**Memahami Robust dan Non Robust dari Source Code**

diajukan untuk memenuhi salah satu tugas mata kuliah APPL 1



Disusun oleh :

Hafiyyan Putra (101511007)

Laras Ervintyana D. K. S (101511016)

Yoga Adhi Pradana (101511032)

Kelas 2 - A

**D-3 Jurusan Teknik Komputer dan Informatika**

**Politeknik Negeri Bandung**

**2012**

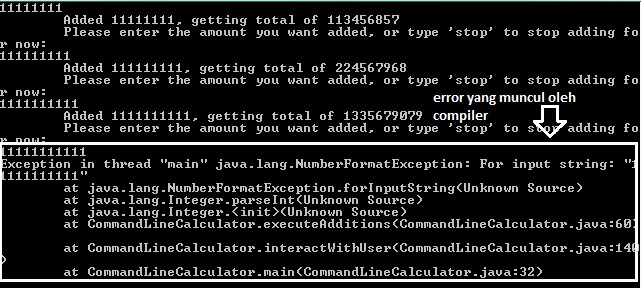
1. **Pemahaman Cara Kerja Source Code *CommandLineCalculator***
2. ***Non Robust Source Code***

Menurut kelompok kami, program ini lebih berarah pada program penjumlahan dan pengurangan sederhana, yang menurut kelompok kami program ini merupakan aplikasi bank, dimana tersedia pilihan berapa *account* yang akan kita buat, dan setiap *account* memiliki *balance*-nya sendiri. Tipe data yang digunakan adalah untuk *account* dan *balance* adalah *integer*, sehingga ada saatnya apabila *balance* yang kita punyai melebihi batas *integer* akan menjadi minus, dan juga kita tidak bisa memasukkan input angka lebih dari 10 digit.

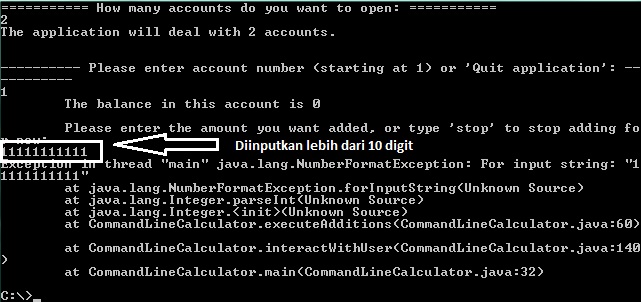
1. ***Robust Source Code***

Menurut kelompok kami, program ini sama seperti program pertama, lebih kepada program aplikasi bank. Pengerjaan perhitungan sama seperti *non robust source code*, hanya setelah dicoba dengan beberapa kesalahan *input user* telah dapat diatasi oleh program sendiri.

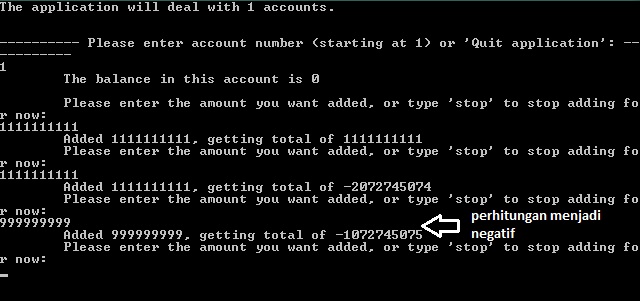
1. **Penjelasan Mengapa *Source Code CommandLineCalculator* Dikatakan *Non Robust* dan *Robust***
2. ***Non Robust Source Code***
3. Setelah dicoba untuk menginputkan, rupanya program ini tidak mempunyai *exception* untuk menangani kesalahan dari user. Pada Gambar 1.1 user menginputkan digit lebih dari 10, sehingga muncul *error* oleh *compiler*. Pada Gambar 1.2, user menginputkan nomor *account* lebih dari yang telah dibuat. Pada Gambar 1.3, kesalahan terjadi saat perhitungan, ini disebabkan tipe data yang dipakai adalah *intger* sehingga perhitungan yang melebihi *range*-nya (-2147483648 sampai 2147483647) akan dijadikan negatif, seharusnya pada hal ini, pembuat memberikan validasi terhadap data yang diinput sehingga setidaknya dapat menghindari hasil negatif yang seharusnya positif.

****

**Gambar 1.1**

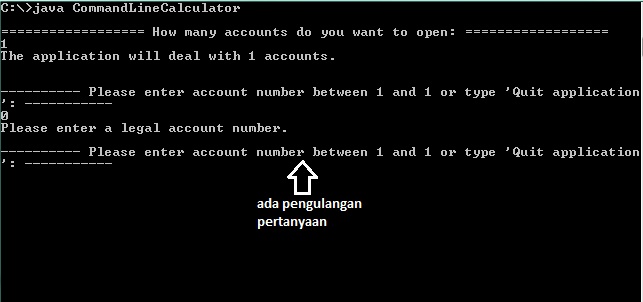
****

**Gambar 1.2**

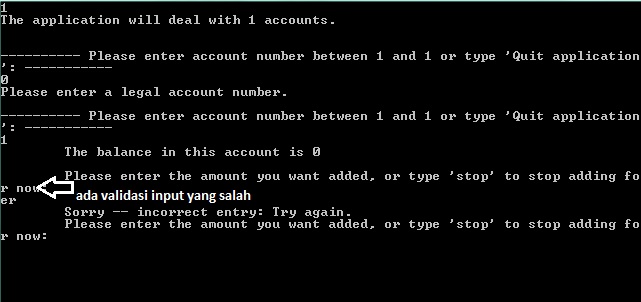
****

**Gambar 1.3**

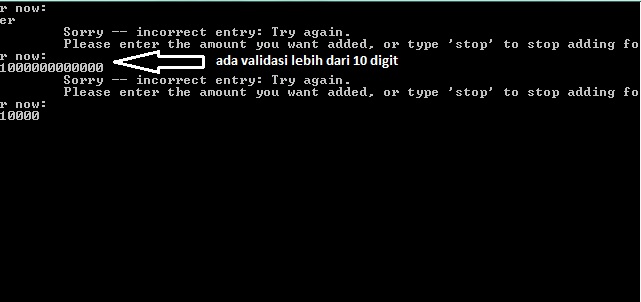
1. Kesimpulannya, mengapa *source code* ini dianggap *non robust*, karena tidak adanya kemampuan dari program untuk mengatasi kesalahan *input* dari *user* sehingga mengandalkan *compiler*, dan tidak adanya validasi dari *input user*, hal ini berguna karena bisa saja *user* menginputkan hasil yang melebihi dari batas tipe data yang dipakai.
2. ***Robust Source Code***
3. Setelah dicoba untuk menginputkan, program ini telah menyertakan *exception* untuk menangani kesalah *input user*. Pada Gambar 1.4, diinputkan kesalahan, program telah dapat mengatasi dengan adanya pernyataan “*Please enter a legal account number.*”, dan diulanginya pernyataan untuk menginputkan *account number* yang valid. Pada gambar 1.5, dicobakan untuk input selain *integer* dan *stop*, program tetap dapat menanganinya. Pada gambar 1.6, dicoba untuk input melebihi dari batas *integer*, perlakuannya tetap sama, program tetap dapat menanganinya, demikian pula bila diinput angka negatif. Dapat diulanginya input tanpa harus keluar dari program.



**Gambar 1.4**

****

**Gambar 1.5**

****

**Gambar 1.6**

1. Kesimpulannya, mengapa *source code* dianggap *robust*, karena adanya kemampuan dari program untuk mengatasi kesalahan *input user*, dan dapat menguntungkan *user*, karena *user* tidak harus mengulangi program. Dan pesan error yang disampaikan sudah jelas dan mudah dipahami.