Прилог Б:

Упутство за одржавање 3Д видео игре *Arena*

Садржај

[1. Увод 3](#_Toc33546983)

[2. Пакети 3](#_Toc33546984)

[3. Пакет concepts 3](#_Toc33546985)

[4. Пакет game 5](#_Toc33546986)

[4.1 Класа Game 6](#_Toc33546987)

[4.2 Класа Menu 6](#_Toc33546988)

[4.3 Класа SoundPlayer 6](#_Toc33546989)

[4.4 Класа HighScores 6](#_Toc33546990)

[5. Пакет objects 7](#_Toc33546991)

[5.1 Класе који наслеђују DamagingObject 7](#_Toc33546992)

[5.2 Класа Floor и њени потомци 8](#_Toc33546993)

[5.3 Класа Player 9](#_Toc33546994)

[5.4 Остале класе 10](#_Toc33546995)

[6. Пакет rooms 11](#_Toc33546996)

[7. Пакет sprites 12](#_Toc33546997)

[8. Додатак – увећане слике појединих дијаграма 12](#_Toc33546998)

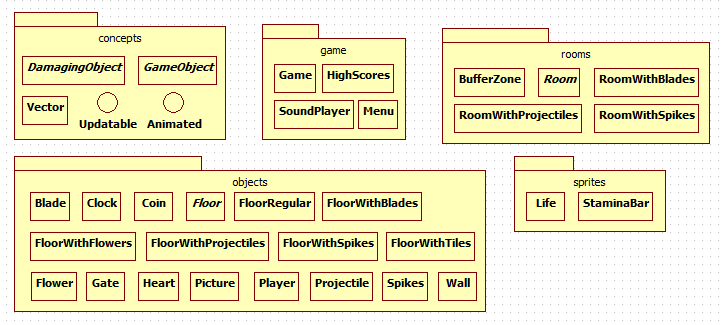
# 1. Увод

Циљ овог документа је да читаоцу олакша анализу и евентуално модификовање програмског кода игре *Arena*. У наредним одељцима биће представљене све битне класе и начин на који је код структуриран. Извесна поља и методе су изостављени јер немају значајнијег утицаја на архитектуру апликације, односно нису битни за само разумевање структуре кода, већ служе као мања спона између његових делова или прости подаци.

У пакету resources налазе се искључиво слике и аудио записи који се користе у игри, па се тај пакет неће даље разматрати.

# 2. Пакети

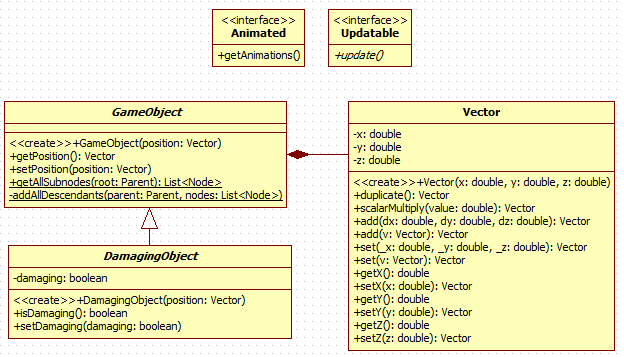
На Слици 2-1 може се видети дијаграм пакета са свим имплементираним класама у решењу. Намера овакве поделе је да буде довољно једноставна и интуитивна, а у сваком одељку везаном за појединачне пакете уследиће појашњења.



Слика 2-1

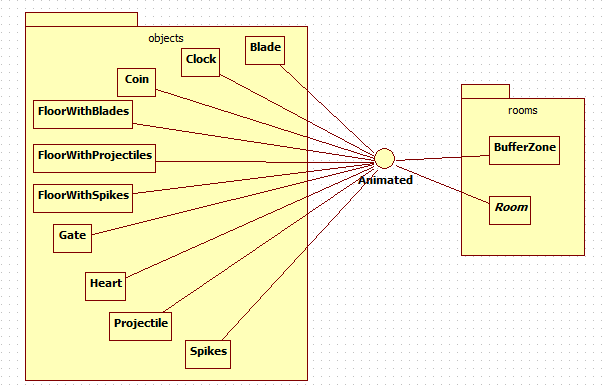
# 3. Пакет concepts

У овом пакету (Слика 3-1) налазе се класе и интерфејси који служе као заједничка основа за објекте који се исцртавају у игри – првенствено апстрактна класа GameObject која наслеђује класу Group из пакета javafx.concepts. Класа Vector има поља за чување x, y и z координата и омогућава детекцију колизија и олакшава позиционирање објеката (поготово при стварању јер се налази у конструктору класе GameObject). Апстрактна класа DamagingObject проширује GameObject додавањем поља и метода за објекте који причињавају штету играчу (Blade, Spikes и Projectile).



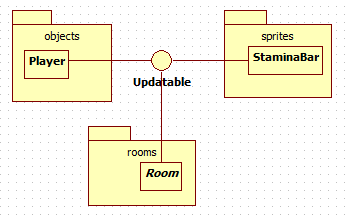
Слика 3-1

Интерфејс Animated имплементирају класе које поседују анимације (Слика 3-2) и њена метода getAnimations() служи да се дохвате анимације нпр. при паузирању игре како би се зауставиле.



Слика 3-2

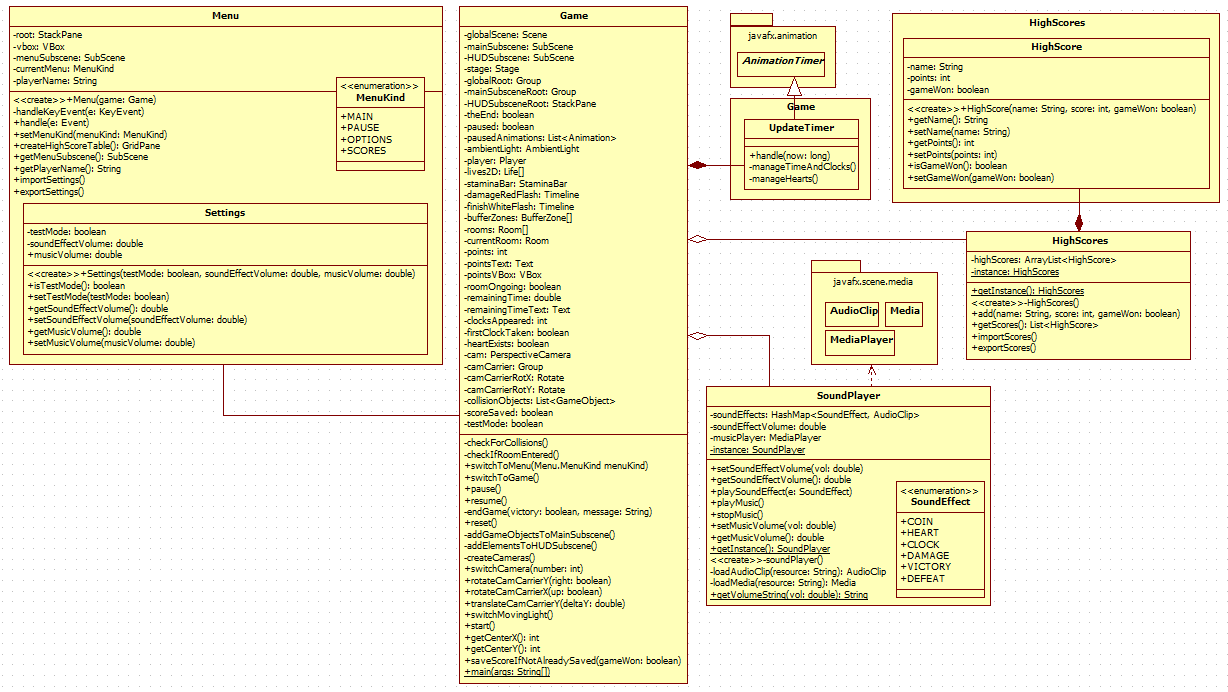
Интефејс Updatable имплементирају класе (Слика 3-2) које је потребно ажурирати у откуцајима тајмера за анимацију.



Слика 3-3

# 4. Пакет game

Овде се налазе класе које служе као функционална основа апликације. Оне повезују све њене делове и ефективно покрећу сва дешавања у игри. Дијаграм класа у пакету је приказан на Слици 4-1 (увећана слика у поглављу 8).



Слика 4-1

## 4.1 Класа Game

Ово је најсложенија класа у програму и сарджи већи број поља и метода, међу којима и main(). У њој постоји и инстанца њене унутрашње класе UpdateTimer. Она покреће update() методу класа које имплементирају раније поменути интерфејс Updatable и управља протоком времена у игри и стварањем сатова и срца. У Game постоје разне методе чији називи јасно назначују намену, а додатна објашњења вреди навести за следеће: checkForCollisions() за детекцију колизија између објеката у игри, а такође и обраду последица додира различитих објеката (нпр. играча и пројетила, играча и новчића, пројектила и пројектила и слично); checkIfRoomEntered() служи да се у тајмеру провери да ли је играч ступио на тло нове просторије како би се започело одбројавање времена и онемогућио повратак у претходну просторију; switchToMenu() и switchToGame() мењају подсцену која се исцртава и подешавају руковаоце догађајима зарад контроле у менију, односно самој игри.

## 4.2 Класа Menu

Имплементира интефејс javafx.event.EventHandler и његовим методама омогућава сналажење у менију који се исцртава у подсцени menuSubscene. Методом setMenuKind() се одабира врста менија која ће се приказати прослеђивањем параметра набројивог типа Menu.MenuKind. Унутрашња класа Menu.Settings служи за чување подешавања у фајл, односно читање из фајла. Она имплементира интерфејс java.io.Serializable. Класа Menu то чување и читање постиже методама importSettings() и exportSettings().

## 4.3 Класа SoundPlayer

Сходно имену, има намену да репродукује звучне ефекте и музику. Поседује инстанцу класе HashMap која мапира инстанцу унутрашње класе набројивог типа SoundEffect у AudioClip из пакета javafx.scene.media, која се користи за одабир звучног ефекта који се пушта методом playSoundEffect(), док се музика пушта помоћу playMusic() у којој се позивају методе поља musicPlayer типа MediaPlayer, такође из поменутог пакета. Методе loadAudioClip() и loadMedia() учитавају звукове из фајл-система. Класа SoundPlayer примењује пројектни узорак *уникат* (*Singleton*).

## 4.4 Класа HighScores

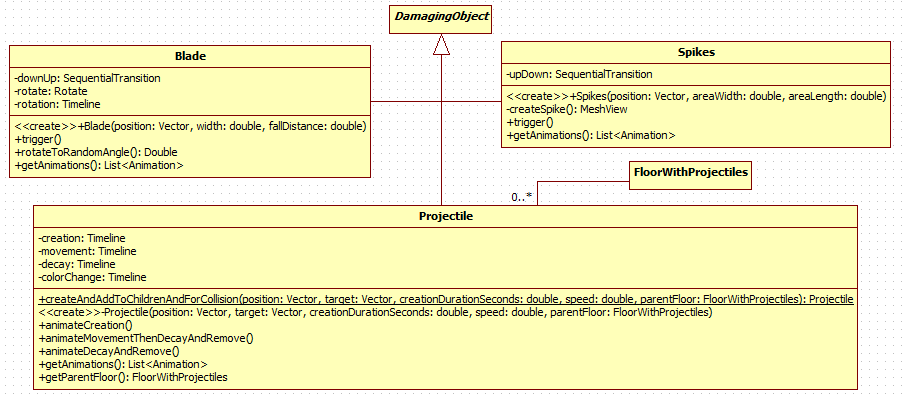
И ова класа примењује пројектни узорак *уникат*. Аналогно класи SoundPlayer, поседује унутрашњу класу HighScore за похрањивање и учитавање резултата методама importScores() и exportScores(). Резултати се у току извршавања програма налазе у листи highScores.

# 5. Пакет objects

У пакету objects махом су смештене класе које саме за себе представљају целине, а које се у просторијама у игри користе као градивни елементи, односно представљају појединачне елементе окружења. Све класе у овом пакету су потомци класе GameObject. У одељцима испод биће представљене оне најбиније, по поднасловима груписане према заједничким цртама.

## 5.1 Класе који наслеђују DamagingObject

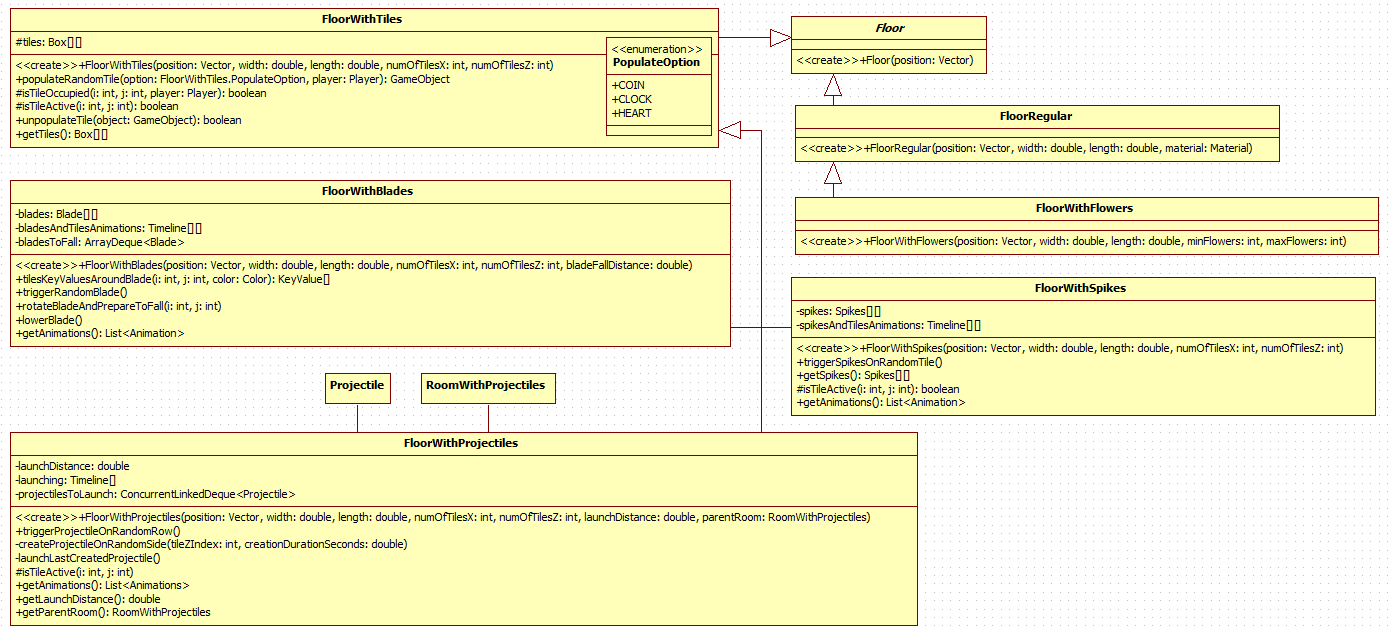
У питању су Blade, Projectile и Spikes (Слика 5-1). Све поседују поља и методе везане за анимације. Те методе су trigger() у Blade и Spikes и animateCreation(), animateMovementThenDecayAndRemove() и animateDecayAndRemove() у Projectile. Неке од тих метода се позивају из update() методе одговарајућих просторија (посредно, преко метода подкласе класе Floor за ту просторију). Статичка метода класе Projectile createAndAddToChildrenAndForCollision() служи за стварање објеката класе (конструктор је приватан) и се у њој примењују поступци за избегавање проблема са конкуретним нитима.



Слика 5-1

## 5.2 Класа Floor и њени потомци

Апстрактна ласа Floor је надкласа свих класа које представљају тло у просторијама у арени (Слика 5-2; увећана слика у поглављу 8). FloorRegular и FloorWithFlowers немају посебне функционалности.



Слика 5-2

FloorWithTiles поседује поље tiles које представља плоче на које је тло подељено и та класа је основа за тла у различитим просторијама. У конструктору су међу параметрима и два која означавају на колико ће редова и колона бити подељено. Насумично одабрана плоча се попуњава објектом за сакупљање методом populateRandomTile() чији један од параметара типа набрајања, FloorWithTiles.PopulateOption, одређује шта је у питању. Методе isTileOccupied() и isTileActive() служе (с намером да се евентуално надјачају) за проверавање да ли је плоча већ заузета или је на њој активна препрека како се не би могли створити предмети за сакупљање на истој.

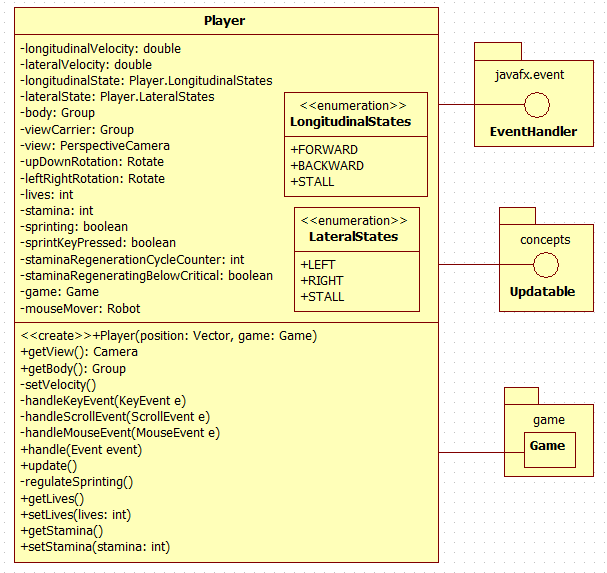
FloorWithSpikes има поља која представљају шиљке и њихове анимације и методу triggerSpikesOnRandomTile() за њихово покретање на некој од плоча. FloorWithBlades је слична по том питању (поља blades и bladesAndTilesAnimations; метода triggerRandomBlade()), али има и методе као што су: tilesKeyValuesAroundBlade() која враћа низ објеката javafx.animation.KeyValue ради одређивања плоча које ће променити боју као упозорење пред пад сечива; rotateBladeAndPrepareToFall() ротира одређено сечиво и додаје га у ред bladesToFall; lowerBlade() узима прво сечиво из реда и изазива његов пад.

FloorWithProjectiles садржи поље launchDistance које се иницијализује у конструктору и одређује раздаљину са које ће се с леве и десне стране тла лансирати пројектили. Поље launching је низ објеката типа Timeline из пакета javafx.animation и у питању су анимације промене боје редова плоча као упозорење пред лансирање пројектила, као и само покретање пројектила коришћењем ламбда функција (ту могућност има класа KeyFrame чији се објекти додају у Timeline). Метода triggerProjectileOnRandomRow() покреће play() насумично одабраног објекта из низа launching и тиме се позивају createProjectileOnRandomSide() и launchLastCreatedProjectile() путем поменуте могућности класе KeyFrame. Те две методе комуницирају са објектима типа Projectile позивањем њихових метода, а ти објекти су све време садржани у projectilesToLaunch типа ConcurrentLinkedDeque<Projectile>.

## 5.3 Класа Player

Player представља играча и садржи већи број функционалности у поређењу с другим класама из истог пакета (Слика 5-3).

Поља longitudinalVelocity, lateralVelocity, longitudinalState и lateralState се користе за одређивање брзине играча при кретању у хоризонталној равни. Поглед из првог лица се реализује кроз view типа PerspectiveCamera, а та камера је део viewCarrier како би могле да се врше ротације камере кроз повезане трансформације upDownRotation и leftRightRotation. Класа Player имплементира интерфејсе Updatable и EventHandler. Прва омогућава да се кретање играча реализује кроз надјачану update() методу коришћењем информација садржаних у пољима longitudinalVelocity и lateralVelocity, као и позивањем методе regulateSprinting() која врши додатне измене брзине кретања и регулише залихе енергије за брзо трчање у зависности од вредности поља stamina, sprinting, staminaRegenerationCycleCounter и staminaRegeneratingBelowCritical. Надјачана наслеђена метода handle() из EventHandler интерфејса, у зависности од врсте догађаја, позива handleKeyEvent, handleScrollEvent или handleMouseEvent, у којима се врше обраде догађаја (ту се најпре огледа спрега класе Player са класом Game – преко поља game). Поље mouseMover типа класе Robot из пакета java.awt одржава курсор миша центрираним у прозору игре. Класа Game позива getBody() да би дохватила дотично поље и искористила га нпр. за детекцију колизија.



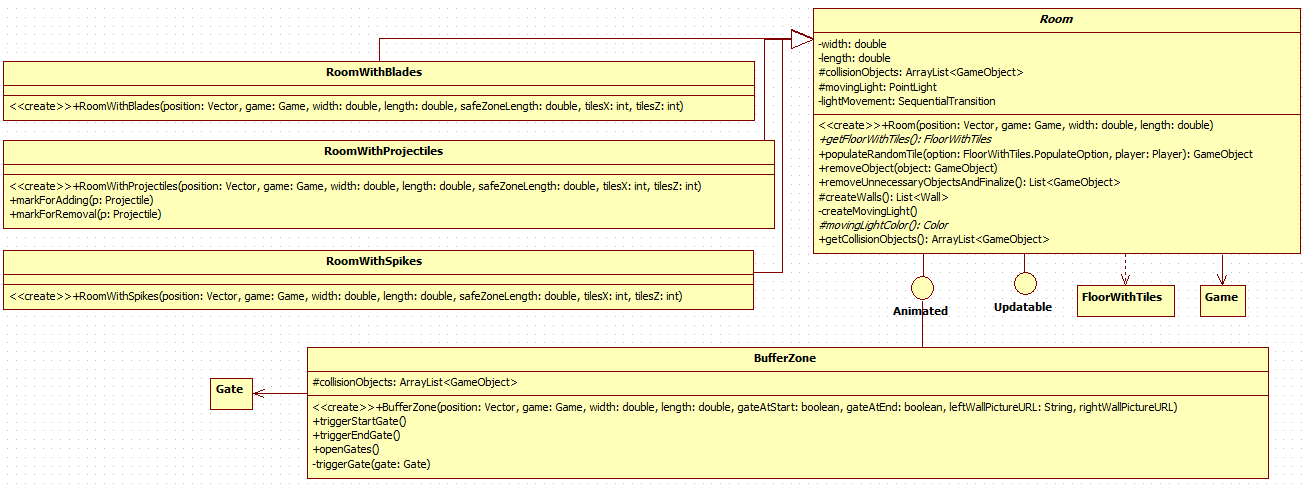
Слика 5-3

## 5.4 Остале класе

Остале класе (Clock, Coin, Flower, Gate, Heart, Picture и Wall) махом имају конструкторе и покоју помоћну функцију и евентуално пар поља за остваривање анимација. Пошто њихове унутрашње функционалности имају претежно естетске сврхе, а само као целинама им се приступа и користе се у другим деловима кода, неће бити детаљније описиване.

# 6. Пакет rooms

Иако се и овде налазе класе које су све потомци GameObject, издвојене су у сопствени пакет (Слика 6-1; увећана слика у поглављу 8) јер представљају склопове који се састоје од већег броја објеката из пакета objects.



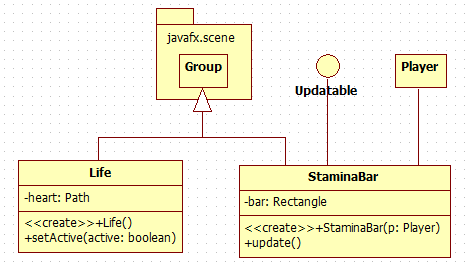
Слика 6-1

Апстрактна класа Room је основа за просторије арене. Листи collisionObjects додају се објекти које ће класа Game додати свом истоименом пољу и потом их узимати у обзир при детекцији судара. Метода populateRandomTile() енкапсулира позив истoимене методе из FloorWithTiles и врши додавање у поменуту листу. Аналогно томе функционише и removeObjects() за уклањање објеката. Метода removeUnnecessaryObjectsAndFinalize() предузима неопходне мере након што истекне време у просторији – прекида извршавање анимација, уклања непотребне објекте и слично. Подкласе надјачавају апстрактну методу movingLightColor() да би вратила жељену боју светла које кружи про просторији. Такође надјачавају и наслеђени update(), а по потреби и removeUnnecessaryObjectsAndFinalize(). Од потомака ваља напоменути да RoomWithProjectiles има методе markForAdding() и markForRemoval() које се позивају у update() да би се пројектили синхронизовано додавали или уклањали из листа.

BufferZone није потомак апстракне класе Room. Она представља ходник с капијама који се налази на улазу и излазу из просторије. Методама triggerStartGate(), triggerEndGate(), openGates() и triggerGate() се управља капијама и њиховим присуством у листи објеката за колизију.

# 7. Пакет sprites

Садржи 2D објекте за исцртавање на играчевом интерфејсу – Life за срца која означавају преостале животе, StaminaBar за индикатор енергије (Слика 7-1).

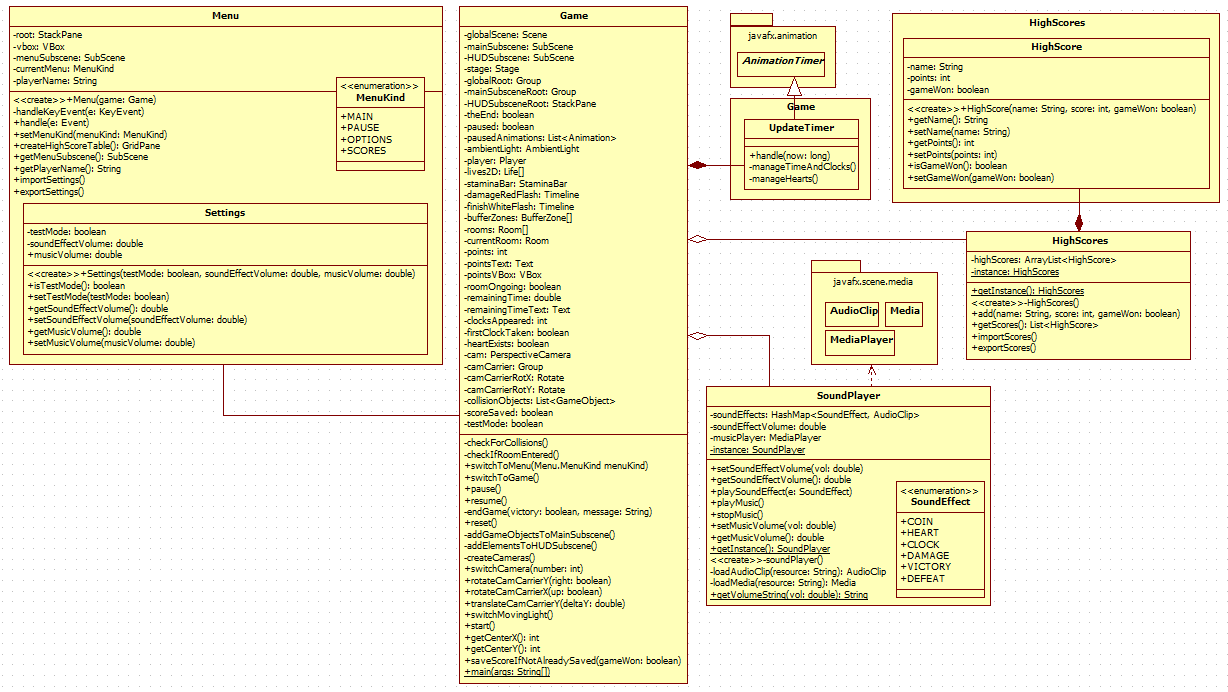


Слика 7-1

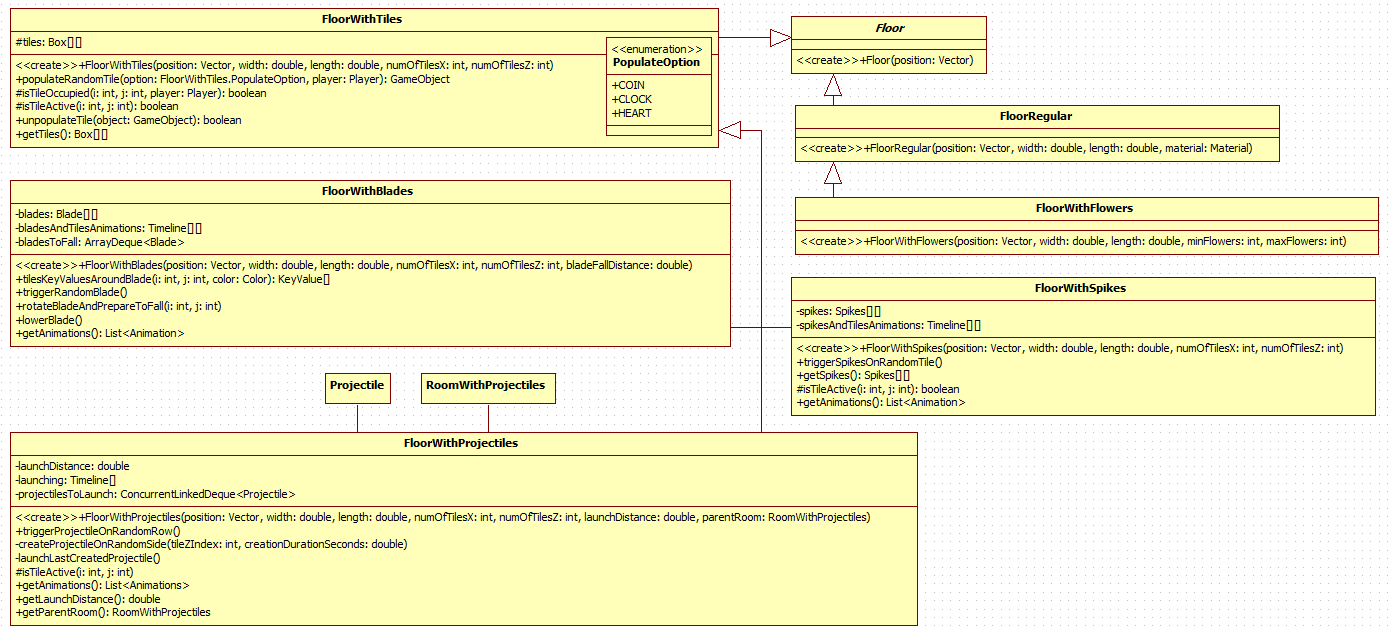
Метода setActive() у класи Life мења боју срца за које је позвано. Метода update() у StaminaBar позива getStamina() класе Player и дели је максималном количином играчеве енергије да би израчунала колико енергије индикатор треба да приказује.

# 8. Додатак – увећане слике појединих дијаграма

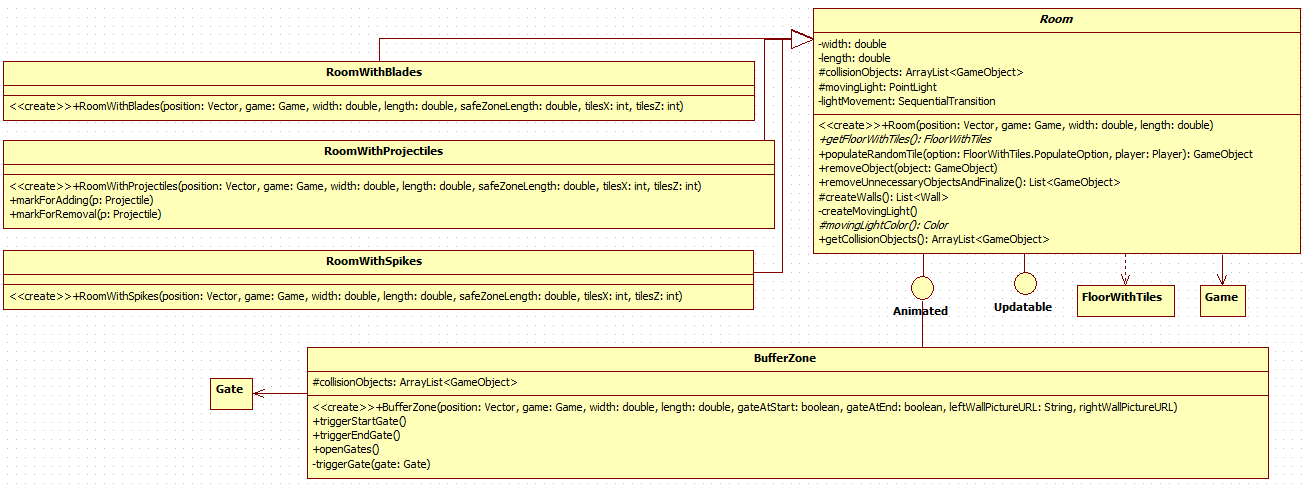
Због потенцијалне непрегледности неких од досадашњих дијаграма, у овом одељку су приложене увећане и другачије оријентисане слике.



Слика 4-1



Слика 5-2



Слика 6-1