1 Introduction

This is the SOftware Design Description for the Tower Defense game application.

1.1 Purpose

This document is the serve as the blueprint for the construction of the Tower Defense application. This design will use UML class diagrams to provide complete detail regarding all packages, classes, instance variables, class variables, and method signatures to build the application. In addition, the UML Sequence diagrams will be used to specify object interactions post-initialization of the applications, meaning in response to user interactions or timed events

1.2 Scope

This tower defense game will be the first I will have made as a CS major at Stony Brook University.

1.3 Definitions, Acronyms, and Abbreviations

**Class Diagram –** A UML document format that describes classes graphically. Specifically, it describes their instance variables, method headers, and relationships to other classes.

**IEEE –** Institute of Electrical and Electronics Engineers, the “world’s largest professional association for the advancement of technology”.

**Framework** – In an object-oriented language, a collection of classes and interfaces that collectively provide a service for building applications or additional frameworks all with a common need.

**Java** – A high-level programming language that uses a virtual machine layer between the Java application and the hardware to provide program portability.

**Sequence Diagram** – A UML document format that specifies how object methods interact with one another.

**UML** – Unified Modeling Language, a standard set of document formats for designing software graphically.

1.5 Overview

This SDD document provides a working design for the application as described in the Tower Defense SRS. Section 2 of this document will provide the Package-Level Viewpoint, specifying the packages and frameworks to be designed. Section 3 will provide the Class-Level Viewpoint, using UML Class Diagrams to specify how the classes should be constructed. Section 4 will provide the Method-Level System Viewpoint, describing how methods will interact with one another. Section 5 provides deployment information like file structures and formats to use. Section 6 provides a Table of Contents, an Index, and References.

2 Package-Level Design Viewpoint

In building this project, we will rely on JavaFX and the Java API to provide services. Following are descriptions of the components to be built, as well as how the Java API will be used to build them.

2.1 Tower Defense Overview

Tower Defense

* Application

2.2 Java API Usage

Javafx.application

* Application
* Platform

Javafx.scene

* Stage
* Scene

Javafx.scene.layout

* Pane
* VBox
* HBox

Javafx.scene.paint

* Color

Javafx.scene.shape

* Shape
* Rectangle
* Circle
* Polyline
* Path

Javafx.scene.text

* Text

Javafx.scene.control

* Button

Javafx.animation

* PathTransition
* Interpolator

Javafx.util

* Duration
* HashSet
* Timer
* TimerTask

2.3 Java API Usage Descriptions

Javafx.application

* Application - to run JavaFX applications
* Platform - to quit out of JavaFX applications

Javafx.scene

* Stage - to show the JavaFX program
* Scene - to show the JavaFX program

Javafx.scene.layout

* Pane - to separate and show different sections in a JavaFX program
* VBox - to stack components vertically
* HBox - to stack components horizontally

Javafx.scene.paint

* Color - to color nodes in JavaFX

Javafx.scene.shape

* Shape - used in creating a HashMap of enemies
* Rectangle - used to create the playing area
* Circle - used to show the enemies
* Polyline - used to show the path the enemies take
* Path - used in creating the PathTransition for the enemies

Javafx.scene.text

* Text - to display text on the screen in this JavaFX application

Javafx.scene.control

* Button - to create a button

Javafx.animation

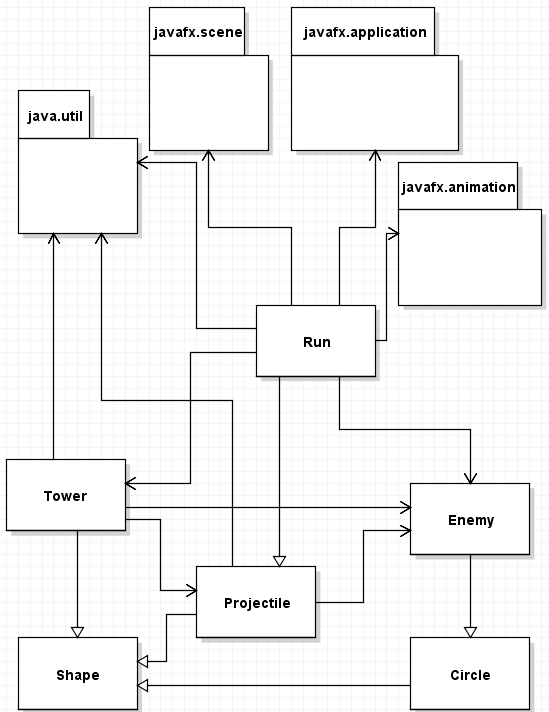
* PathTransition - to display the movement of enemies and projectiles
* Interpolator - used to set the PathTransition interpolator to LINEAR

Javafx.util

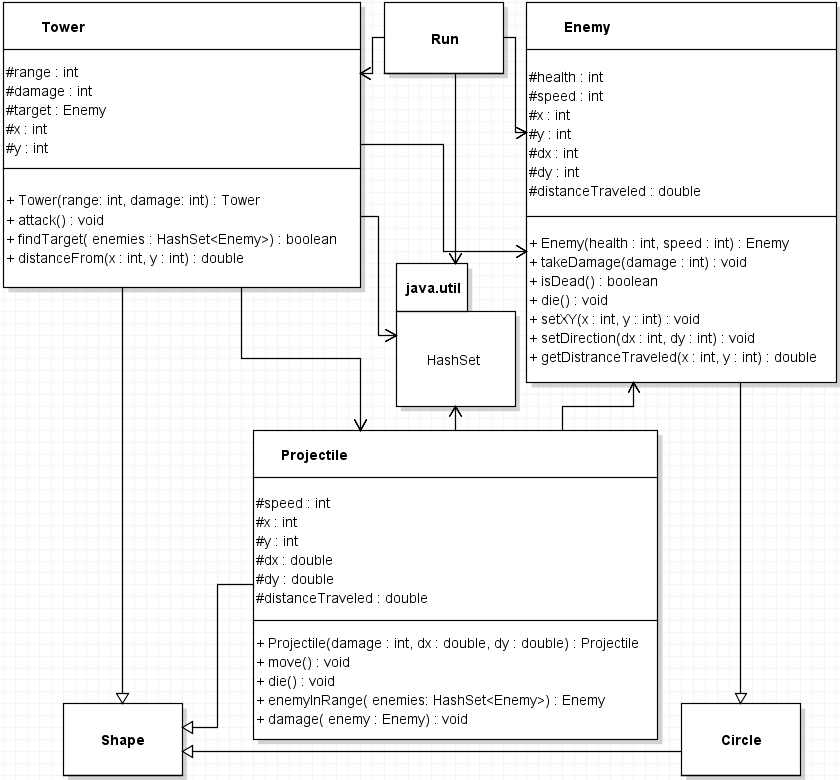
* Duration - used to create a PathTransition. Also used to set the speed of enemies
* HashSet - to create a set of enemies, a set of towers, and a set of projectiles
* Timer - used to time the spawn of the enemies and time the attacks of the towers
* TimerTask - used to time the spawn of the enemies and time the attacks of the towers

3 Class-Level Design Viewpoint

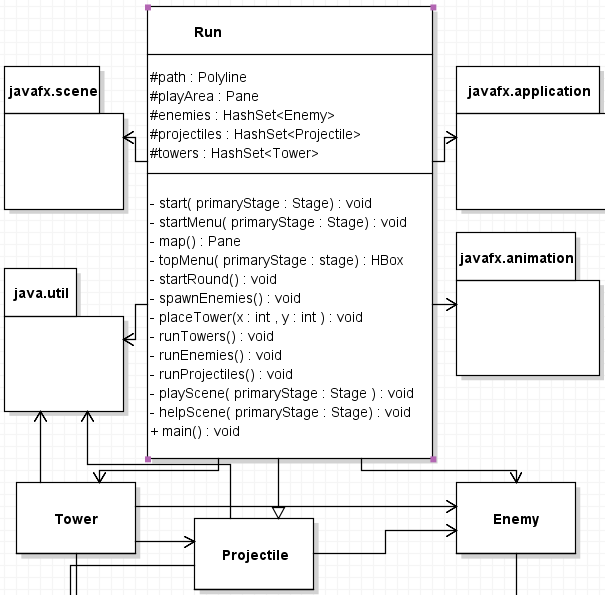
General Overview



Detailed Tower, Enemy, and Projectile UML Class Diagram

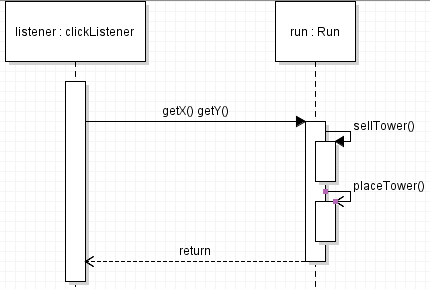


Detailed Run UML Class Diagram



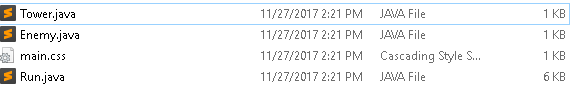
4 Method Level Design Viewpoint

This is the sequence when it comes to clicking on any part of the playing area. The pane will have a listener, which will get its coordinates to run. Run will sell or place a tower accordingly and then return.



5 File Structure and Formats

As of right now, this is the file structure. This is subject to change as images and more files are added. These classes will be provided in a zip file



6 Supporting Information

6.1 Table of contents

1. Introduction

1.1. Purpose

1.2. Scope

1.3. Definitions, acronyms, and abbreviations

1.4. References

1.5. Overview

2. Package-Level Design Viewpoint

2.1. Class overview

2.2. Java API Usage

2.3. Java API Usage Descriptions

3. Class-Level Design Viewpoint

4. Method-Level Design Viewpoint

5. File Structure and Formats

6. Supporting Information