<Online Ticket System>

Object Design

<1.0>

<23.12.2019>

<Özay Ezerceli, Mert Mısırlıoğlu, Gizem Gümüşçekiççi, Dilara Ünbay>

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc436772639)

[1.1. Object Design Trade-offs 1](#_Toc436772640)

[1.2. Interface Documentation Guidelines 1](#_Toc436772641)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc436772642)

[1.4. References 1](#_Toc436772643)

[2. Packages 1](#_Toc436772644)

[3. Class Interfaces 1](#_Toc436772645)

OBJECT DESIGN DOCUMENT

Object Design Document (ODD) describes object design trade-offs made by developers, guidelines they followed for subsystem interfaces, the decomposition of subsystems into packages and classes, and the class interfaces. The ODD is **used** to exchange interface information among teams and **as a reference during testing**. The audience for the ODD includes system architects (i.e., the developers who participate in the system design), developers who implement each subsystem, and testers.

Among three approaches to generate ODD, we follow “**ODD embedded into source code**” approach in SE301, since the other methods create many redundancies, inconsistencies.

The initial version of the ODD can be written soon after the subsystem decomposition is stable. Both packages and class interfaces can be generated from source code (comments!) by using a tool, which is named Javadoc. Keeping material for the ODD with the source code enables the developers to maintain consistency more easily and rapidly.

# Introduction (Gizem)

Describes the general trade-offs made by developers (e.g., buy vs. build, memory space vs. response time), guidelines and conventions (e.g., naming conventions, boundary cases, exception handling mechanisms), and an overview of the document. Interface documentation guidelines and coding conventions are the single most important factor that can improve communication between developers during object design. These include a list of rules that developers should use when designing and naming interfaces.

## Object Design Trade-offs

## Interface Documentation Guidelines

## Definitions, Acronyms, and Abbreviations

## References

References to existing systems, etc.

# Packages (Mert)

Describes the decomposition of subsystems into packages and the **file organization of the code.** This includes an overview of each package, its dependencies with other packages, and its expected usage.

# Class Interfaces (Dilara-Özay)

**Describes the classes and their public interfaces.** This includes an overview of each class, its dependencies with other classes and packages, its public attributes, operations, and the exceptions they can raise.