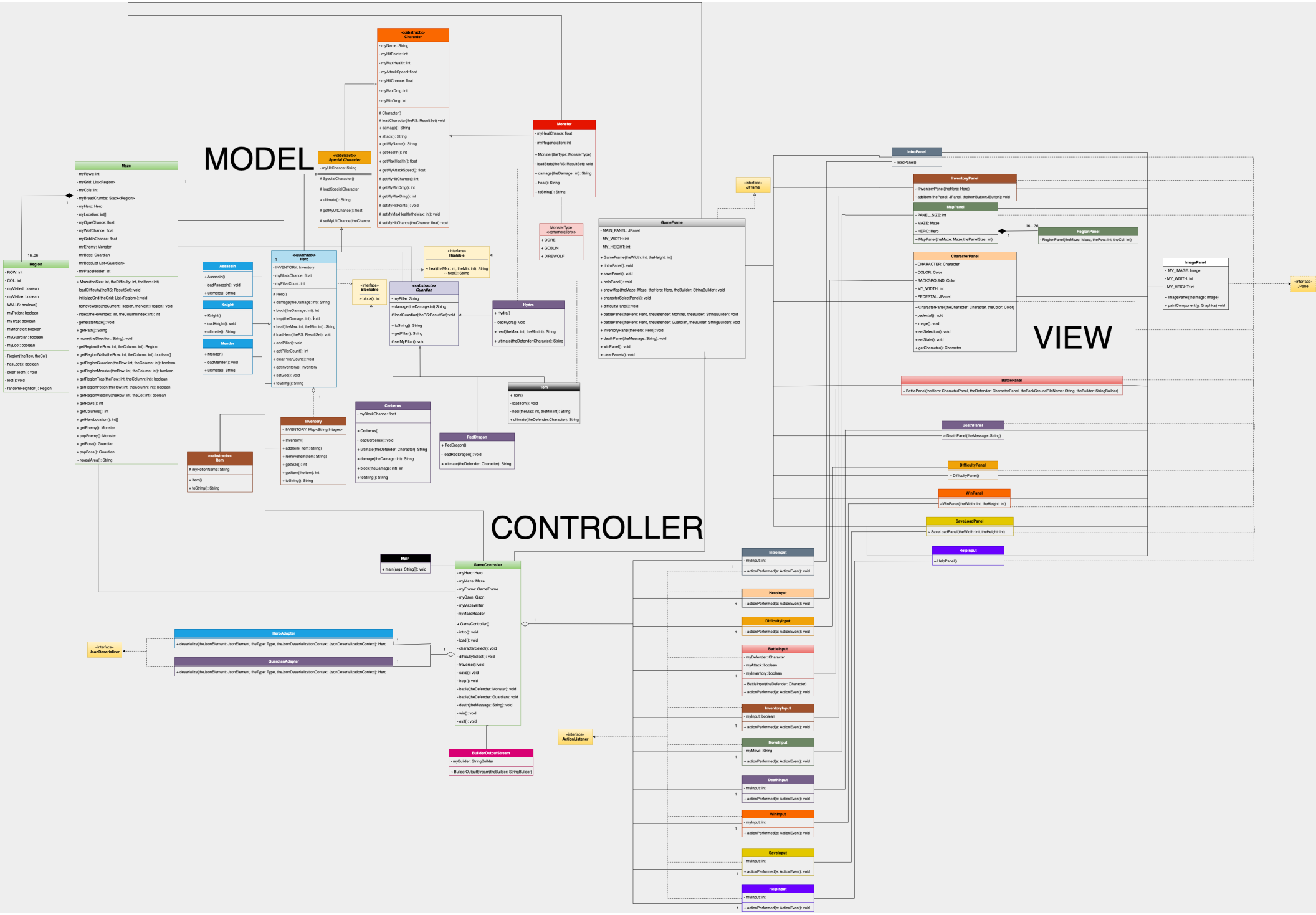
MODEL
VIEW
CONTROLLER
Maze
- myRows: int
- myGrid: List<Region>
- myCols: int
- myBreadCrumbs: Stack<Region>
- myHero: Hero
- myLocation: int[]
- myOgreChance: float
- myWolfChance: float
- myGoblinChance: float
- myEnemy: Monster
- myBoss: Guardian
- myBossList: List<Guardian>
- myPlaceHolder: int
+ Maze(theSize: int, theDifficulty: int, theHero: int)
- loadDifficulty(theRS: ResultSet): void
- initializeGrid(theGrid: List<Region>): void
- removeWalls(theCurrent: Region, theNext: Region): void
- index(theRowIndex: int, theColumnIndex: int): int
- generateMaze(): void
+ getPath(): String
+ move(theDirection: String): void
- getRegion(theRow: int, theColumn: int): Region
+ getRegionWalls(theRow: int, theColumn: int): boolean[]
+ getRegionGuardian(theRow: int, theColumn: int): boolean
+ getRegionMonster(theRow: int, theColumn: int): boolean
+ getRegionTrap(theRow: int, theColumn: int): boolean
+ getRegionPotion(theRow: int, theColumn: int): boolean
+ getRegionVisibility(theRow: int, theCol: int): boolean
+ getRows(): int
+ getColumns(): int
+ getHeroLocation(): int[]
+ getEnemy(): Monster
+ popEnemy(): Monster
+ getBoss(): Guardian
+ popBoss(): Guardian
~ revealArea(): String
1
1
16..36
Region
- ROW: int
- COL: int
- myVisited: boolean
- myVisible: boolean
- WALLS: boolean[]
- myPotion: boolean
- myTrap: boolean
- myMonster: boolean
- myGuardian: boolean
- myLoot: boolean
- Region(theRow, theCol)
- hasLoot(): boolean
- clearRoom(): void
- loot(): void
- randomNeighbor(): Region
<<abstract>>
Special Character
- myUltChance: String
SpecialCharacter()
loadSpecialCharacter
+ ultimate(): String
# getMyUltChance(): float
# setMyUltChance(theChance
<<abstract>>
Character
- myName: String
- myHitPoints: int
- myMaxHealth: int
- myAttackSpeed: float
- myHitChance: float
- myMaxDmg: int
- myMinDmg: int
Character()
loadCharacter(theRS: ResultSet) void
+ damage(): String
+ attack(): String
+ getMyName(): String
+ getHealth(): int
+ getMaxHealth(): float
+ getMyAttackSpeed(): float
+ getMyHitChance(): int
+ getMyMinDmg(): int
+ getMyMaxDmg(): int
+ setMyHitPoints(): void
# setMyMaxHealth(theMax: int): void
# setMyHitChance(theChance: float): void
Monster
- myHealChance: float
- myRegeneration: int
+ Monster(theType: MonsterType)
- loadStats(theRS: ResultSet): void
+ damage(theDamage: int): String
+ heal(): String
+ toString(): String
MonsterType
<<enumeration>>
+ OGRE
+ GOBLIN
+ DIREWOLF
+interface+
Healable
~ heal(theMax: int, theMin: int): String
~ heal(): String
+interface+
Blockable
~ block(): int
<<abstract>>
Hero
1
- INVENTORY: Inventory
- myBlockChance: float
- myPillarCount: int
Hero()
+ damage(theDamage: int): String
+ block(theDamage: int): int
+ trap(theDamage: int): void
+ heal(theMax: int, theMin: int): String
# loadHero(theRS: ResultSet): void
+ addPillar(): void
+ getPillarCount(): int
+ clearPillarCount(): void
+ getInventory(): Inventory
+ setGod(): void
+ toString(): String
Assassin
+ Assassin()
- loadAssassin(): void
+ ultimate(): String
Knight
+ Knight()
- loadKnight(): void
+ ultimate(): String
Mender
+ Mender()
- loadMender(): void
+ ultimate(): String
<<abstract>>
Guardian
- myPillar: String
+ damage(theDamage:int):String
# loadGuardian(theRS:ResultSet):void
+ toString(): String
+ getPillar(): String
# setMyPillar(): void
Hydra
+ Hydra()
- loadHydra(): void
+ heal(theMax: int, theMin:int): String
+ ultimate(theDefender:Character): String
Tom
+ Tom()
- loadTom(): void
- heal(theMax: int, theMin:int): String
+ ultimate(theDefender:Character): String
Cerberus
- myBlockChance: float
+ Cerberus()
- loadCerberus(): void
+ ultimate(theDefender: Character): String
+ damage(theDamage: int): String
+ block(theDamage: int): int
+ toString(): String
RedDragon
+ RedDragon()
- loadRedDragon(): void
+ ultimate(theDefender: Character): String
<<abstract>>
Item
myPotionName: String
+ Item()
+ toString(): String
Inventory
- INVENTORY: Map<String,Integer>
+ Inventory()
+ addItem(Item: String)
+ removeItem(Item: String)
+ getSize(): int
+ getItem(theItem): int
+ toString(): String
GameFrame
- MAIN_PANEL: JPanel
- MY_WIDTH: int
- MY_HEIGHT: int
+ GameFrame(theWidth: int, theHeight: int)
+ introPanel(): void
+ savePanel(): void
+ helpPanel(): void
+ showMap(theMaze: Maze, theHero: Hero, theBuilder: StringBuilder): void
+ characterSelectPanel(): void
+ difficultyPanel(): void
+ battlePanel(theHero: Hero, theDefender: Monster, theBuilder: StringBuilder): void
+ battlePanel(theHero: Hero, theDefender: Guardian, theBuilder: StringBuilder): void
+ inventoryPanel(theHero: Hero): void
+ deathPanel(theMessage: String): void
+ winPanel(): void
+ clearPanels(): void
+interface+
JFrame
IntroPanel
~ IntroPanel()
InventoryPanel
~ InventoryPanel(theHero: Hero)
- addItem(thePanel: JPanel, theItemButton:JButton): void
MapPanel
- PANEL_SIZE: int
- MAZE: Maze
- HERO: Hero
~ MapPanel(theMaze: Maze,thePanelSize: int)
16 .. 36
1
RegionPanel
- RegionPanel(theMaze: Maze, theRow: int, theCol: int)
CharacterPanel
- CHARACTER: Character
- COLOR: Color
- BACKGROUND: Color
- MY_WIDTH: int
- PEDESTAL: JPanel
~ CharacterPanel(theCharacter: Character, theColor: Color)
- pedestal(): void
- image(): void
+ setSelection(): void
+ setStats(): void
+ getCharacter(): Character
ImagePanel
- MY_IMAGE: Image
- MY_WIDTH: int
- MY_HEIGHT: int
~ ImagePanel(theImage: Image)
+ paintComponent(g: Graphics) void
+interface+
JPanel
BattlePanel
~ BattlePanel(theHero: CharacterPanel, theDefender: CharacterPanel, theBackGroundFileName: String, theBuilder: StringBuilder)
DeathPanel
~ DeathPanel(theMessage: String)
DifficultyPanel
~ DifficultyPanel()
WinPanel
~WinPanel(theWidth: int, theHeight: int)
SaveLoadPanel
~ SaveLoadPanel(theWidth: int, theHeight: int)
HelpInput
~ HelpPanel()
Main
+ main(args: String[]): void
GameController
- myHero: Hero
- myMaze: Maze
- myFrame: GameFrame
- myGson: Gson
- myMazeWriter
-myMazeReader
+ GameController()
- intro(): void
- load(): void
- characterSelect(): void
- difficultySelect(): void
- traverse(): void
- save(): void
- help(): void
- battle(theDefender: Monster): void
- battle(theDefender: Guardian): void
- death(theMessage: String): void
- win(): void
- exit(): void
1
BuilderOutputStream
- myBuilder: StringBuilder
~ BuilderOutputStream(theBuilder: StringBuilder)
+interface+
JsonDeserializer
HeroAdapter
+ deserialize(theJsonElement: JsonElement, theType: Type, theJsonDeserializationContext: JsonDeserializationContext): Hero
GuardianAdapter
+ deserialize(theJsonElement: JsonElement, theType: Type, theJsonDeserializationContext: JsonDeserializationContext): Hero
1
1
1
+interface+
ActionListener
IntroInput
- myInput: int
+ actionPerformed(e: ActionEvent): void
HeroInput
+ actionPerformed(e: ActionEvent): void
DifficultyInput
+ actionPerformed(e: ActionEvent): void
BattleInput
- myDefender: Character
- myAttack: boolean
- myInventory: boolean
+ BattleInput(theDefender: Character)
+ actionPerformed(e: ActionEvent): void
InventoryInput
- myInput: boolean
+ actionPerformed(e: ActionEvent): void
MoveInput
- myMove: String
+ actionPerformed(e: ActionEvent): void
DeathInput
- myInput: int
+ actionPerformed(e: ActionEvent): void
WinInput
- myInput: int
+ actionPerformed(e: ActionEvent): void
SaveInput
- myInput: int
+ actionPerformed(e: ActionEvent): void
HelpInput
- myInput: int
+ actionPerformed(e: ActionEvent): void