1. Pendahuluan

1.1 Latar belakang

Pengkodean merupakan pekerjaan yang tidak mungkin terpisahkan dengan bidang informatika. Pengkodean biasanya dilakukan oleh seseorang apabila akan membuat suatu perangkat lunak perangkat lunak komputer. Banyaknya baris kode yang ditulis sering kali membuat seseorang bingung dengan kode yang telah dibuatnya. Kebingungan muncul ketika terdapat banyaknya kode yang terhubung antara satu dan lainnya.

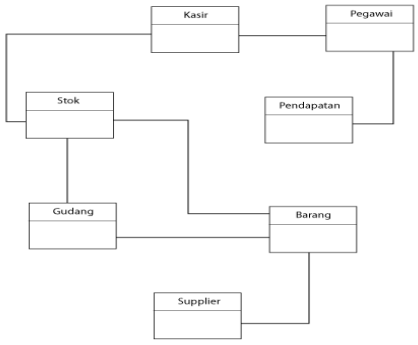
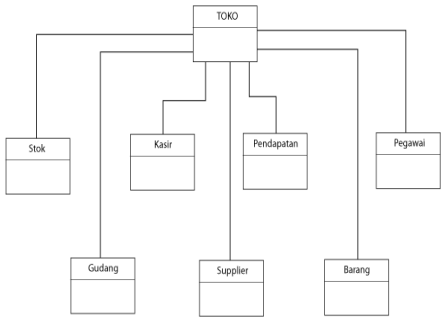
Keterhubungan antar kode menjadikan kode-kode tersebut menjadi sangat rumit dan sulit untuk dipahami. Kerumitan biasanya terjadi saat sistem akan menjalankan suatu fungsionalitas. Pemanggilan fungsionalitas dilakukan melalui kelas GUI, kelas-kelas GUI akan memanggil *method* yang berada di kelas logik. Pemanggilan *method* dilakukan berulang kali, sehingga terjadi banyak hubungan antara kelas-kelas GUI dan kelas-kelas logik.

Masalah kerumitan yang terjadi pada suatu perangkat lunak berorientasi dapat diselesaikan dengan menggunakan *design pattern. Design Pattern* adalah sebuah solusi pemecahan masalah pada pemrograman berorientasi objek. Pemecahan masalah dilakukan dengan melakukan pembelajaran terhadap masalah serupa yang telah terjadi sebelumnya. Mencari solusi pada suatu masalah dan solusi tersebut akan digunakan untuk menangani masalah serupa yang terjadi di waktu mendatang.

Tugas akhir ini akan membahas suatu *design pattern* yang merupakan bagian struktural, yaitu bagian yang membahas hubungan atau relasi antar kelas atau objek. Fokus dari *structural pattern* yang akan di bahas pada tugas akhir ini adalah *facade pattern.* Dengan menggunakan *facade pattern* maka hubungan antar kelas akan dapat disederhanakan tanpa mengganggu fungsionalitas dari perangkat lunak yang dibangun.

Suatu perangkat lunak berbasis objek umumnya terdiri dari banyak kelas yang memiliki fungsionalitas yang berbeda. Antara kelas satu dan lainnya dapat memiliki hubungan yang rumit. Selain itu, dimungkinkan suatu proses dapat terlaksana apabila telah melakukan banyak proses yang saling terhubung.

Dengan menggunakan *facade pattern* maka pengaksesan dari kelas-kelas yang rumit tersebut dapat dilakukan dengan menggunakan suatu kelas tambahan. Dapat dilihat pada Gambar 1-1, kelas *facade* dapat menyembunyikan kerumitan yang ada pada suatu perangkat lunak. Sehingga pengaksesan akan dilakukan hanya melalui kelas *facade.* Diharapkan pengkodean menjadi sederhana dan mengurangi kerumitan program, serta dapat memepermudah dalam melakukan pemeliharaan.

Keterhubungan Kelas Tanpa *Facade Pattern*	**Keterhubungan Kelas Dengan *Facade Pattern***
	

Gambar 1-1: Perbandingan keterhubungan kelas tanpa facade.

1.2 Perumusan masalah

Beberapa persoalan yang dapat dirumuskan untuk menyelesaikan tugas akhir ini sebagai berikut:

1. bagaimana cara menerapkan *facade pattern* untuk menyelesaikan kerumitan pengkodean saat kelas GUI melakukan pengaksesan kelas pada subsistem untuk menjalankan suatu fungsionalitas.
2. bagaimana pengaruh *facade pattern* pada sebuah aplikasi apabila diukur dengan menggunakan *Object Oriented Metrics.*

Permasalahan yang akan dibahas pada tugas akhir ini akan dibatasi untuk mengindari melebarnya cakupan permasalahan. Batasan masalah pada tugas akhir ini meliputi:

1. Implementasi *facade pattern* dilakukan terhadap aplikasi *desktop*.
2. Penggunaan basis data diperlukan dalam pengimplementasian aplikasi yang akan dibangun, tetapi basis data bukanlah persoalan utama dalam tugas akhir ini.
3. Komponen *object oriented metrics* yang akan dianalisis adalah:
   a. *Coupling*
      Coupling between object (CBO), adalah jumlah kelas kelas yang terhubung atau berpasangan. Kondisis *couple* terjadi apabila *method* pada suatu kelas dibentuk dari *method* kelas lainnya. CBO semakin besar maka tingkat penggunaan kembali kelas tersebut akan semakin kecil.
      Coupling Factor (COF), adalah proporsi perbandingan jumlah *coupling* sebenarnya dan kemungkinan *coupling* yang dapat muncul pada sistem.
      Response For a Class adalah *metrics* yang digunakan untuk menghitung banyaknya jumlah *method* yang digunakan oleh suatu kelas.

b. *Complexity*
   Cyclomatic Complexity merupakan salah satu *metrics* yang digunakan untuk menghitung kompleksitas suatu perangkat lunak
c. Baris program
   Line of Code (LOC), adalah banyaknya baris program yang menyusun suatu aplikasi.

1.3 Tujuan

Tujuan dari pengerjaan tugas akhir ini yaitu.

1. Mengimplementasikan *facade pattern* pada sebuah aplikasi.
2. Membuktikan adanya peningkatan *reusability* dalam aplikasi yang telah diterapkan *facade pattern* dengan menggunakan *Object Oriented Metrics.*
3. Membuktikan adanya penurunan kompleksitas dalam aplikasi yang telah diterapkan *facade pattern* dengan menggunakan *Object Oriented Metrics.*

1.4 Metodologi penyelesaian masalah

Beberapa metodologi yang digunakan dalam pengerjaan tugas akhir ini yaitu.

1. Studi literatur, pencarian referensi dan materi yang pendukung dalam pengerjaan tugas akhir ini.
2. Analisis dan perancangan, malakukan analisis permasalahan, kebutuhan, rancangan basis data, dan rancangan sistem, baik yang telah menerapkan *facade pattern* maupun yang belum menerapkan.
3. Implementasi, mengimplementasikan rancangan ke dalam baris kode program, baik program yang telah menerapkan *facade pattern* maupun yang belum menerapkan.
4. Pengujian dan analisis hasil, mengadakan pengujian secara menyeluruh dan menganalisis hasil dari pengimplementasian system dengan menggunakan *Object Oriented Metrics*.
5. Pembuatan kesimpulan dan laporan, penyusunan laporan hasil penelitian dan menyimpulkan hasil dari analisis yang telah dilakuakan sebelumnya.