik sesuai bahasa Anda, misal:
    - Nama kelas berupa kata benda (noun) diawali huruf besar
    - Nama *method* diawali kata kerja (*verb*) diawali huruf kecil, menggunakan *camel case*

Karena semakin banyak pertanyaan mengenai bahasa yang diperbolehkan, berikut bahasa yang diperbolehkan (namun tidak terbatas pada bahasa berikut saja):

- C++
- Java
- Objective-C
- Swift
- Kotlin
- Smalltalk
- Ruby
- Python
- Object Pascal
- Visual Basic
- Golang

Jika ingin menggunakan bahasa lain selain bahasa di atas, silakan bertanya ke dosen terlebih dahulu. Untuk bahasa yg *multi paradigm*, Anda diwajibkan menggunakan paradigma *Object Oriented*. Framework berikut juga diperbolehkan:

- UWP
- WPF
- WinForms
- Gtk
- Tk
- WxWidgets
- Cocoa
- Java Swing
- Java AWT
- Qt

Sebagai tambahan, Anda **tidak** diperbolehkan menggunakan:

- Electron atau webview-based GUI framework lainnya

- Apapun yang menggunakan javascript
- Game Engine
- Mobile App
- Web App

Panduan Laporan dan Dokumentasi

Sebagai programmer yang baik, Anda dilatih tidak hanya untuk membuat kode, tapi juga membuat dokumentasi dan tes.

Anda wajib membuat sebuah README (disarankan dalam markdown) berisikan setidak-tidaknya stuktur kode, cara *compile*, cara *run*, dan *screenshot* aplikasi.

Berikanlah dokumentasi dari kode Anda, minimal dalam bentuk komentar di kode, misalnya penjelasan dari kegunaan kelas, tujuan tiap method kelas, dll. Dengan dokumentasi yang baik, programmer lain dapat membaca dokumentasi Anda dan paham cara melakukan modifikasi terhadap kode Anda, melakukan *debugging*, dan menggunakan kode Anda.

Anda juga perlu membuat tes untuk kode Anda. Lebih baik lagi, bila Anda dapat membagi tugas dan setiap kode dites oleh orang yang berbeda. Tes yang baik akan mencoba semua kasus yang mungkin terjadi. Anda dapat membaca juga mengenai Unit Testing dan Integration Testing. Beberapa bahasa juga menyediakan framework untuk melakukan test (misal googletest, junit, jest, dll). Penggunaan framework testing seperti ini opsional (tidak wajib), namun Anda wajib membuat test untuk kode Anda.