**Singleton**

**Anggota :**

* 2440068902 Leonardo Wijaya
* 2440047980 Benyamin Kristiawan Aliwinoto
* 2440021161 Matthew Tirtasaputra
* 2440062810 Bunga Sari Ariansyah

**Definisi dari Singleton :**

Singleton adalah salah satu dari klasifikasi Creational Design Pattern. Pattern ini memastikan bahwa kamu hanya perlu satu instance , sambil menyediakan akses secara global ke instance. Hal ini berguna ketika kamu perlu sebuah objek yang dapat mengkoordinasikan tindakan diseluruh sistem. Dalam pemrograman, sebuah objek bisa dikatakan Singleton, apabila hanya memiliki sebuah instance yang digunakan diseluruh kode logikamu.

**Kenapa Singleton?** Dalam kenyataannya Singleton Pattern dapat menyelesaikan dua masalah sekaligus :

1. Untuk Memastikan Bahwa Sebuah Class Hanya Memiliki Satu Instance untuk apa orang ingin mengatur banyak instance dalam class yang ia buat? Alasan paling umum untuk ini adalah ia ingin mengendalikan akses sebuah shared resource, contohnya sebuah Database atau File.

Contoh Kesalahan: Bayangkan kamu membuat sebuah objek, tapi beberapa saat kemudian kamu memutuskan untuk membuat yang baru. Bukannya menerima objek baru, malah menerima objek yang telah kamu buat.

1. Untuk Menyediakan Sebuah Akses Global Ke Instance Tersebut ingat apa itu [Global Variabel](https://anbidev.com/tutorials/python/python-variable/) yang digunakan untuk menyimpan dan mengakses sebuah objek dimanapun itu? Meskipun ini sangat praktis, hal ini juga sangat beresiko. Karena variabel ini sangat rentan di overwrite oleh kode lainnya dan membuat app crash.

* Kapan Kita Menggunakan Singleton?
* Gunakan Singleton ketika class pada programmu seharusnya memiliki satu instance untuk semua modul. Contoh, seperti shared database atau file.
* Gunakan Singleton ketika kamu ingin mengontrol atas global variabel pada program yang kamu buat.
* Kelebihan Singleton
* Kamu bisa memastikan bahwa hanya ada satu instance yang dibuat.
* Kamu dapat mengontrol instance yang memiliki akses secara global.
* Objek Singleton hanya diinisialisasi ketika ia direquest untuk pertama kalinya.
* Kekurangan Singleton
* Singleton dapat menutupi desain program yang buruk oleh penciptanya, misalnya, ketika setiap komponen program tahu terlalu banyak informasi tentang satu sama lain.
* Singleton membutuhkan perlakuan khusus saat berada di environment multi-threaded, agar tidak membuat objek Singleton berkali - kali.

Dari project yang kami buat, kami akan menampilkan dan menjelaskan bagian mana saja yang merupakan Singleton pada project kami.

Implementasi Singleton di project kami dapat di lihat dibagian Repository.java terdapat

**private** **static** Repository *instance*; //Global variable

**private** Connect connect; //Global variable

Kemudian kami membuat sebuah function getInstance untuk menginisialisasi jika instance == null maka instance = new Repository ( **Connect** 🡪Database class : Connect =get.Connection()🡪sebuah method yang digunakan untuk get instance dari class connect ), jika tidak ==null maka return instance; Penjelasan diatas dapat dilihat pada code sinppet dibawah ini :

**private** Repository() {

connect=Connect.*getConnection*();

}

**public** **static** Repository getInstance() {

**if** (*instance*==**null**) *instance* = **new** Repository();

**return** *instance*;

}

Sehingga setelah kita mengimplementasikan design pattern : Singleton maka tidak perlu membuat object baru untuk mengaskes shared file (dalam hal ini database miniose) Mari kita lihat before pengimplementasian dan after pengimplementasian design pattern : single ton

**BEFORE**

Sebuah gambar berisi teks

Description automatically generated

**AFTER**

Sebuah gambar berisi teks

Description automatically generated

1. Mari kita mulai dengan table berikut, bisa kita lihat bahwa dari tiap table memiliki satu class/instancenya masing-masing yang sudah sesuai dengan ketentuan Singleton. Dimana satu class sudah mewakili semuanya, misalnya pada table transaction, pada class ini jika kita ingin melakukan transaction checkout barang kita memerlukan seperti TransactionID, CustomerID, ProductID, ShippingID, dimana semua data tersebut sudah ada dalam classnya masing-masing dan kita hanya perlu memanggilnya saja.

Graphical user interface, text, application, Word

Description automatically generated

1. Bisa kita lihat pada bagian code dari class Transaction, seperti yang sudah saya jelaskan tadi pada bagian ini hanya perlu memanggil masing-masing dari tiap class tersebut jika sudah selesai berbelanja dan ingin melanjutkan ke tahap transaksi. Pada code dibawah ini kita tinggal menambahkan setter getternya saja. Method setter dan getter adalah dua method yang tugasnya untuk**mengambil dan mengisi data ke dalam objek**. Dalam OOP kita sering mendengar istilah encapsulation (pembungkusan), di mana data dibungkus dengan modifier private agar tidak bisa diakses secara langsung dari luar class.

Text

Description automatically generated

1. Selanjutnya pada bagian class Shipping, dimana kita membuat String untuk ID dan nama untuk user, lalu kita tambahkan lagi setter getternya.

Text

Description automatically generated

1. Lanjut kita ke bagian class Customer, dimana pada bagian ini berfungsi untuk membuat daftar profil dari customer yang ingin berbelanja, seperti customerID, customerName, customerGender, dan lainnya, lalu kita buatkan lagi setter getter pada class Customer.

Text

Description automatically generated

1. Selanjutnya ada bagian class Product, dimana pada bagian ini fungsinya yaitu untuk membuat string untuk productid, product name, product type, dan product type, lalu buat setter getternya.

Text

Description automatically generated

1. Selanjutkan pada code kita juga tidak lupa untuk membuat class Main, gunanya untuk membuat tampilan menu dan menjalankan program yang akan kita buat. Pada bagian class Main ini kami membuat perintah untuk tampilan Menunya, seperti :

* Buy Product
* Change Product
* View All Transaction
* Delete Transaction
* Exit

Text

Description automatically generated