Abstrak

Design pattern muncul akibat adanya permasalahan yang sama yang sering muncul pada desain pembuatan perangkat lunak. Pada perkembangannya sudah banyak *design pattern* yang sudah ditemukan oleh para programmer. Saat ini, *design pattern* dikelompokkan ke dalam tiga tujuan berbeda, yaitu *creational, structural,* dan *behavioral.*

Composite pattern merupakan salah satu *design pattern* yang tergolong ke dalam *structural pattern.* Salah satu permasalahan yang kerap muncul dalam pembuatan perangkat lunak tanpa design pattern adalah dimana sebuah simple interface dapat mengakses kumpulan object dalam sebuah struktur composite dan memiliki kemampuan untuk membedakan antara object primitive dan composite object. Permasalahan seperti ini dapat diselesaikan dengan *composite pattern.*

Pada tugas akhir kali ini, dibuat sebuah perangkat lunak yang mengimplementasikan *composite pattern* untuk menyelesaikan sebuah kasus yang memiliki beberapa kondisi. Untuk mengevaluasi *composite pattern*, dilakukan pengujian serta perhitungan dengan *object-oriented metrics*. Sedangkan untuk mengetahui kelebihan dan kekurangan yang ada pada *composite pattern*, akan dibandingkan hasil perhitungan *object-oriented metrics* perangkat lunak yang menerapkan *composite pattern* dengan perangkat lunak yang tidak menerapkan *composite pattern* untuk sebuah kasus yang serupa.

Berdasarkan hasil analisis dan pengujian, jika dilihat dari perhitungan *object-oriented metrics,* perangkat lunak dengan *composite pattern* memiliki kompleksitas yang lebih tinggi dan membutuhkan usaha yang lebih besar ketika dilakukan *maintenance* dibandingkan perangkat lunak tanpa *composite pattern*. Namun, jika dilihat dari strukturnya, perangkat lunak dengan *composite pattern* memiliki struktur yang dapat dengan mudah untuk diterapkan kembali di platform yang berbeda (*reusability*).

Kata kunci: *design pattern, composite pattern, reusability dan object-oriented metrics*.