

Design Class Diagram
Team GulDukat

```

classDiagram
    class Preset {
        +numberOfCharacters : int
        +areCharsPlayablePreset : bool
        +defaultProfessionPreset : PROFESSION
        +defaultProfessionLevelPreset : int
        +defaultReviveOptionPreset : HEAL_OPTION
        +defaultCdPreset : int
        +defaultStartingDifficultyPreset : DIFFICULTY
        +defaultStartingLevelPreset : int
        +defaultMaxLevelPreset : int
        +defaultEnvironmentPreset : ENVIRONMENT
        +defaultRepeatScenarioPreset : int
    }
    class LoginController {
        +verifyUser(username : string, password : string) : User
    }
    class LoginGUI {
        +username : string
        +password : string
        +loginStatus : string
    }
    class User {
        -username : string
        -password : string
        -isAdmin : bool
    }
    class ConfigGUI {
        -areNumericsValid() : bool
        +areFieldsEmpty() : bool
        -initSimulationSettings() : void
        -mapScenarioSettings() : void
        -mapSceneSettings() : void
    }
    class ConfigController {
        -configGUI : ConfigGUI
        -simController : SimulationController
        -scenarioSettings : ScenarioSettings
        -sceneSettings : SceneSettings
        -characters : List<Character>
        -monsters : List<Monster>
        -user : User
        +initSimulation(monsterTypes : List<string>) : void
        +initializeMonsters(monsterTypes : List<string>) : int
        -autoFillMonstersToMatchCD( curDL : int) : void
    }
    class SimulatorGUI {
        -scenarioSettings : ScenarioSettings
        -sceneSettings : SceneSettings
    }
    class SimulatorController {
        -scenarioSettings : ScenarioSettings
        -sceneSettings : SceneSettings
        -combatController : CombatRoundController
        -config : ConfigController
        -characters : List<Character>
        -monsters : List<Monster>
        -curLevel : int
        -runNum : int
        +diceRoll(numRolls : int, sides : int) : int
        +checkForDeath() : void
        +automateMove()
        +meleeAttack()
        +spellAttack()
    }
    class ScenarioSettings {
        +difficulty : DIFFICULTY
        +startLevel : int
        +endLevel : int
        +numberOfRuns : int
        +numberOfCharacters : int
        +initCD : int
    }
    class SceneSettings {
        +environment : ENVIRONMENT
        +level : int
        +challengeDifficulty : int
        +monsters : List<Monster>
        +characters : List<Character>
    }
    class Report {
        -scenario : Scenario
        -scene : Scene
        -totalNumOfChars : int
        -totalNumOfMonsters : int
        -totalCD : int
        -numOfCharsDefeated : int
        -numOfMonstersDefeated : int
        -totalDamageTaken : int
        -totalDamageGiven : int
        -numOfLevelsProgressed : int
        -numOfTimesSimRan : int
        -index : int
        -simulationDifficulty : string
        -environmentOfBattle : string
        -treasure : int
    }
    class ReportController {
        -report : Report
        -reports : List<Report>
        +initializeReport() : void
        +addReport() : void
    }
    class ReportGUI {
        +populateReportsListBox() : void
        +populateReportStatsBreakDown(report : Report) : void
    }
    class ModifyAccountController {
        +updateUserUsername(user : User, username : string)
        +updateUserPassword(user : User, password : string)
    }
    class Character {
        -profession : PROFESSION
        -healOption : HEAL_OPT
        -level : int
        -maxHealth : int
        -curHealth : int
        -strength : int
        -intelligence : int
        -wisdom : int
        -dexterity : int
        -constitution : int
        -playable : bool
        -initiative : int
    }
    class Monster {
        -maxHealth : int
        -curHealth : int
        -type : TYPE
        -name : VARIANT
        -numAttacks : int
        -damage : int
        -difficultyLevel : int
        -initiative : int
    }
    class CombatRoundController {
        -scenario : ScenarioSettings
        -scene : SceneSettings
        -characters : List<Character>
        -monsters : List<Monster>
        +meleeAttack(character : int, monster : int) : void
        +spellAttack(character : int, monster : int) : void
        +monsterAttack(character : int, monster : int) : void
        -getPhysAttackDef( monster : int) : void
        -getMagicAttackDef(monster : int) : void
    }
    Preset --> DBManager
    LoginController --> DBManager
    LoginGUI --> User
    User --> ConfigController
    User --> DBManager
    ConfigGUI --> ConfigController
    ConfigController --> SimulatorController
    SimulatorGUI --> SimulatorController
    ScenarioSettings --> SimulatorController
    ScenarioSettings --> SceneSettings
    SceneSettings --> SimulatorController
    Report --> ReportController
    ReportController --> ReportGUI
    ModifyAccountController --> DBManager
    Character --> SimulatorController
    Character --> Monster
    Monster --> CombatRoundController
    
```

The diagram illustrates the architecture of a game system. It features several key classes and their interactions:

- Preset**: Manages game presets like character counts, professions, and starting levels.
- LoginController** and **LoginGUI**: Handle user authentication. **LoginGUI** provides input fields for username and password, while **LoginController** verifies them.
- User**: Represents a game user with attributes like username, password, and admin status.
- ConfigGUI** and **ConfigController**: Manage simulation settings. **ConfigGUI** provides input for settings, and **ConfigController** initializes the simulation and monsters.
- SimulatorGUI** and **SimulatorController**: Manage the simulation. **SimulatorGUI** provides input for scenario and scene settings, while **SimulatorController** handles the simulation logic, including dice rolls and attacks.
- ScenarioSettings** and **SceneSettings**: Define game parameters like difficulty, level, and monster types.
- Report** and **ReportController**: Manage game reports. **ReportController** initializes and adds reports, while **ReportGUI** displays them.
- ModifyAccountController**: Handles user account updates like username and password.
- Character** and **Monster**: Represent game entities with attributes like health, strength, and intelligence.
- CombatRoundController**: Manages combat rounds, including melee and spell attacks.

