Attack of the Trees:
Final Project

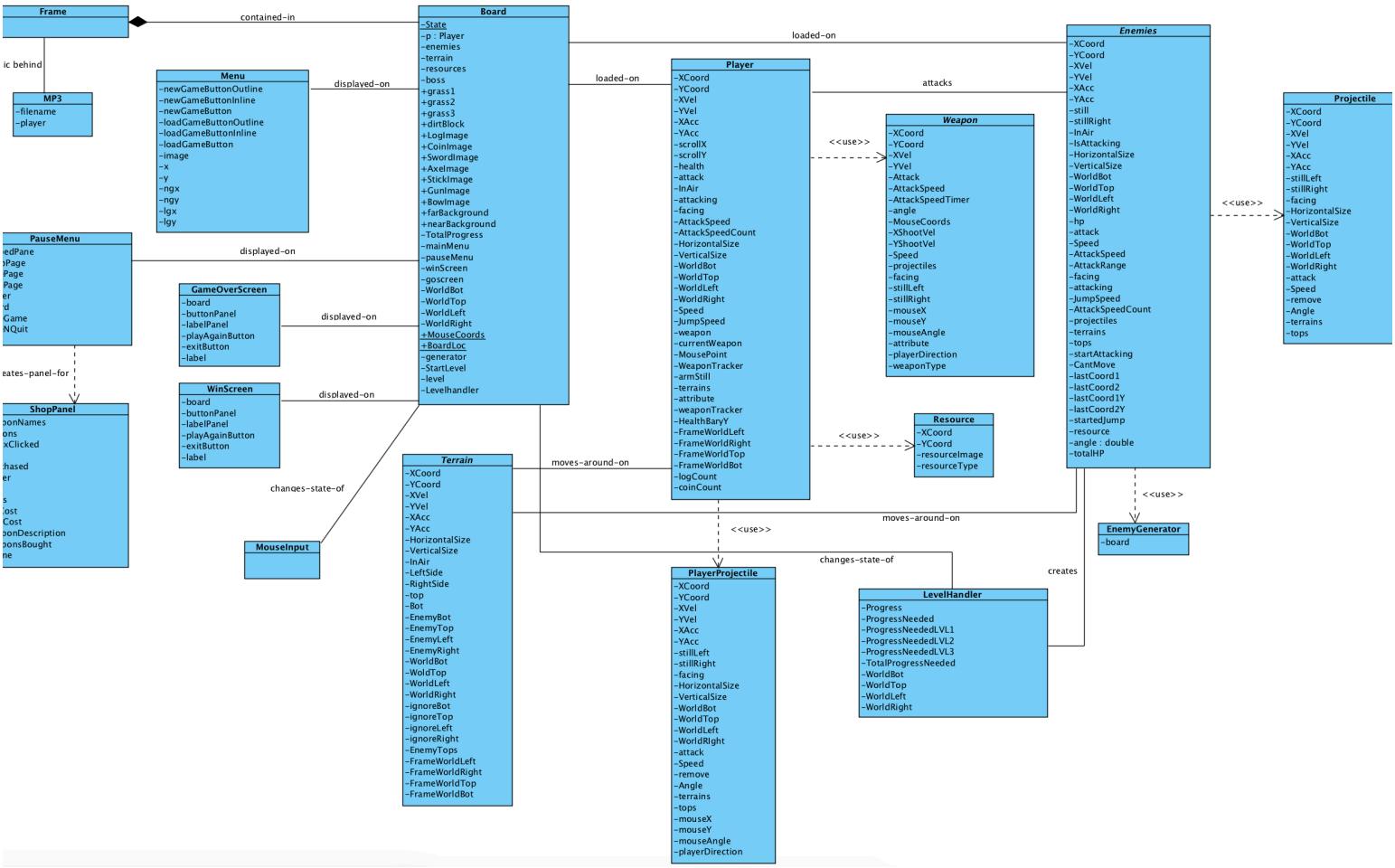
G11
Shadee Barzin
Andrew Ferguson
Michele Haque
Brendan Murphy
Fengyu Wang
CMPSC 48
Mohammad Amiri
March 13, 2015

I. Domain Analysis Artifacts

The following diagrams represent an Object-Oriented Analysis of the system. The system is viewed as a black box, with specific implementation details (i.e. methods and method names) omitted. The domain concepts, shown in the class diagram as classes, are identified and attributes of the classes listed as well. Associations between the various classes have been drawn.

A. Static Class Diagram

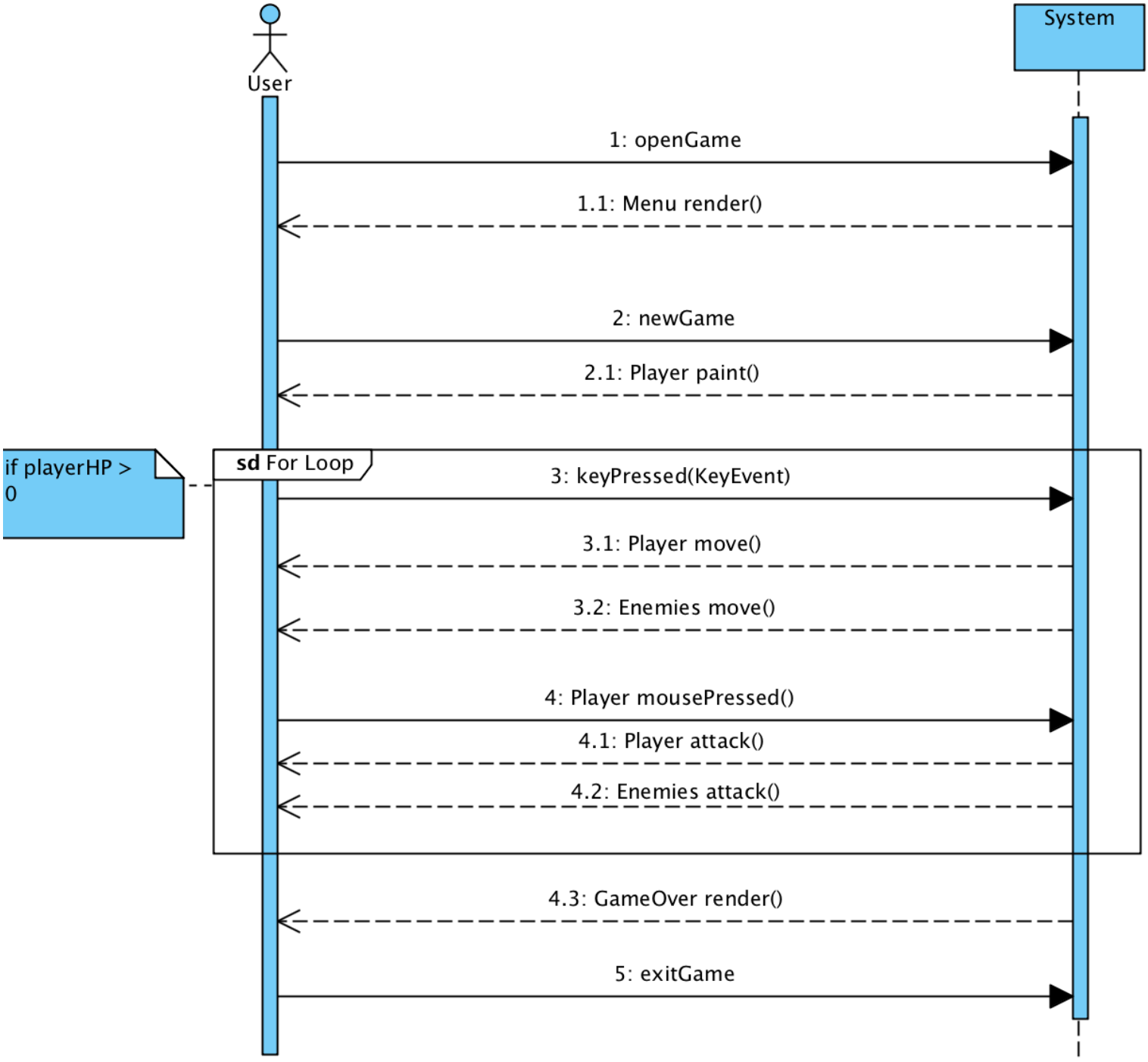
The static class diagram shows a visual representation of the domain concepts of the system and the associations between them. Some attributes have been identified based on the information the classes must hold and its interactions with other classes. It will be extended in the system design stage of development.



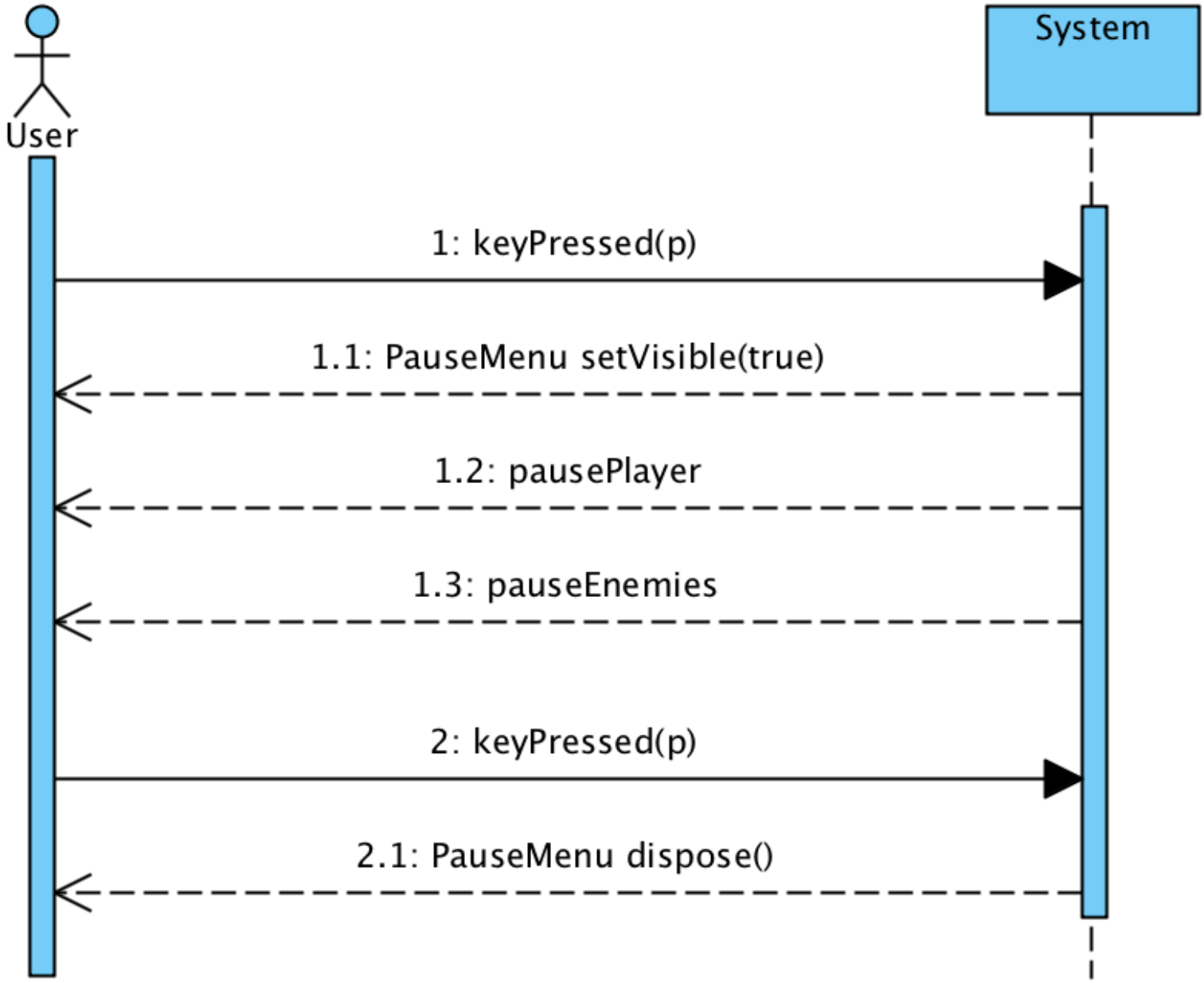
B. System Sequence Diagrams

The following system sequence diagrams illustrate two individual use cases of the system. The first shows the user's interactions with the system in carrying out the "Play Game" scenario, when the user plays the game through to the end normally. The second is for the "Pause Game" scenario, which occurs when the user is playing the game and decides to pause the game for any reason.

1. Play Game Scenario



2. Pause Game Scenario



C. System Operation Contracts

The system operation contracts display the pre-conditions (things about the system that must be true in order for the specific method to be carried out) and post-conditions (associations created or destroyed, instance variables created or deleted, and attribute values modified) of two specific operations the system carries out. The first is for a move() method contained in the "Play Game" use case, which moves the character on the screen. The second paints the player onto the game board.

1. Contract CO1: move()

Operation:	move(terrain: ArrayList<terrain>)
Cross References:	Use Cases: Play Game
Pre-conditions:	A player is created in board
Post-conditions:	Player's coordinates change based upon keyboard input and is aware of the terrain

2. Contract CO2: paintPlayer()

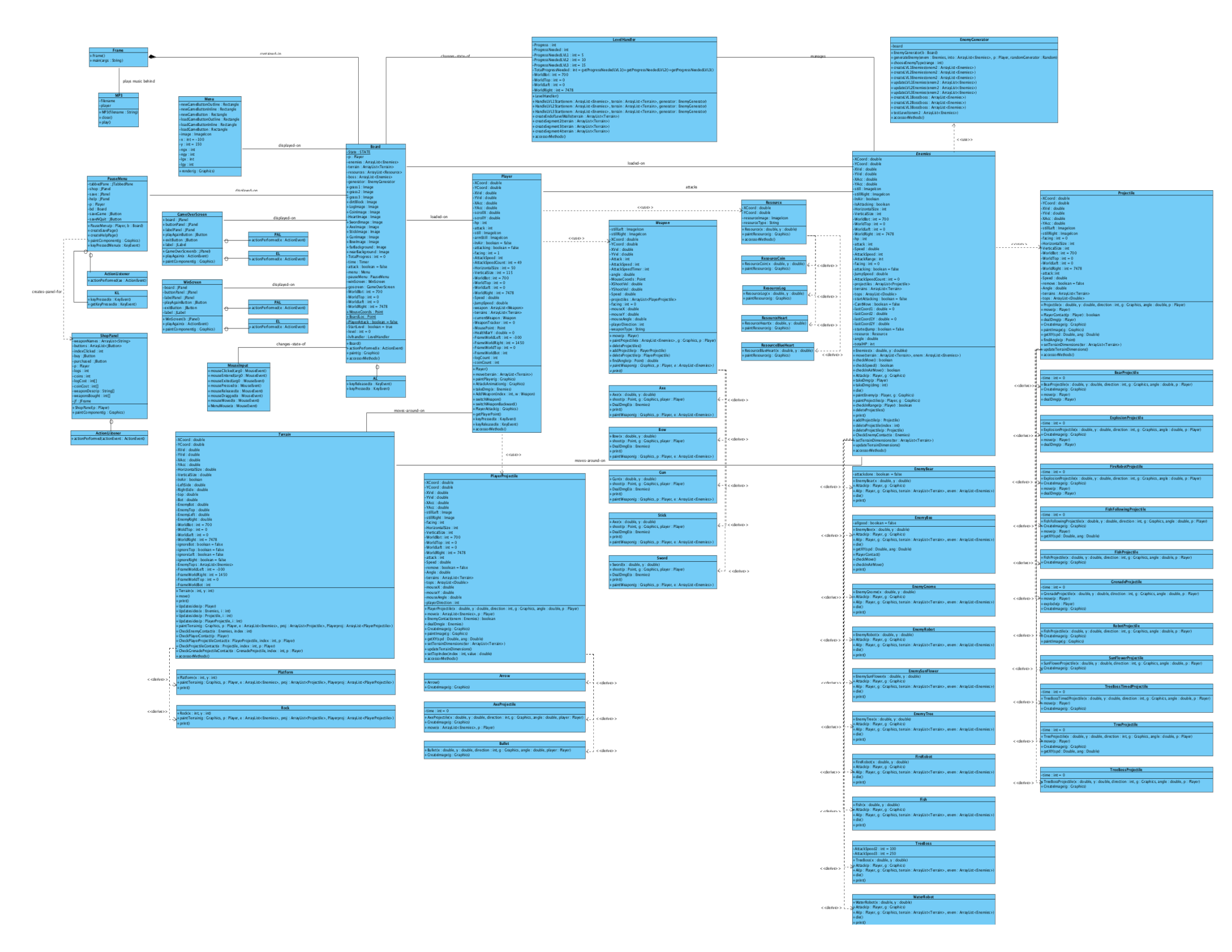
Operation:	paintPlayer(g: Graphics)
Cross References:	Use Cases: Play Game
Pre-conditions:	There is a frame with board created
Post-conditions:	Player is drawn on the Board

II. System Design Artifacts

These next diagrams represent an Object-Oriented Design view of the system. The attributes and method names of each class have been identified, as well as the attribute types and parameter and return types of the methods. At this stage, the specific implementation details have been decided; interface classes have been agreed upon and separated from the service classes. Additional classes have been created in order to account for the decisions made about implementation details. To add onto the associations drawn between classes in domain analysis, interactions between the classes have been modeled.

A. Extended Class Diagram

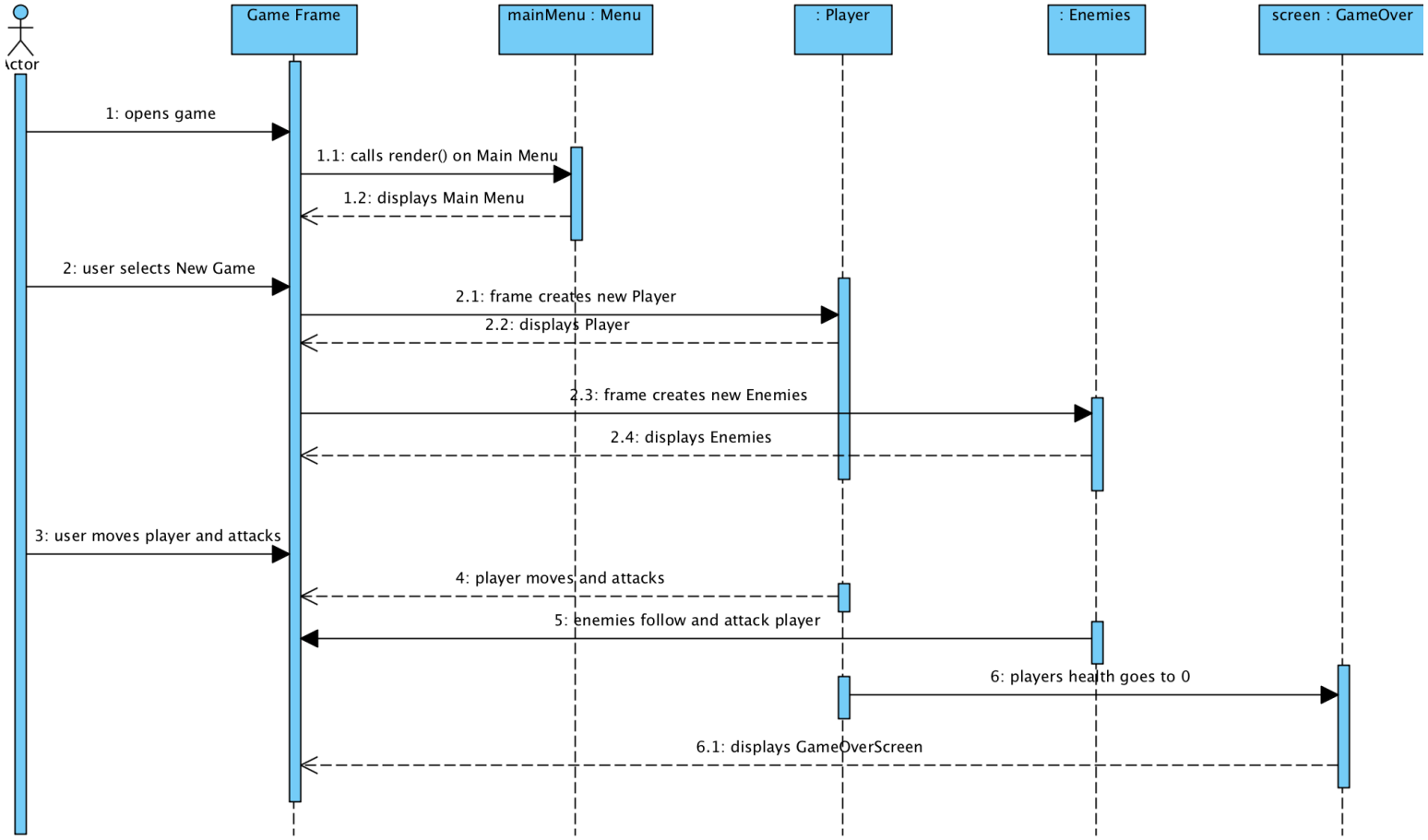
The extended class diagram builds upon the static class diagram from domain analysis. Operations have been included within each class, specifying the parameter names and types as well as return types. Additionally, derived classes have also been shown in the diagram. Due to its large size, the diagram is shown on the next page.


Frame
+ Frame()
+ main(args : String)
contained-in
changes-state-of
plays music behind
MP3
-filename
-player
+ MP3(filename : String)
+ close()
+ play()
Menu
-newGameButtonOutline : Rectangle
-newGameButtonInline : Rectangle
-newGameButton : Rectangle
-loadGameButtonOutline : Rectangle
-loadGameButtonInline : Rectangle
-loadGameButton : Rectangle
-image : ImageIcon
-x : int = -100
-y : int = 150
-ngx : int
-ngy : int
-lgx : int
-lgy : int
+ render(g : Graphics)
displayed-on
Board
-State : STATE
-p : Player
-enemies : ArrayList<Enemies>
-terrain : ArrayList<Terrain>
-resources : ArrayList<Resource>
-boss : ArrayList<Enemies>
-generator : EnemyGenerator
+ grass1 : Image
+ grass2 : Image
+ grass3 : Image
+ dirtBlock : Image
+ logImage : Image
+ Cointmage : Image
-HeartImage : Image
+ SwordImage : Image
+ AxeImage : Image
+ StickImage : Image
+ GunImage : Image
+ BowImage : Image
+ fullbackground : Image
+ mainBackground : Image
-TotalProgress : int = 0
-time : Timer
-attack : boolean = false
-menu : Menu
-pauseMenu : PauseMenu
-winScreen : WinScreen
-goscreen : GameOverScreen
-WorldBot : int = 700
-WorldTop : int = 0
-WorldLeft : int = 0
-WorldRight : int = 7478
+ MouseCoords : Point
+ BoardLoc : Point
- PlayerAttack : boolean = false
-StartLevel : boolean = true
-level : int = 0
-lvlhandler : LevelHandler
+ Board()
+ actionPerformed(e : ActionEvent)
+ paint(g : Graphics)
+ accessorMethods()
loaded-on
LevelHandler
-Progress : int
-ProgressNeeded : int
-ProgressNeededLVL1 : int = 5
-ProgressNeededLVL2 : int = 10
-ProgressNeededLVL3 : int = 15
-TotalProgressNeeded : int = getProgressNeededLVL1()+getProgressNeededLVL2()+getProgressNeededLVL3()
-WorldBot : int = 700
-WorldTop : int = 0
-WorldLeft : int = 0
-WorldRight : int = 7478
+ LevelHandler()
+ HandleLVL1Startenem : ArrayList<Enemies>, terrain : ArrayList<Terrain>, generator : EnemyGenerator)
+ HandleLVL2Startenem : ArrayList<Enemies>, terrain : ArrayList<Terrain>, generator : EnemyGenerator)
+ HandleLVL3Startenem : ArrayList<Enemies>, terrain : ArrayList<Terrain>, generator : EnemyGenerator)
+ createEndofLevelWall(terrain : ArrayList<Terrain>)
+ createSegment2(terrain : ArrayList<Terrain>)
+ createSegment3(terrain : ArrayList<Terrain>)
+ createSegment4(terrain : ArrayList<Terrain>)
+ accessorMethods()
manages
EnemyGenerator
-board
+ EnemyGenerator(b : Board)
+ generateEnemy(enem : Enemies, into : ArrayList<Enemies>, p : Player, randomGenerator : Random)
+ chooseEnemyType(range : int)
+ createLVL1Enemies(enem2 : ArrayList<Enemies>)
+ createLVL2Enemies(enem2 : ArrayList<Enemies>)
+ createLVL3Enemies(enem2 : ArrayList<Enemies>)
+ updateLVL1Enemies(enem2 : ArrayList<Enemies>)
+ updateLVL2Enemies(enem2 : ArrayList<Enemies>)
+ updateLVL3Enemies(enem2 : ArrayList<Enemies>)
+ createLVL1Bossboss : ArrayList<Enemies>)
+ createLVL2Bossboss : ArrayList<Enemies>)
+ createLVL3Bossboss : ArrayList<Enemies>)
+ testLevelItem2 : ArrayList<Enemies>)
+ accessorMethods()
<<use>>
displayed-on
PauseMenu
-tabbedPane : JTabbedPane
-shop : JPanel
-save : JPanel
-help : JPanel
-p : Player
-bd : Board
-saveGame : JButton
-saveAndQuit : JButton
+ PauseMenu(p : Player, b : Board)
+ createSavePage()
+ createHelpPage()
+ paintComponent(g : Graphics)
+ keyPressed(Menu)(k : KeyEvent)
ActionListener
+ actionPerformed(ae : ActionEvent)
KL
+ keyPressed(k : KeyEvent)
+ getKeyPressed(k : KeyEvent)
creates-panel-for
GameOverScreen
+ board : JPanel
+ buttonPanel : JPanel
+ labelPanel : JPanel
+ playAgainButton : JButton
+ exitButton : JButton
+ label : JLabel
+ GameOverScreen(b : JPanel)
+ playAgain(e : ActionEvent)
+ paintComponent(g : Graphics)
displayed-on
PAL
+ actionPerformed(e : ActionEvent)
EL
+ actionPerformed(e : ActionEvent)
WinScreen
-board : JPanel
-buttonPanel : JPanel
-labelPanel : JPanel
-playAgainButton : JButton
-exitButton : JButton
-label : JLabel
+ WinScreen(b : JPanel)
+ playAgain(e : ActionEvent)
+ paintComponent(g : Graphics)
displayed-on
PAL
+ actionPerformed(e : ActionEvent)
EL
+ actionPerformed(e : ActionEvent)
changes-state-of
ShopPanel
-weaponNames : ArrayList<String>
-buttons : ArrayList<JButton>
-indexClicked : int
-buy : JButton
-purchased : JButton
-p : Player
-logs : int
-coins : int
-logCost : int[]
-coinCost : int[]
-weaponDesrip : String[]
-weaponsBought : int[]
-jf : JFrame
+ ShopPanel(p : Player)
+ paintComponent(g : Graphics)
ActionListener
+ actionPerformed(actionEvent : ActionEvent)
MouseInput
+ mouseClicked(arg0 : MouseEvent)
+ mouseEntered(arg0 : MouseEvent)
+ mouseExited(arg0 : MouseEvent)
+ mousePressed(e : MouseEvent)
+ mouseReleased(e : MouseEvent)
+ mouseDragged(e : MouseEvent)
+ mouseMoved(e : MouseEvent)
+ MenuMouse(e : MouseEvent)
KL
+ keyReleased(k : KeyEvent)
+ keyPressed(k : KeyEvent)
moves-around-on
Terrain
-XCoord : double
-YCoord : double
-XVel : double
-YVel : double
-XAcc : double
-YAcc : double
-HorizontalSize : double
-VerticalSize : double
-isAir : boolean
-LeftSide : double
-RightSide : double
-Top : double
-Bot : double
-EnemyBot : double
-EnemyTop : double
-EnemyLeft : double
-EnemyRight : double
-WorldBot : int = 700
-WorldTop : int = 0
-WorldLeft : int = 0
-WorldRight : int = 7478
-ignoreBot : boolean = false
-ignoreTop : boolean = false
-ignoreLeft : boolean = false
-ignoreRight : boolean = false
-EnemyTops : ArrayList<Enemies>
-FrameWorldLeft : int = -100
-FrameWorldRight : int = 1410
-FrameWorldTop : int = 0
-FrameWorldBot : int
+ Terrain(x : int, y : int)
+ move()
+ print()
+ Updatesides(p : Player)
+ Updatesides(e : Enemies, i : int)
+ Updatesides(p : Projectile, i : int)
+ Updatesides(p : PlayerProjectile, i : int)
+ paintTerraing : Graphics, p : Player, e : ArrayList<Enemies>, proj : ArrayList<Projectile>, Playerproj : ArrayList <PlayerProjectile>)
+ CheckEnemyContact(x : Enemies, index : int)
+ CheckPlayerContact(p : Player)
+ CheckPlayerProjectileContact(x : PlayerProjectile, index : int, p : Player)
+ CheckProjectileContact(x : Projectile, index : int, p : Player)
+ CheckGrenadeProjectileContact(x : GrenadeProjectile, index : int, p : Player)
+ accessorMethods()
Platform
+ Platform(x : int, y : int)
+ paintTerraing : Graphics, p : Player, e : ArrayList<Enemies>, proj : ArrayList<Projectile>, Playerproj : ArrayList <PlayerProjectile>)
+ print()
<<derive>>
Rock
+ Rock(x : int, y : int)
+ paintTerraing : Graphics, p : Player, e : ArrayList<Enemies>, proj : ArrayList<Projectile>, Playerproj : ArrayList <PlayerProjectile>)
+ print()
<<derive>>
Player
-XCoord : double
-YCoord : double
-XVel : double
-YVel : double
-XAcc : double
-YAcc : double
-scrollX : double
-scrollY : double
-hp : int
-attack : int
-still : ImageIcon
-armStill : ImageIcon
-isAir : boolean = false
-attacking : boolean = false
-facing : int = 1
-AttackSpeed : int
-AttackSpeedCount : int = 49
-HorizontalSize : int = 50
-VerticalSize : int = 115
-WorldBot : int = 700
-WorldTop : int = 0
-WorldLeft : int = 0
-WorldRight : int = 7478
-Speed : double
-jumpSpeed : double
-weapon : ArrayList<Weapon>
-terrains : ArrayList<Terrain>
-currentWeapon : Weapon
-WeaponTracker : int = 0
-MousePoint : Point
-HealthBarY : double = 0
-FrameWorldLeft : int = -300
-FrameWorldRight : int = 1450
-FrameWorldTop : int = 0
-FrameWorldBot : int
-logCount : int
-coinCount : int
+ Player()
+ movet(terrain : ArrayList<Terrain>)
+ paintPlayer(g : Graphics)
+ AttackAnimation(g : Graphics)
+ takeDmg(x : Enemies)
+ AddWeapon(index : int, w : Weapon)
+ switchWeapon()
+ switchWeaponBackward()
+ PlayerAttack(g : Graphics)
+ getPlayerPoint()
+ keyPressed(k : KeyEvent)
+ keyReleased(k : KeyEvent)
+ accessorMethods()
<<use>>
attacks
Enemies
-XCoord : double
-YCoord : double
-XVel : double
-YVel : double
-XAcc : double
-YAcc : double
-still : ImageIcon
-stillRight : ImageIcon
-inAir : boolean
-isAttacking : boolean
-HorizontalSize : int
-VerticalSize : int
-WorldBot : int = 700
-WorldTop : int = 0
-WorldLeft : int = 0
-WorldRight : int = 7478
-hp : int
-attack : int
-Speed : double
-AttackSpeed : int
-AttackRange : int
-facing : int = 0
-attacking : boolean = false
-jumpSpeed : double
-AttackSpeedCount : int = 0
-projectiles : ArrayList<Projectile>
-terrains : ArrayList<Terrain>
-tops : ArrayList<Double>
-isantAttacking : boolean = false
-CanMove : boolean = false
-lastCoord1 : double = 0
-lastCoord2 : double
-lastCoord1Y : double = 0
-lastCoord2Y : double
-istartedjump : boolean = false
-resource : Resource
-angle : double
-totalHP : int
+ Enemies(x : double, y : double)
+ move(terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ checkMove() : boolean
+ checkSpeed() : boolean
+ checkInAirMove() : boolean
+ AttackUp : Player, g : Graphics)
+ takeDmg(p : Player)
+ takeDmg(dmg : int)
+ die()
+ paintEnemy(p : Player, g : Graphics)
+ paintProjectiles(p : Player, g : Graphics)
+ checkInRange(p : Player) : boolean
+ deleteProjectiles()
+ print()
+ addProjectile(p : Projectile)
+ deleteProjectile(index : int)
+ deleteProjectile(p : Projectile)
+ CheckEnemyContact(x : Enemies)
+ setTerrainDimensions(ter : ArrayList<Terrain>)
+ updateTerrainDimensions()
+ accessorMethods()
<<use>>
<<use>>
Resource
-XCoord : double
-YCoord : double
-resourceImage : ImageIcon
-resourceType : String
+ Resource(x : double, y : double)
+ paintResource(g : Graphics)
+ accessorMethods()
ResourceCoin
+ ResourceCoin(x : double, y : double)
+ paintResource(g : Graphics)
ResourceLog
+ ResourceLog(x : double, y : double)
+ paintResource(g : Graphics)
ResourceHeart
+ ResourceHeart(x : double, y : double)
+ paintResource(g : Graphics)
ResourceBlueHeart
+ ResourceBlueHeart(x : double, y : double)
+ paintResource(g : Graphics)
<<derive>>
<<derive>>
<<derive>>
<<derive>>
Weapon
-stillLeft : ImageIcon
-stillRight : ImageIcon
-XCoord : double
-YCoord : double
-XVel : double
-YVel : double
-Attack : int
-AttackSpeed : int
-AttackSpeedTimer : int
-angle : double
-MouseCoords : Point
-XShootVel : double
-YShootVel : double
-Speed : double
-projectiles : ArrayList<PlayerProjectile>
-facing : int = 0
-mouseX : double
-mouseY : double
-mouseAngle : double
-playerDirection : int
-weaponType : String
+ move(p : Player)
+ paintProjectiles : ArrayList<Enemies>, g : Graphics, p : Player)
+ deleteProjectiles()
+ addProjectile(p : PlayerProjectile)
+ deleteProjectile : PlayerProjectile)
+ findAngle(p : Point) : double
+ paintWeapon(g : Graphics, p : Player, e : ArrayList<Enemies>)
+ print()
<<use>>
<<use>>
Axe
+ Axe(x : double, y : double)
+ shoot(p : Point, g : Graphics, player : Player)
+ DealDmg(Ie : Enemies)
+ print()
+ paintWeapon(g : Graphics, p : Player, e : ArrayList<Enemies>)
Bow
+ Bow(x : double, y : double)
+ shoot(p : Point, g : Graphics, player : Player)
+ DealDmg(Ie : Enemies)
+ print()
+ paintWeapon(g : Graphics, p : Player, e : ArrayList<Enemies>)
Gun
+ Gun(x : double, y : double)
+ shoot(p : Point, g : Graphics, player : Player)
+ DealDmg(Ie : Enemies)
+ print()
+ paintWeapon(g : Graphics, p : Player, e : ArrayList<Enemies>)
Stick
+ Axe(x : double, y : double)
+ shoot(p : Point, g : Graphics, player : Player)
+ DealDmg(Ie : Enemies)
+ print()
+ paintWeapon(g : Graphics, p : Player, e : ArrayList<Enemies>)
Sword
+ Sword(x : double, y : double)
+ shoot(p : Point, g : Graphics, player : Player)
+ DealDmg(Ie : Enemies)
+ print()
+ paintWeapon(g : Graphics, p : Player, e : ArrayList<Enemies>)
<<derive>>
<<derive>>
<<derive>>
<<derive>>
<<derive>>
PlayerProjectile
-XCoord : double
-YCoord : double
-XVel : double
-YVel : double
-XAcc : double
-YAcc : double
-stillLeft : Image
-stillRight : Image
-facing : int
-HorizontalSize : int
-VerticalSize : int
-WorldBot : int = 700
-WorldTop : int = 0
-WorldLeft : int = 0
-WorldRight : int = 7478
-attack : int
-Speed : double
-remove : boolean = false
-Angle : double
-terrains : ArrayList<Terrain>
-tops : ArrayList<Double>
-mouseX : double
-mouseY : double
-mouseAngle : double
-playerDirection : int
+ PlayerProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ mouseIs : ArrayList<Enemies>, p : Player)
+ EnemyContactUnem : Enemies) : boolean
+ dealDmg(e : Enemies)
+ CreateImage(g : Graphics)
+ paintImage(g : Graphics)
+ getXVel(pd : Double, ang : Double)
+ setTerrainDimensions(ter : ArrayList<Terrain>)
+ updateTerrainDimensions()
+ setTopIndex(index : int, value : double)
+ accessorMethods()
Arrow
+ Arrow()
+ CreateImage(g : Graphics)
AxeProjectile
-time : int = 0
+ AxeProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, player : Player)
+ CreateImage(g : Graphics)
+ mouseIs : ArrayList<Enemies>, p : Player)
Bullet
+ Bullet(x : double, y : double, direction : int, g : Graphics, angle : double, player : Player)
+ CreateImage(g : Graphics)
<<derive>>
<<derive>>
<<derive>>
EnemyBear
-attackdone : boolean = false
+ EnemyBear(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
EnemyBee
-isgood : boolean = false
+ EnemyBee(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ getXVel(pd : Double, ang : Double)
+ PlayerContact()
+ checkMove()
+ checkInAirMove()
+ print()
EnemyGnome
+ EnemyGnome(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
EnemyRobot
+ EnemyRobot(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
EnemySunFlower
+ EnemySunFlower(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
EnemyTree
+ EnemyTree(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
FireRobot
+ FireRobot(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
Fish
+ Fish(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
TreeBoss
-AttackSpeed2 : int = 100
-AttackSpeed3 : int = 250
+ TreeBoss(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
WaterRobot
+ WaterRobot(x : double, y : double)
+ Attack(p : Player, g : Graphics)
+ Alp : Player, g : Graphics, terrain : ArrayList<Terrain>, enem : ArrayList<Enemies>)
+ die()
+ print()
<<derive>>
<<derive>>
<<derive>>
<<derive>>
<<derive>>
<<derive>>
<<derive>>
<<derive>>
<<derive>>
<<derive>>
Projectile
-XCoord : double
-YCoord : double
-XVel : double
-YVel : double
-XAcc : double
-YAcc : double
-stillLeft : ImageIcon
-stillRight : ImageIcon
-facing : int = 0
-HorizontalSize : int
-VerticalSize : int
-WorldBot : int = 700
-WorldTop : int = 0
-WorldLeft : int = 0
-WorldRight : int = 7478
-attack : int
-Speed : double
-remove : boolean = false
-Angle : double
-terrains : ArrayList<Terrain>
-tops : ArrayList<Double>
+ Projectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ move(p : Player)
+ PlayerContact(p : Player) : boolean
+ dealDmg(p : Player)
+ CreateImage(g : Graphics)
+ paintImage(g : Graphics)
+ getXVel(pd : Double, ang : Double)
+ findAngle(p : Point)
+ setTerrainDimensions(ter : ArrayList<Terrain>)
+ updateTerrainDimensions()
+ accessorMethods()
<<derive>>
BearProjectile
-time : int = 0
+ BearProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ CreateImage(g : Graphics)
+ move(p : Player)
+ dealDmg(p : Player)
<<derive>>
ExplosionProjectile
-time : int = 0
+ ExplosionProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ CreateImage(g : Graphics)
+ move(p : Player)
+ dealDmg(p : Player)
<<derive>>
FireRobotProjectile
-time : int = 0
+ ExplosionProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ CreateImage(g : Graphics)
+ move(p : Player)
+ dealDmg(p : Player)
<<derive>>
FishFollowingProjectile
-time : int = 0
+ FishFollowingProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ CreateImage(g : Graphics)
+ move(p : Player)
+ getXVel(pd : Double, ang : Double)
<<derive>>
FishProjectile
+ FishProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ CreateImage(g : Graphics)
<<derive>>
GrenadeProjectile
-time : int = 0
+ GrenadeProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ move(p : Player)
+ explode(p : Player)
+ CreateImage(g : Graphics)
<<derive>>
RobotProjectile
+ FishProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ CreateImage(g : Graphics)
+ paintImage(g : Graphics)
<<derive>>
SunFlowerProjectile
+ SunFlowerProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ CreateImage(g : Graphics)
<<derive>>
TreeBossTimedProjectile
-time : int = 0
+ TreeBossTimedProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ move(p : Player)
+ CreateImage(g : Graphics)
<<derive>>
TreeProjectile
-time : int = 0
+ TreeProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ move(p : Player)
+ CreateImage(g : Graphics)
+ getXVel(pd : Double, ang : Double)
<<derive>>
TreeBossProjectile
-time : int = 0
+ TreeBossProjectile(x : double, y : double, direction : int, g : Graphics, angle : double, p : Player)
+ CreateImage(g : Graphics)
<<derive>>

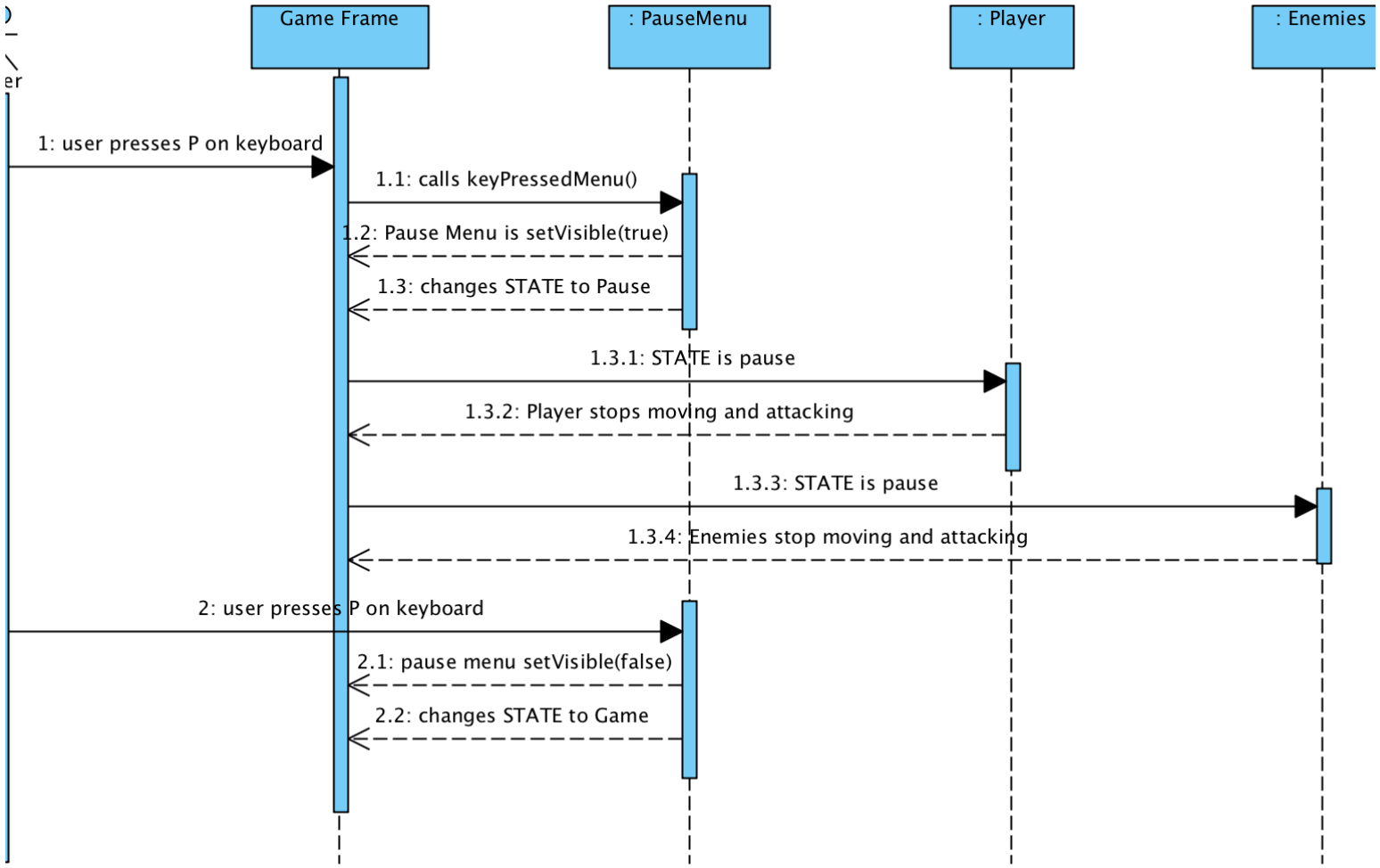
B. Interaction Diagrams (Sequence Diagrams)

The sequence diagrams are similar to the system sequence diagrams from the domain analysis stage, but instead of treating the system as a black box, the classes that send messages to one another are displayed. In the “Play Game” and “Pause Game” sequences, the user interacts with an interface that is constantly there. The interface in turn sends and receives requests from the service classes.

1. Play Game Sequence

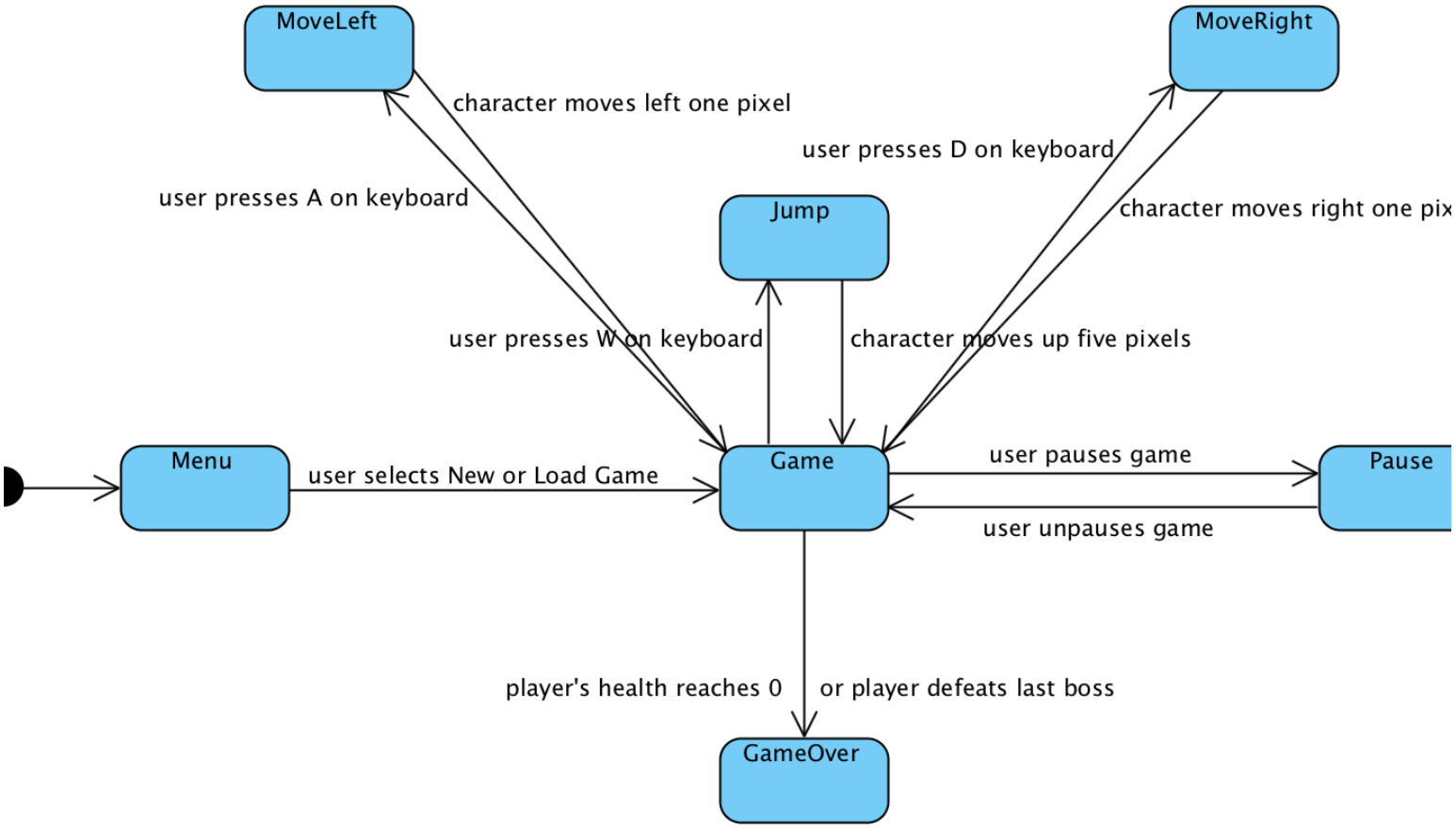


2. Pause Game Sequence



C. State Diagram

The state diagram shows all possible states of gameplay and the actions that change the game from one state to another. The game starts off in the Menu state, then can be changed to Game state. From there, there are several options: the player can move left, right, and jump, and the game can be paused and end.



III. Glossary

Term	Definition/Description
AI()	tells the enemies object what to do next based on it's current situation
AL	Private class inside Board.java, that detects if a key was pressed or released, and if so it calls the Player's keyReleased()/keyPressed() method and the pause menu's keyPressedMenu() method
ArrayList<Enemies> boss	A list that stores all the bosses in the game during that level
ArrayList<Enemies> enemies	List that keeps track of the enemies in the game
ArrayList<Resource> resources	A list that keeps tracks of the weapons dropped by enemies that haven't been picked up by the player
ArrayList<Terrain> terrain	List that stores the terrain of the panel
Arrow/Arrow.java	PlayerProjectile that shoots from a bow weapon
attacking	boolean that tracks whether or not the object is attacking
Attack()	tells one object to attack another
AttackSpeed	How quickly the object is able to attack
AttackSpeedCount	numerical value that when reaches the attackspeed value, performs attack then

	resets
AttackRange	How far the object is able to attack
Axe.java	Weapon that is an axe
AxeProjectile.java	PlayerProjectile that shoots from an axe, but doesn't have an image because it doesn't have a range
BearProjectile.java	Projectile that the bear enemy shoots at the player
Board.java	Displays the game board. Where player, enemies, terrain, etc. are shown.
Board::actionPerformed()	Checks to see if the user does anything and changes the players, enemies, and bosses attributes accordingly
boolean PlayerAttack	Boolean that keeps track of if the player is attacking of not
Boss	Enemy player encounters at end of level
Bow/Bow.java	A weapon that propels arrows.
Bullet.java	Class to hold projectiles shot from Player's gun. Used to attack enemies. Subclass of PlayerProjectile.java
cantMove	Enemies boolean that keeps track of whether the object is stuck or not
Checkpoint	A set point in player's progress of the game. Game autosaves at this point.
CreateImage(Graphics)	Draws the image of the object that uses it (e.g.

	Arrow, Bow, TreeProjectile)
currentWeapon	the current weapon being used by the Player
dealDmg()	Method projectiles have that lowers the player's hp by the attack strength of the weapon
DealDmgE()	Calls the enemy's takeDmg() method when invoked, passing in the weapon's attack strength
Enemies.java	Abstract parent class for all forms of enemies
Enemies::checkInRange()	checks if the enemies object is in attack range of the player
Enemies::checkMove()	checks if enemies object is stuck
Enemies::deleteProjectile()	if the enemies' projectile needs to be deleted, it deletes it
Enemies::findAngle()	finds the angle between the enemy and the player
Enemies::PaintProjectile()	moves the enemies' projectiles, then paints the projectiles
Enemy	Characters player must defeat to win game
EnemyBear.java	An enemy player must defeat. Image is a bear.
EnemyBee.java	An enemy player must defeat. Image is a bee.
EnemyGenerator.java	Class that randomly generates enemies off screen that the player must defeat

EnemyGnome.java	An enemy player must defeat. Image is a gnome.
EnemyRobot.java	An enemy player must defeat. Image is a robot.
EnemySunFlower.java	An enemy player must defeat. Image is a sunflower.
EnemyTree.java	An enemy player must defeat. Image is a tree.
ExplosionProjectile.java	Projectile an enemy shoots
enum STATE	Holds the different state constants (MENU, GAME, PAUSE, GAMEOVER, LOAD)
facing	keeps track of which way the object is facing using 0 for left and 1 for right
FireRobot.java	An enemy player must defeat. Image is a robot.
FireRobotProjectile.java	The projectile that the enemy fireRobot shoots at the player
Fish.java	A boss that the player must defeat, who's image is a fish
FishFollowingProjectile.java	One type of projectile that the fish boss shoots, that follows the player
FishProjectile.java	Another type of projectile that fish boss shoots at the player
Frame.java	Creates new frame for board class to display game on.
GameOverScreen.java	JFrame that shows up when the player loses the game, which will allow them to exit or play again
GrenadeProjectile.java	Projectile that the enemy

	shoots at the player.
Gun.java	Weapon used by player. Shoots bullets.
Health Bar	Tracks player's condition. Goes down when hit by an enemy.
help	JPanel in the pause menu that tells the user how to play the game
Image AxeImage	Image in board that is an axe
Image BowImage	Attribute in board that is the image for the bow weapon
Image CoinImage	Image for the coin resource in board
Image DirtBlock	Image for the dirt in the board panel
Image Grass	Image the grass at the bottom of the screen in board
Image GunImage	Attribute in board that holds the image for the gun
Image LogImage	Image for the log resource in board
Image nearBackground	Image for the closer part of the background of the game in the board class
Image Near2	Another image that makes up the closer background in the game
Image farBackground	Image in the board that is the one of the backgrounds for the game in the board class
Image Far2	Image that makes up the background in the game

Image Far3	Another image that its a part of the background of the game
Image StickImage	Attribute in board that holds the image for the stick
Image SwordImage	Image in Board that is a sword
int level	Int thtat stores what level that player is at in the game
int WorldBot	Attribute in board that stores the location of the bottoms of the screen
int WorldLeft	Int in the board that is the x coordinate of the leftmost part of the frame
int WorldRight	Attribute in board that is the x coordinate of the rightmost part of the game
int WorldTop	Int in board that is the y coordinate of the top of the screen
Inventory Shop	Where the player can spend his/her collected resources
JumpSpeed	How fast the object is able to jump
KL	Private class in PauseMenu that extends from KeyAdapter; listens to key input
lastCoord1	Enemy double that keeps track of the object's last x coord
lastCoord2	Enemy double that keeps track of the object's second to last x coord
LevelHandler.java	Class that tracks the progress needed/achieved

	for user to complete the game
loadGameButton	Draws a button on Main Menu screen that should allow you to load the game
Lumberjack	User controlled character
Main Menu/Menu.java	Display that allows a player to start a new game or load an existing game
Menu::render(Graphics)	Method in the Menu.java class that paints the menu onto Board.java
MouseInput.java	Gets users input from mouse so player can attack
MouseInput::MenuMouse(MouseEvent)	Called by mousePressed to see if user clicks on buttons in Main Menu, and changes the state accordingly
MouseInput::mouseDragged(MouseEvent)	Sees if the user drags the mouse and gives Board the coordinates of the mouse input and tells Board that PlayerAttack is true
MouseInput::mousePressed(MouseEvent)	Gets input if the mouse is pressed, and then either calls MenuMouse or gives the board the coordinates of where the mouse was clicked
MouseInput::mouseReleased(MouseEvent)	Recognizes if the user released the mouse, and the method changes the boolean in Board of PlayerAttack to false
MousePoint	Point in player that tracks the Mouse Coords
move()	adjusts the object's velocities based on acceleration, the

	adjusts coordinates based on velocity. Also checks contact, and resolves contact issues
newGameButton	Creates a button in the Main Menu screen will allow you to start a new game
paint()	Method in board that paints the panel based on which state it is in. Draws all of the background images, enemies, terrain, and player.
paintWeapon()	When called, it invokes the paintProjectile and deleteProjectiles methods
Pause Menu/ PauseMenu.java	Frame that appears when user presses p on keyboard. Has shop, save, and help options.
PauseMenu::createHelpPage()	This initializes the JPanel help and has text to describe how to play the game
PauseMenu::createSavePage()	This initializes the JPanel save and adds buttons to save and to save and exit
PauseMenu::createShopPage()	This initializes the JPanel shop and adds weapons to the panel
PauseMenu::keyPressedMenu(KeyEvent)	Takes the input from the KeyAdapter and sees if “P” was pressed; if so, change State from Game to Pause or vice versa
Platform.java	A type of terrain that is a board the player and enemies must traverse.
Player.java	Controls user’s character
Player::AddWeapon()	adds another weapon and

	makes it the current weapon
Player::AttackAnimation()	changes player image based on attacking/facing
Player::switchWeapon()	switches the player's current weapon
PlayerProjectile.java	Super class for various types of projectiles used by player to attack enemies (i.e. Bullet.java)
Point BoardLoc	A point that keeps track of where the frame is relative to the screen
Point MouseCoords	Point that keeps track of the coodinates of the mouse
Progress Bar	Tracks player's progress throughout the game. Depletes as game goes on.
projectiles	ArrayList of the Projectile object used by Enemy
Projectile.java	Super class for various projectiles used by enemies to attack player
Ramp.java	Subclass of Terrain.java. Type of obstacle for player and enemies to maneuver around.
Resource/Resource.java	Items player collects to advance game. Can use these to gain more advanced weapons. Super class for the different types of resources (i.e. ResourceCoin.java)
ResourceBlueHeart.java	Type of resource that the boss drops that will reset the player's health back to 100, if collected

ResourceCoin.java	Type of resource that the player can collect that is a coin
ResourceHeart.java	Type of resource that may be dropped by the enemies, which adds 20 hp to the player. It's image is a heart
ResourceLog.java	Resource dropped by trees that the player can collect to buy weapons. It's image is a log
RobotProjectile.java	Subclass of Projectile.java. Type of projectile used by robot to attack enemy.
Rock.java	Subclass of Terrain.java Type of obstacle for player and enemies to maneuver around
save	JPanel in the pause menu that will allow the user to save the game and/or exit
shop/ShopPanel.java	JPanel in the pause menu that displays the weapons
Shoot()	For a weapon, this method checks to see if the AttackSpeedTimer equals the AttackSpeed, and if it is, it will create a projectile and add it to the weapon's arraylist of projectiles
STATE state	Attribute in board that stores the current state of the game
Stick/Stick.java	A weak weapon the player can use. Extends Weapon.java
SunFlowerProjectile.java	A projectile that a sunflower shoots at the player. (Extends Projectile.java)

Sword	A pointy weapon
tabbedPane	JTabbedPane in the PauseMenu class that shows the different tabs in the pause menu
takeDmg()	subtracts health from the object by some value
Terrain.java	Superclass of obstacles player and enemies must maneuver around
TreeBoss.java	Final boss that the player must defeat, which is a huge tree
TreeBossProjectile.java	Type of projectile that the boss tree fires at the player
TreeBossTimedProjectile.java	Another type of projectile that the tree boss shoots at the player
TreeProjectile.java	A projectile fired by the tree enemy, which is a leaf
WaterRobot.java	Another type of enemy that is also a robot
Weapon/Weapon.java	Item used by player to defeat enemies
WeaponTracker	keeps track of which weapon to switch to next
WinScreen.java	JFrame that shows when the player wins the game, from here the player can play again or exit the game
XCoord	tracks the x coordinate of the object

YCoord	tracks the y coordinate of the object
XVelocity	the velocity in the x direction of the object
YVelocity	the velocity in the y direction of the object
still	holds the image of an object
LeftSide	The Left boundary of an object
RightSide	The Right boundary of an object
Top	The top boundary of an object
Bot	The bottom boundary of an object
ignoreLeft	Tells object to ignore the left boundary of the terrain
ignoreRight	Tells object to ignore the Right boundary of the terrain
ignoreTop	Tells object to ignore the Top boundary of the terrain
ignoreBot	Tells object to ignore the bottom boundary of the terrain

HorizontalSize	Horizontal size of the object
VerticalSize	Vertical size of the object
InRange	tracks if enemy is within attack range of the player
hp	tracks the remaining health of the object
paint()	paints the object to the screen
attack	the attack value of the object
speed	the movement speed of the object