

CS 319

Object-Oriented Software Engineering  
Fall 2014

**Conquerors**

Design Report

Group H  
Çağla Arıkan

Metin Enes Uçan

Can Akgün

İsmail Can Dereli

Table of Contents

[1. INTRODUCTION 3](#_Toc403742638)

[1.1. Purpose of the System 3](#_Toc403742639)

[1.2. Design Goals 3](#_Toc403742640)

[1.2.1. Goals 3](#_Toc403742641)

[1.2.2. Trade-Offs 4](#_Toc403742642)

[1.3. Definitions, acronyms, and abbreviations 5](#_Toc403742643)

[1.3.1. Definitions 5](#_Toc403742644)

[1.3.2. Abbreviations 5](#_Toc403742645)

[1.4. References 5](#_Toc403742646)

[2. SOFTWARE ARCHITECTURE 7](#_Toc403742647)

[2.1. Subsystem Decomposition 7](#_Toc403742648)

[2.2. Hardware / Software Mapping 8](#_Toc403742649)

[2.3. Persistent Data Management 8](#_Toc403742650)

[2.4. Access Control and Security 8](#_Toc403742651)

[2.5. Boundary Conditions 8](#_Toc403742652)

[2.5.1. Initialization 8](#_Toc403742653)

[2.5.2. Termination 9](#_Toc403742654)

[2.5.3. Error 9](#_Toc403742655)

[2.5.4. Game Updating 9](#_Toc403742656)

[3. SUBSYSTEM SERVICES 9](#_Toc403742657)

# INTRODUCTION

In this section of the report we will talk about the purpose of the system of our game, our goals of design and some definitions, acronyms and abbreviations we will use throughout he report.

## Purpose of the System

The game “Conquerors” is an arcade game which is inspired from the Atari game “*Volfied*”. The aim of this game is to conquer the castle by drawing lines before time runs out. During the game player will face with different enemies that will obstruct our hero to reach the goal. Unlike the original game, our game has historical and military elements in it. We aim to make the game more enjoyable than its ancestor. That’s why we added different bonuses to make the player’s job easier or harder. Our game has 5 different levels and as levels passed, completing the levels will be harder and the player will face with more enemies. With its user-friendly interface, the “Conquerors” will be easy to learn and play but hard to win.

## Design Goals

In order to make our game a high-quality one, we should set some design goals. We set those goals based on the non-functional requirements which are determined in the analysis phase.

### Goals

* ***Reliability:*** Making a reliable game is very important for us. We aim to create a game with no bugs. Moreover game will not crush unexpectedly. When an unknown input is given, it won’t stop working or close itself.
* ***Modifiability [1]:*** It is vital to create a system which can be modified easily. We may want to add new features to our game or remove some other. In order to reach that goal we will minimize the coupling of our subsystems. Furthermore using encapsulation and polymorphism in our design will make our system easier to modify.
* ***Adaptability:*** It is necessary for us to create a game which can be used in other platforms. We choose Java for writing the code of the game. Java provides us cross-platform portability. In every platform that JRE is installed, our game can be executed. That feature of Java made us choose it.
* ***Ease of Use:*** We aim to create a game which has a wide range of audience so *Conquerors* should be very easy to use. The interface of the game will be very friendly. Menu part will be not complex but informative. Moreover, game is only controlled by the arrow keys on the keyboard. Thus there will be no confusion for the player and it will be easier to play. However, those features will not make the game itself easy since we aim to create hard and fun game.
* ***Ease of Learning:*** The game will be very easy to learn. There will be a “View Help” part for the player. In there, the player can see what will create the movement of the Hero, and shows all the objects. The game logic will be very easy too.
* ***Response Time:*** This is a very important aspect for our system. Since we are planning to create an arcade game, the player would like to see the object move when s/he asks for it. If it does not happen, it will be very annoying. That problem will make people not to play our game. Thus we aim to make our game respond the actions of the player (nearly) immediately.

### Trade-Offs

* ***Ease of Use and Learning vs. Functionality:*** As we mentioned before, we want our game to have a wide range of audience. That’s why making the game easy to use and understand is a high priority for us. This fact means that we will does not put functionality of the system prior than ease of use and learning. Thus we won’t create complex functionalities. For instance the player can only use arrow keys to move the Hero. However if we wanted to make our game have more functionalities, we may had “A, W, S, D” too but this would made the use of the game harder.
* ***Performance vs. Memory:*** We believe that performance of a game is really important. We want our game to have smooth graphics and high response time. To reach our goal we need to make sacrifices of the memory. For instance we have a bonus which is Enemies Gone. When user got that bonus all of the enemies on the screen should be destroyed quickly. To make that happen fast we need to use a large memory to hold all of the enemies inside the game and destroying all together, other than removing them one by one.

## Definitions, acronyms, and abbreviations

### Definitions

* ***Coupling [2]:*** It is the degree to which software components are dependent upon each other.
* ***Cross-Platform [3]:*** It is an attribute conferred to computer software or computing methods and concepts that are implemented and inter-operate on multiple computer platforms.

### Abbreviations

* ***JRE [4]:*** Java Runtime Environment

## References

[1] <http://www.sei.cmu.edu/reports/07tr002.pdf>

[2] <http://whatis.techtarget.com/definition/coupling>

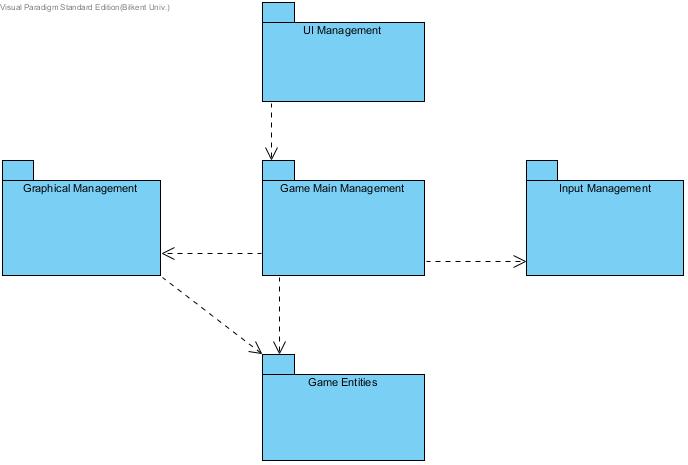
[3] <http://en.wikipedia.org/wiki/Cross-platform>

[4] <http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html>

# SOFTWARE ARCHITECTURE

## Subsystem Decomposition

In this section, it is going to be clarified that how our system will be divided into subsystems as well as how they will be reacted each other according to the main structure of the game. We have decided to use a most common use of multi-tier architecture for our system decomposition which is three-tier architecture. This architecture is a well-established software design pattern and it provides a flexible and reusable model since presentation, application processing, and data management functions are separated. The below diagram illustrates the three-tier implementation of our system from an overview perspective.



**Image 1:** High-Level Demonstration of the Subsystem Decomposition

## Hardware / Software Mapping

Conquerors game will be implemented in Java programming language; therefore we will use latest JDK (8u25). As hardware configuration, user needs a basic keyboard to give input to the system. Because of Java implementation, system requirements will be minimal so any kinds of computer, which can compile java and run .java file, easily run conquerors.

Storage will base on .txt structures to high score list because of that, the operating system should support .txt file formats. On the other hand, this game will not have multiplay mode so system does not need any network connection.

## Persistent Data Management

Conquerors game does not need complex database system. System will only store high score list as text files in disk. System will also also store some pictures for game objects in hard disk drive.

## Access Control and Security

There is no requiring of any kind of network connection such as database system, Conquerors game can be played by anyone. Therefore there is no limited access or restriction for, controlling. The only access to the system is through Game Manager which is the main game logic class, for the security. Controlling images and objects, managing input and the game map is delegating by game logic.

## Boundary Conditions

### Initialization

Conquerors game does not require installing since it does not have regular .exe extension; instead game will be constituted as an executable .jar file.

### Termination

In any moment of the game user can terminate the game by simply pressing the “Exit” button on the JFrame.

### Error

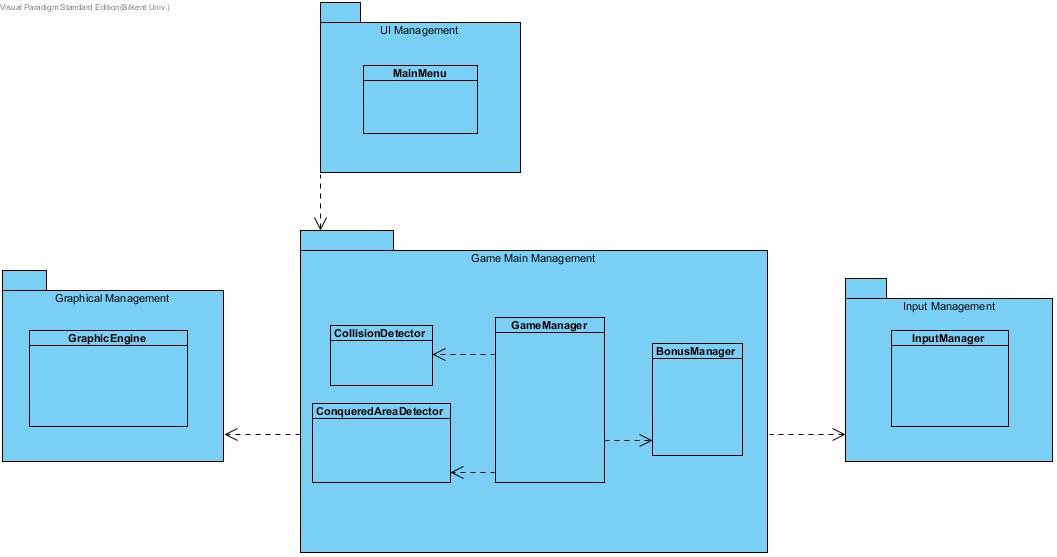
If the program does not respond because of a performance issue, player loses his current data because game data such as high scores or player’s current level are updated the end of the game.

### Game Updating

When the life of hero is finished, system automatically return the main menu and in case of that , the high scores will be uptaded if a new record is broken and also level is updated.

# SUBSYSTEM SERVICES

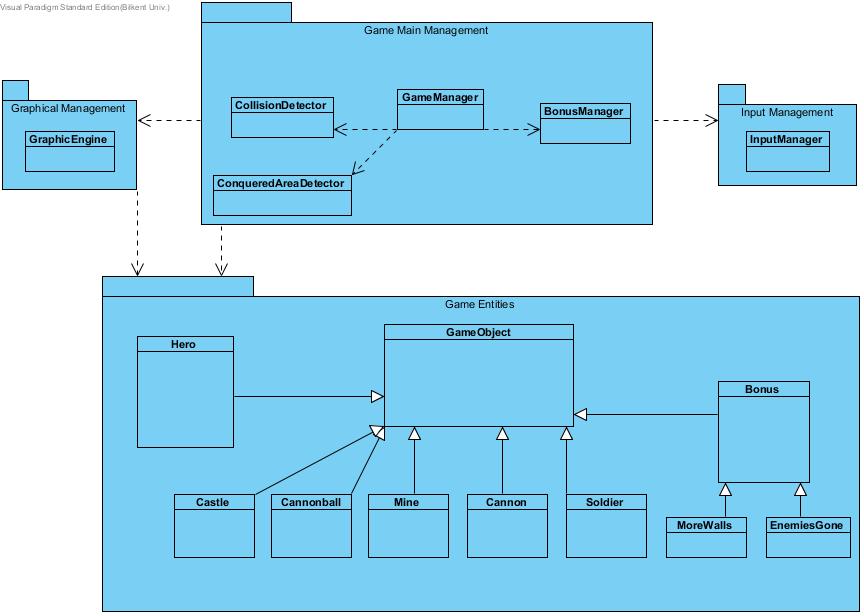
The first layer of the three-tier architecture is defined as presentation of the system so that interaction between the end users of the program and the system itself can be established. Therefore, in our game, players start the first interaction between the game layers with the help of the interface that is provided by the presentation layer. For that reason the “MainMenu” class is located in the UI Management package and with the methods of this class player choices can be transferred to the Game Main Management package which is stated in the application layer such as sound and hero type options or level selection. The diagram that states in the below demonstrates the connection between presentation and the application layers graphically.



**Image 2:** Illustration of the connection between presentation and application layers

The second layer is the application processing layer where the main logic of the program is managed. After an end user starts the interaction from presentation layer to the application layer, in other words when player passes information from the interface of UI Management package to the Game Main Management package, systems starts to run the logic of the game. Three packages of Graphical Management, Input Management and Game Main Management work together for the processing of game logic. Input Management package keeps track of the user inputs with the help of classes in it while game is running; Graphical Management package responsible from the rendering game objects and the game map, and Game Main Management package serves as a controller package that manages internal game events as well as has a connection with other two Graphical and Input Management packages in that same layer.

Last layer in three-tier architecture is the data management layer, thus in our design we have Game Entities package in that layer in order to control the manipulations on game objects. That’s why this data management layer interacts with the application layer. Game Main Management package passes information about changes in the properties of objects; Graphical Management package pursues the locations of game entities to re-draw them. The below package diagram depicts the interaction between layers of data management and the application.



**Image 3:** Illustration of the connection between data management and application layers