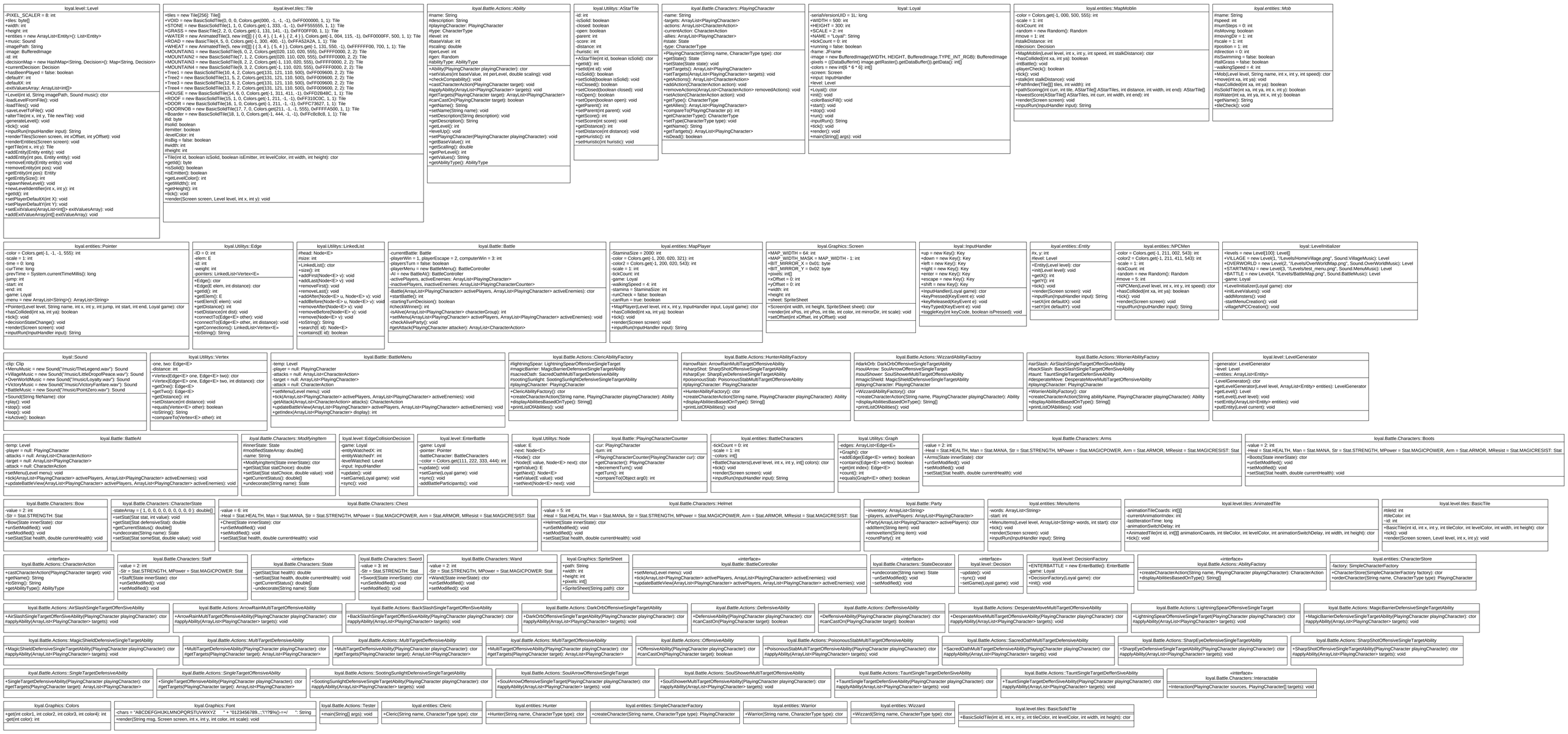
loyal.level::Level
-PIXEL_SCALER = 8: int
+tiles: byte[]
+width: int
+height: int
+entities = new ArrayList<Entity>(): List<Entity>
+music: Sound
-imagePath: String
-image: BufferedImage
-tit: int
-decisionMap = new HashMap<String, Decision>(): Map<String, Decision>
+currentDecision: Decision
+hasBeenPlayed = false: boolean
-defaultX: int
-defaultX: int
-exitValuesArray: ArrayList<int[]>
+Level(int id, String imagePath, Sound music): ctor
-loadLevelFromFile(): void
-fixedTiles(): void
-saveLevelToFile(): void
+alterTile(int x, int y, Tile newTile): void
-generateLevel(): void
+tick(): void
+inputRun(InputHandler input): String
+renderTiles(Screen screen, int xOffset, int yOffset): void
+renderEntities(Screen screen): void
+getTile(int x, int y): Tile
+addEntity(Entity entity): void
+addEntity(int pos, Entity entity): void
+removeEntity(Entity entity): void
+removeEntity(int pos): void
+getEntity(int pos): Entity
+getEntitySize(): int
+spawnNewLevel(): void
+newLevelIdentifier(int x, int y): int
+getId(): int
+setPlayerDefaultX(int X): void
+setPlayerDefaultY(int Y): void
+setExitValues(ArrayList<int[]> exitValuesArray): void
+addExitValueArray(int[] exitValueArray): void
loyal.level.tiles::Tile
+tiles = new Tile[256]: Tile[]
+VOID = new BasicSolidTile(0, 0, 0, Colors.get(000, -1, -1, -1), 0xFF000000, 1, 1): Tile
+STONE = new BasicSolidTile(1, 1, 0, Colors.get(-1, 333, -1, -1), 0xFF555555, 1, 1): Tile
+GRASS = new BasicTile(2, 2, 0, Colors.get(-1, 131, 141, -1), 0xFF00FF00, 1, 1): Tile
+WATER = new AnimatedTile(3, new int[][] { { 0, 4 }, { 1, 4 }, { 2, 4 } }, Colors.get(-1, 004, 115, -1), 0xFF0000FF, 500, 1, 1): Tile
+ROAD = new BasicTile(4, 5, 0, Colors.get(-1, 300, 400, -1), 0xFFA52A2A, 1, 1): Tile
+WHEAT = new AnimatedTile(5, new int[][] { { 3, 4 }, { 5, 4 } }, Colors.get(-1, 131, 550, -1), 0xFFFFFF00, 700, 1, 1): Tile
+MOUNTAIN1 = new BasicSolidTile(6, 0, 2, Colors.get(020, 110, 020, 555), 0xFFFF0000, 2, 2): Tile
+MOUNTAIN2 = new BasicSolidTile(7, 1, 2, Colors.get(020, 110, 020, 555), 0xFFFF0000, 2, 2): Tile
+MOUNTAIN3 = new BasicSolidTile(8, 2, 2, Colors.get(-1, 110, 020, 555), 0xFFFF0000, 2, 2): Tile
+MOUNTAIN4 = new BasicSolidTile(9, 3, 2, Colors.get(-1, 110, 020, 555), 0xFFFF0000, 2, 2): Tile
+Tree1 = new BasicSolidTile(10, 4, 2, Colors.get(131, 121, 110, 500), 0xFF009600, 2, 2): Tile
+Tree2 = new BasicSolidTile(11, 5, 2, Colors.get(131, 121, 110, 500), 0xFF009600, 2, 2): Tile
+Tree3 = new BasicSolidTile(12, 6, 2, Colors.get(131, 121, 110, 500), 0xFF009600, 2, 2): Tile
+Tree4 = new BasicSolidTile(13, 7, 2, Colors.get(131, 121, 110, 500), 0xFF009600, 2, 2): Tile
+HOUSE = new BasicSolidTile(14, 6, 0, Colors.get(-1, 311, 411, -1), 0xFFD2B48C, 1, 1): Tile
+ROOF = new BasicSolidTile(15, 1, 0, Colors.get(-1, 211, -1, -1), 0xFF31DC0C, 1, 1): Tile
+DOOR = new BasicSolidTile(16, 1, 0, Colors.get(-1, 211, -1, -1), 0xFFC73627, 1, 1): Tile
+DOORNOB = new BasicSolidTile(17, 7, 0, Colors.get(211, -1, -1, 555), 0xFFFFA500, 1, 1): Tile
+Boarder = new BasicSolidTile(18, 1, 0, Colors.get(-1, 444, -1, -1), 0xFFCBC8B, 1, 1): Tile
#id: byte
#solid: boolean
#emitter: boolean
-levelColor: int
#isBig = false: boolean
#width: int
#height: int
+Tile(int id, boolean isSolid, boolean isEmitter, int levelColor, int width, int height): ctor
+getId(): byte
+isSolid(): boolean
+isEmitter(): boolean
+getLevelColor(): int
+getWidth(): int
+getHeight(): int
+tick(): void
+render(Screen screen, Level level, int x, int y): void
loyal.Battle.Actions::Ability
#name: String
#description: String
#playingCharacter: PlayingCharacter
#type: CharacterType
#level: int
#baseValue: int
#scaling: double
#perLevel: int
#gen: Random
#abilityType: AbilityType
+Ability(PlayingCharacter playingCharacter): ctor
+setValues(int baseValue, int perLevel, double scaling): void
+checkCompability(): void
+castCharacterAction(PlayingCharacter target): void
#applyAbility(ArrayList<PlayingCharacter> targets): void
#getTargets(PlayingCharacter target): ArrayList<PlayingCharacter>
#canCastOn(PlayingCharacter target): boolean
+getDescription(): String
+getLevel(): int
+levelUp(): void
+setPlayingCharacter(PlayingCharacter playingCharacter): void
+getBaseValue(): int
+getScalling(): double
+getPerLevel(): int
+getValues(): String
+getAbilityType(): AbilityType
loyal.Utilityy::AStarTile
-id: int
-isSolid: boolean
-closed: boolean
-open: boolean
-parent: int
-score: int
-distance: int
-huristic: int
+AStarTile(int id, boolean isSolid): ctor
+getId(): int
+setId(int id): void
+isSolid(): boolean
+setSolid(boolean isSolid): void
+isClosed(): boolean
+setClosed(boolean closed): void
+isOpen(): boolean
+setOpen(boolean open): void
+getParent(): int
+setParent(int parent): void
+getScore(): int
+setScore(int score): void
+getDistance(): int
+setDistance(int distance): void
+getHuristic(): int
+setHuristic(int huristic): void
loyal.Battle.Characters::PlayingCharacter
-name: String
-targets: ArrayList<PlayingCharacter>
-actions: ArrayList<CharacterAction>
-currentAction: CharacterAction
-allies: ArrayList<PlayingCharacter>
#state: State
-type: CharacterType
+PlayingCharacter(String name, CharacterType type): ctor
+getState(): State
+setState(State state): void
+getTargets(): ArrayList<PlayingCharacter>
+setTargets(ArrayList<PlayingCharacter> targets): void
+getActions(): ArrayList<CharacterAction>
+addAction(CharacterAction action): void
+removeActions(ArrayList<CharacterAction> removedActions): void
+setAction(CharacterAction action): void
+getType(): CharacterType
+getAllies(): ArrayList<PlayingCharacter>
+compareToPlayingCharacter pi): int
+getCharacterType(): CharacterType
+setType(CharacterType type): void
+getName(): String
+getTargets(): ArrayList<PlayingCharacter>
+isDead(): boolean
loyal::Loyal
-serialVersionUID = 1L: long
+WIDTH = 500: int
+HEIGHT = 300: int
+SCALE = 2: int
+NAME = "Loyal": String
-tickCount = 0: int
+running = false: boolean
-frame: JFrame
-image = new BufferedImage(WIDTH, HEIGHT, BufferedImage.TYPE_INT_RGB): BufferedImage
-pixels = ((DataBufferInt) image.getRaster().getDataBuffer()).getData(): int[]
-colors = new int[6 * 6 * 6]: int[]
-screen: Screen
+input: InputHandler
#level: Level
+Loyal(): ctor
+init(): void
-colorBasicFill(): void
+start(): void
+stop(): void
+run(): void
-inputRun(): String
+tick(): void
+render(): void
+main(String[] args): void
loyal.entities::MapMoblin
-color = Colors.get(-1, 000, 500, 555): int
-scale = 1: int
-tickCount: int
-random = new Random(): Random
#move = 1: int
#stalkDistance: int
#decision: Decision
+MapMoblin(Level level, int x, int y, int speed, int stalkDistance): ctor
+hasCollided(int xa, int ya): boolean
-initBattle(): void
-playerCheck(): boolean
+tick(): void
+stalk(int stalkDistance): void
+PathFinder(Tile[][] tiles, int width): int
+pathScoring(int curr, int tile, AStarTile[] AStarTiles, int distance, int width, int end): AStarTile[]
+lowestScore(AStarTile[] AStarTiles, int curr, int width, int end): int
+render(Screen screen): void
+inputRun(InputHandler input): String
loyal.entities::Mob
#name: String
#speed: int
#numSteps = 0: int
#isMoving: boolean
#movingDir = 1: int
#scale = 1: int
#xposition = 1: int
#direction = 0: int
#isSwimming = false: boolean
#tallGrass = false: boolean
-walkingSpeed = 4: int
+Mob(Level level, String name, int x, int y, int speed): ctor
+move(int xa, int ya): void
+hasCollided(int xa, int ya): boolean
#isSolidTile(int xa, int ya, int x, int y): boolean
#isWater(int xa, int ya, int x, int y): boolean
+getName(): String
+tileCheck(): void
loyal.entities::Pointer
-color = Colors.get(-1, -1, -1, 555): int
-scale = 1: int
-time = 0: long
-curTime: long
-prevTime = System.currentTimeMillis(): long
-temp: int
-start: int
-end: int
-game: Loyal
-menu = new ArrayList<String>(): ArrayList<String>
+Pointer(Level level, String name, int x, int y, int jump, int start, int end, Loyal game): ctor
+hasCollided(int xa, int ya): boolean
+tick(): void
+decisionStateChange(): void
+render(Screen screen): void
+inputRun(InputHandler input): String
loyal.Utilityy::Edge
-ID = 0: int
-elem: E
-id: int
-weight: int
-pointers: LinkedList<Vertex<E>
+Edge(): ctor
+Edge(E elem, int distance): ctor
+getId(): int
+getElem(): E
+setElem(E elem): void
+getDistance(): int
+setDistance(int dist): void
+connectTo(Edge<E> other): void
+connectTo(Edge<E> other, int distance): void
+getConnections(): LinkedList<Vertex<E>
+toString(): String
loyal.Utilityy::LinkedList
#head: Node<E>
#size: int
+LinkedList(): ctor
+size(): int
+addFirst(Node<E> v): void
+addLast(Node<E> v): void
+removeFirst(): void
+removeLast(): void
+addAfter(Node<E> u, Node<E> v): void
+addBefore(Node<E> u, Node<E> v): void
+removeAfter(Node<E> v): void
+removeBefore(Node<E> v): void
+remove(Node<E> v): void
+toString(): String
+search(E id): Node<E>
+contains(E id): boolean
loyal.Battle::Battle
-currentBattle: Battle
-playerWin = 1, playerEscape = 2, computerWin = 3: int
-playersTurn = false: boolean
-playerMenu = new BattleMenu(): BattleController
-AI = new BattleAI(): BattleController
-activePlayers, activeEnemies: ArrayList<PlayingCharacter>
-inactivePlayers, inactiveEnemies: ArrayList<PlayingCharacterCounter>
-Battle(ArrayList<PlayingCharacter> activePlayers, ArrayList<PlayingCharacter> activeEnemies): ctor
+startBattle(): int
-startingTurnDecision(): boolean
#checkWinner(): int
-isAlive(ArrayList<PlayingCharacter> characterGroup): int
+setMenu(ArrayList<PlayingCharacter> activePlayers, ArrayList<PlayingCharacter> activeEnemies): void
-checkAliveParty(): void
#getAttack(PlayingCharacter attacker): ArrayList<CharacterAction>
loyal.entities::MapPlayer
-StaminaSize = 2000: int
-color = Colors.get(-1, 200, 020, 321): int
-color2 = Colors.get(-1, 200, 020, 543): int
-scale = 1: int
-tickCount: int
-game: Loyal
-walkingSpeed = 4: int
-stamina = StaminaSize: int
-runCheck = false: boolean
-canRun = true: boolean
+MapPlayer(Level level, int x, int y, InputHandler input, Loyal game): ctor
+hasCollided(int xa, int ya): boolean
+tick(): void
+render(Screen screen): void
+inputRun(InputHandler input): String
loyal.Graphics::Screen
+MAP_WIDTH = 64: int
+MAP_WIDTH_MASK = MAP_WIDTH - 1: int
+BIT_MIRROR_X = 0x01: byte
+BIT_MIRROR_Y = 0x02: byte
+pixels: int[]
+xOffset = 0: int
+yOffset = 0: int
+width: int
+height: int
+sheet: SpriteSheet
+Screen(int width, int height, SpriteSheet sheet): ctor
+render(int xPos, int yPos, int tile, int color, int mirrorDir, int scale): void
+setOffset(int xOffset, int yOffset): void
loyal::InputHandler
+up = new Key(): Key
+down = new Key(): Key
+left = new Key(): Key
+right = new Key(): Key
+enter = new Key(): Key
+escape = new Key(): Key
+shift = new Key(): Key
+InputHandler(Loyal game): ctor
+keyPressed(KeyEvent e): void
+keyReleased(KeyEvent e): void
+keyTyped(KeyEvent e): void
+toggleKey(int keyCode, boolean isPressed): void
loyal.entities::Entity
#x, y: int
#level: Level
+Entity(Level level): ctor
+init(Level level): void
+getX(): int
+getY(): int
+tick(): void
+render(Screen screen): void
+inputRun(InputHandler input): String
+setX(int defaultX): void
+setY(int defaultY): void
loyal.entities::NPCMen
-color = Colors.get(-1, 211, 002, 543): int
-color2 = Colors.get(-1, 211, 411, 543): int
-scale = 1: int
-tickCount: int
-random = new Random(): Random
#move = 5: int
+NPCMen(Level level, int x, int y, int speed): ctor
+hasCollided(int xa, int ya): boolean
+tick(): void
+render(Screen screen): void
+inputRun(InputHandler input): String
loyal::LevelInitializer
+levels = new Level[100]: Level[]
+VILLAGE = new Level(1, "/Levels/HomeVillage.png", Sound.VillageMusic): Level
+OVERWORLD = new Level(2, "/Levels/OverWorldMap.png", Sound.OverWorldMusic): Level
+STARTMENU = new Level(3, "/Levels/test_menu.png", Sound.MenuMusic): Level
+BATTLE = new Level(4, "/Levels/BattleMap.png", Sound.BattleMusic): Level
-game: Loyal
+LevelInitializer(Loyal game): ctor
+initLevelValues(): void
-addMonsters(): void
-startMenuCreation(): void
-villageNPCCreation(): void
loyal::Sound
-clip: Clip
+MenuMusic = new Sound("/music/TheLegend.wav"): Sound
+VillageMusic = new Sound("/music/LittleDroopofPeace.wav"): Sound
+OverWorldMusic = new Sound("/music/Loyalty.wav"): Sound
+VictoryMusic = new Sound("/music/VictoryFanfare.wav"): Sound
+BattleMusic = new Sound("/music/PointZero.wav"): Sound
+Sound(String fileName): ctor
+play(): void
+stop(): void
+loop(): void
+isActive(): boolean
loyal.Utilityy::Vertex
-one, two: Edge<E>
-distance: int
+Vertex(Edge<E> one, Edge<E> two): ctor
+Vertex(Edge<E> one, Edge<E> two, int distance): ctor
+getOne(): Edge<E>
+getTwo(): Edge<E>
+getDistance(): int
+setDistance(int distance): void
+equals(Vertex<E> other): boolean
+toString(): String
+compareTo(Vertex<E> other): int
loyal.Battle::BattleMenu
-temp: Level
-player = null: PlayingCharacter
-attacks = null: ArrayList<CharacterAction>
-target = null: ArrayList<PlayingCharacter>
-attack = null: CharacterAction
+setMenu(Level menu): void
+tick(ArrayList<PlayingCharacter> activePlayers, ArrayList<PlayingCharacter> activeEnemies): void
-getAttack(ArrayList<CharacterAction> attacks): CharacterAction
+updateBattleView(ArrayList<PlayingCharacter> activePlayers, ArrayList<PlayingCharacter> activeEnemies): void
+getIndex(ArrayList<PlayingCharacter> display): int
loyal.Battle.Actions::ClericAbilityFactory
#lightningSpear: LightningSpearOffensiveSingleTarget
#magicBarrier: MagicBarrierDefensiveSingleTargetAbility
#sacredOath: SacredOathMultiTargetDefensiveAbility
#isotingSunlight: SootingSunlightDefensiveSingleTargetAbility
#playingCharacter: PlayingCharacter
+ClericAbilityFactory(): ctor
+createCharacterAction(String name, PlayingCharacter playingCharacter): Ability
+displayAbilitiesBasedOnType(): String[]
+printListOfAbilities(): void
loyal.Battle.Actions::HunterAbilityFactory
#arrowRain: ArrowRainMultiTargetOffensiveAbility
#sharpShot: SharpShotOffensiveSingleTargetAbility
#sharpEye: SharpEyeDefensiveSingleTargetAbility
#poisonousStab: PoisonousStabMultiTargetOffensiveAbility
#playingCharacter: PlayingCharacter
+HunterAbilityFactory(): ctor
+createCharacterAction(String name, PlayingCharacter playingCharacter): Ability
+displayAbilitiesBasedOnType(): String[]
+printListOfAbilities(): void
loyal.Battle.Actions::WizzardAbilityFactory
#darkOrb: DarkOrbOffensiveSingleTargetAbility
#soulArrow: SoulArrowOffensiveSingleTarget
#soulShower: SoulShowerMultiTargetOffensiveAbility
#magicShield: MagicShieldDefensiveSingleTargetAbility
#playingCharacter: PlayingCharacter
+WizzardAbilityFactory(): ctor
+createCharacterAction(String name, PlayingCharacter playingCharacter): Ability
+displayAbilitiesBasedOnType(): String[]
+printListOfAbilities(): void
loyal.Battle.Actions::WorrierAbilityFactory
#airSlash: AirSlashSingleTargetOffenSiveAbility
#backSlash: BackSlashSingleTargetOffenSiveAbility
#taunt: TauntSingleTargetDefenSiveAbility
#desperateMove: DesperateMoveMultiTargetOffensiveAbility
#playingCharacter: PlayingCharacter
+WorrierAbilityFactory(): ctor
+createCharacterAction(String abilityName, PlayingCharacter playingCharacter): Ability
+displayAbilitiesBasedOnType(): String[]
+printListOfAbilities(): void
loyal.level::LevelGenerator
-generator: LevelGenerator
-level: Level
-entities: ArrayList<Entity>
-LevelGenerator(): ctor
+getLevelGenerator(Level level, ArrayList<Entity> entities): LevelGenerator
+getLevel(): Level
+setLevel(Level level): void
+setEntity(ArrayList<Entity> entities): void
+putEntity(Level current): void
loyal.Battle::BattleAI
-temp: Level
-player = null: PlayingCharacter
-attacks = null: ArrayList<CharacterAction>
-target = null: ArrayList<PlayingCharacter>
-attack = null: CharacterAction
+setMenu(Level menu): void
+tick(ArrayList<PlayingCharacter> activePlayers, ArrayList<PlayingCharacter> activeEnemies): void
+updateBattleView(ArrayList<PlayingCharacter> activePlayers, ArrayList<PlayingCharacter> activeEnemies): void
loyal.Battle.Characters::ModifyingItem
#innerState: State
#modifiedStateArray: double[]
-name: String
+ModifyingItem(State innerState): ctor
+getStat(Stat statChoice): double
+setStat(Stat statChoice, double value): void
+getCurrentStatus(): double[]
+undecorate(String name): State
loyal.level::EdgeCollisionDecision
-game: Loyal
-entityWatchedX: int
-entityWatchedY: int
-levelWatched: Level
-input: InputHandler
+update(): void
+setGame(Loyal game): void
+sync(): void
loyal.level::EnterBattle
-game: Loyal
-pointer: Pointer
-battleCharacter: BattleCharacters
~color = Colors.get(111, 222, 333, 444): int
+update(): void
+setGame(Loyal game): void
+sync(): void
-addBattleParticipants(): void
loyal.Utilityy::Node
-value: E
-next: Node<E>
+Node(): ctor
+Node(E value, Node<E> next): ctor
+getValue(): E
+getNext(): Node<E>
+setValue(E value): void
+setNext(Node<E> next): void
loyal.Battle::PlayingCharacterCounter
-cur: PlayingCharacter
-turn: int
+PlayingCharacterCounter(PlayingCharacter cur): ctor
+getCharacter(): PlayingCharacter
+decrementTurn(): void
+getTurn(): int
+compareTo(Object arg0): int
loyal.entities::BattleCharacters
-tickCount = 0: int
-scale = 1: int
~colors: int[]
+BattleCharacters(Level level, int x, int y, int[] colors): ctor
+tick(): void
+render(Screen screen): void
+inputRun(InputHandler input): String
loyal.Utilityy::Graph
-edges: ArrayList<Edge<E»
+Graph(): ctor
+addEdge(Edge<E> vertex): boolean
+contains(Edge<E> vertex): boolean
+get(int index): Edge<E>
+count(): int
+equals(Graph<E> other): boolean
loyal.Battle.Characters::Arms
-value = 2: int
-Heal = Stat.HEALTH, Man = Stat.MANA, Str = Stat.STRENGTH, MPower = Stat.MAGICPOWER, Arm = Stat.ARMOR, MResist = Stat.MAGICRESIST: Stat
+Arms(State innerState): ctor
+unSetModified(): void
+setModified(): void
+setStat(Stat health, double currentHealth): void
loyal.Battle.Characters::Boots
-value = 2: int
-Heal = Stat.HEALTH, Man = Stat.MANA, Str = Stat.STRENGTH, MPower = Stat.MAGICPOWER, Arm = Stat.ARMOR, MResist = Stat.MAGICRESIST: Stat
+Boots(State innerState): ctor
+unSetModified(): void
+setModified(): void
+setStat(Stat health, double currentHealth): void
loyal.Battle.Characters::Bow
-value = 2: int
-Str = Stat.STRENGTH: Stat
+Bow(State innerState): ctor
+unSetModified(): void
+setModified(): void
+setStat(Stat health, double currentHealth): void
loyal.Battle.Characters::CharacterState
-stateArray = { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 }: double[]
+setStat(Stat stat, int value): void
+getStat(Stat defensiveStat): double
+getCurrentStatus(): double[]
+undecorate(String name): State
+setStat(Stat someStat, double value): void
loyal.Battle.Characters::Chest
-value = 6: int
-Heal = Stat.HEALTH, Man = Stat.MANA, Str = Stat.STRENGTH, MPower = Stat.MAGICPOWER, Arm = Stat.ARMOR, MResist = Stat.MAGICRESIST: Stat
+Chest(State innerState): ctor
+unSetModified(): void
+setModified(): void
+setStat(Stat health, double currentHealth): void
loyal.Battle.Characters::Helmet
-value = 5: int
-Heal = Stat.HEALTH, Man = Stat.MANA, Str = Stat.STRENGTH, MPower = Stat.MAGICPOWER, Arm = Stat.ARMOR, MResist = Stat.MAGICRESIST: Stat
+Helmet(State innerState): ctor
+unSetModified(): void
+setModified(): void
+setStat(Stat health, double currentHealth): void
loyal.Battle::Party
~inventory: ArrayList<String>
~players, activePlayers: ArrayList<PlayingCharacter>
+Party(ArrayList<PlayingCharacter> activePlayers): ctor
-addItem(String item): void
-removeItem(String item): void
+countParty(): int
loyal.entities::MenuItems
-words: ArrayList<String>
-start: int
+MenuItems(Level level, ArrayList<String> words, int start): ctor
+tick(): void
+render(Screen screen): void
+inputRun(InputHandler input): String
loyal.level.tiles::AnimatedTile
-animationTileCoords: int[][]
-currentAnimationIndex: int
-lasttiterationTime: long
-animationSwitchDelay: int
+AnimatedTile(int id, int[][] animationCoards, int tileColor, int levelColor, int animationSwitchDelay, int width, int height): ctor
+tick(): void
loyal.level.tiles::BasicTile
#tileId: int
#tileColor: int
~id: int
+BasicTile(int id, int x, int y, int tileColor, int levelColor, int width, int height): ctor
+tick(): void
+render(Screen screen, Level level, int x, int y): void
«interface»
loyal.Battle.Actions::CharacterAction
+castCharacterAction(PlayingCharacter target): void
+getName(): String
+toString(): String
+getAbilityType(): AbilityType
loyal.Battle.Characters::Staff
-value = 2: int
-Str = Stat.STRENGTH, MPower = Stat.MAGICPOWER: Stat
+Staff(State innerState): ctor
+unSetModified(): void
+setModified(): void
«interface»
loyal.Battle.Characters::State
~getStat(Stat health): double
~setStat(Stat health, double currentHealth): void
~getCurrentStatus(): double[]
~undecorate(String name): State
loyal.Battle.Characters::Sword
-value = 3: int
-Str = Stat.STRENGTH: Stat
+Sword(State innerState): ctor
+unSetModified(): void
+setModified(): void
loyal.Battle.Characters::Wand
-value = 2: int
-Str = Stat.STRENGTH, MPower = Stat.MAGICPOWER: Stat
+Wand(State innerState): ctor
+unSetModified(): void
+setModified(): void
loyal.Graphics::SpriteSheet
+path: String
+width: int
+height: int
+pixels: int[]
+SpriteSheet(String path): ctor
«interface»
loyal.Battle::BattleController
+setMenu(Level menu): void
+tick(ArrayList<PlayingCharacter> activePlayers, ArrayList<PlayingCharacter> activeEnemies): void
+updateBattleView(ArrayList<PlayingCharacter> activePlayers, ArrayList<PlayingCharacter> activeEnemies): void
«interface»
loyal.Battle.Characters::StateDecorator
+undecorate(String name): State
~unSetModified(): void
~setModified(): void
«interface»
loyal.level::Decision
~update(): void
~sync(): void
~setGame(Loyal game): void
loyal.level::DecisionFactory
+ENTERBATTLE = new EnterBattle(): EnterBattle
-game: Loyal
+DecisionFactory(Loyal game): ctor
+init(): void
«interface»
loyal.Battle.Actions::AbilityFactory
+createCharacterAction(String name, PlayingCharacter playingCharacter): CharacterAction
+displayAbilitiesBasedOnType(): String[]
loyal.entities::CharacterStore
-factory: SimpleCharacterFactory
+CharacterStore(SimpleCharacterFactory factory): ctor
+orderCharacter(String name, CharacterType type): PlayingCharacter
loyal.Battle.Actions::AirSlashSingleTargetOffenSiveAbility
+AirSlashSingleTargetOffenSiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::ArrowRainMultiTargetOffensiveAbility
+ArrowRainMultiTargetOffensiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::BackSlashSingleTargetOffenSiveAbility
+BackSlashSingleTargetOffenSiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::DarkOrbOffensiveSingleTargetAbility
+DarkOrbOffensiveSingleTargetAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::DefensiveAbility
+DefensiveAbility(PlayingCharacter playingCharacter): ctor
#canCastOn(PlayingCharacter target): boolean
loyal.Battle.Actions::DeffensiveAbility
+DeffensiveAbility(PlayingCharacter playingCharacter): ctor
#canCastOn(PlayingCharacter target): boolean
loyal.Battle.Actions::DesperateMoveMultiTargetOffensiveAbility
+DesperateMoveMultiTargetOffensiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::LightningSpearOffensiveSingleTarget
+LightningSpearOffensiveSingleTarget(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::MagicBarrierDefensiveSingleTargetAbility
+MagicBarrierDefensiveSingleTargetAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::MagicShieldDefensiveSingleTargetAbility
+MagicShieldDefensiveSingleTargetAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::MultiTargetDefensiveAbility
+MultiTargetDefensiveAbility(PlayingCharacter playingCharacter): ctor
#getTargets(PlayingCharacter target): ArrayList<PlayingCharacter>
loyal.Battle.Actions::MultiTargetDeffensiveAbility
+MultiTargetDeffensiveAbility(PlayingCharacter playingCharacter): ctor
#getTargets(PlayingCharacter target): ArrayList<PlayingCharacter>
loyal.Battle.Actions::MultiTargetOffensiveAbility
+MultiTargetOffensiveAbility(PlayingCharacter playingCharacter): ctor
#getTargets(PlayingCharacter target): ArrayList<PlayingCharacter>
loyal.Battle.Actions::OffensiveAbility
+OffensiveAbility(PlayingCharacter playingCharacter): ctor
#canCastOn(PlayingCharacter target): boolean
loyal.Battle.Actions::PoisonousStabMultiTargetOffensiveAbility
+PoisonousStabMultiTargetOffensiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::SacredOathMultiTargetDefensiveAbility
+SacredOathMultiTargetDefensiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::SharpEyeDefensiveSingleTargetAbility
+SharpEyeDefensiveSingleTargetAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::SharpShotOffensiveSingleTargetAbility
+SharpShotOffensiveSingleTargetAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::SingleTargetDefensiveAbility
+SingleTargetDefensiveAbility(PlayingCharacter playingCharacter): ctor
#getTargets(PlayingCharacter target): ArrayList<PlayingCharacter>
loyal.Battle.Actions::SingleTargetOffensiveAbility
+SingleTargetOffensiveAbility(PlayingCharacter playingCharacter): ctor
#getTargets(PlayingCharacter target): ArrayList<PlayingCharacter>
loyal.Battle.Actions::SootingSunlightDefensiveSingleTargetAbility
+SootingSunlightDefensiveSingleTargetAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::SoulArrowOffensiveSingleTarget
+SoulArrowOffensiveSingleTarget(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::SoulShowerMultiTargetOffensiveAbility
+SoulShowerMultiTargetOffensiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::TauntSingleTargetDefenSiveAbility
+TauntSingleTargetDefenSiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
loyal.Battle.Actions::TauntSingleTargetDeffenSiveAbility
+TauntSingleTargetDeffenSiveAbility(PlayingCharacter playingCharacter): ctor
#applyAbility(ArrayList<PlayingCharacter> targets): void
«interface»
loyal.Battle.Characters::Interactable
+Interaction(PlayingCharacter sources, PlayingCharacter[] targets): void
loyal.Graphics::Colors
+get(int color1, int color2, int color3, int color4): int
-get(int color): int
loyal.Graphics::Font
-chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ " + "0123456789...;:\'!?$%()-=+/ ^: String
+render(String msg, Screen screen, int x, int y, int color, int scale): void
loyal.Battle.Actions::Tester
+main(String[] args): void
loyal.entities::Cleric
+Cleric(String name, CharacterType type): ctor
loyal.entities::Hunter
+Hunter(String name, CharacterType type): ctor
loyal.entities::SimpleCharacterFactory
+createCharacter(String name, CharacterType type): PlayingCharacter
loyal.entities::Warrior
+Warrior(String name, CharacterType type): ctor
loyal.entities::Wizzard
+Wizzard(String name, CharacterType type): ctor
loyal.level.tiles::BasicSolidTile
+BasicSolidTile(int id, int x, int y, int tileColor, int levelColor, int width, int height): ctor