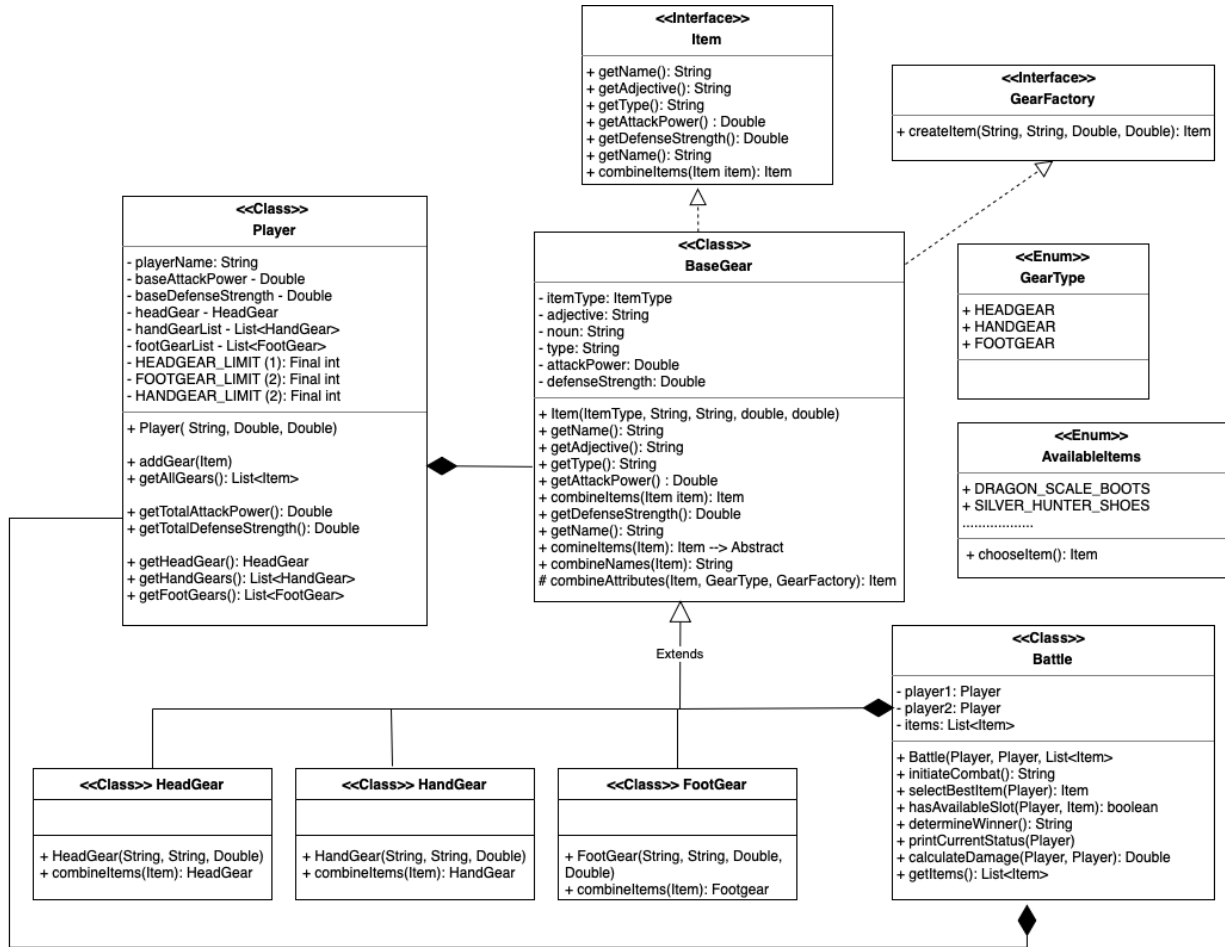# LAB 2 – ROLE-PLAYING GAMES

## CS5010 – Programming Design Paradigm

UML Class diagram and testing plan for Part - 1

Kiruthiga Muralidharan

# UML Class Diagram:

**<<Interface>>**
**Item**

+ getName(): String
+ getAdjective(): String
+ getType(): String
+ getAttackPower() : Double
+ getDefenseStrength(): Double
+ getName(): String
+ combineItems(Item item): Item

**<<Interface>>**
**GearFactory**

+ createItem(String, String, Double, Double): Item

**<<Class>>**
**Player**

- playerName: String
- baseAttackPower - Double
- baseDefenseStrength - Double
- headGear - HeadGear
- handGearList - List<HandGear>
- footGearList - List<FootGear>
- HEADGEAR_LIMIT (1): Final int
- FOOTGEAR_LIMIT (2): Final int
- HANDGEAR_LIMIT (2): Final int

+ Player( String, Double, Double)

+ addGear(Item)
+ getAllGears(): List<Item>

+ getTotalAttackPower(): Double
+ getTotalDefenseStrength(): Double

+ getHeadGear(): HeadGear
+ getHandGears(): List<HandGear>
+ getFootGears(): List<FootGear>

**<<Class>>**
**BaseGear**

- itemType: ItemType
- adjective: String
- noun: String
- type: String
- attackPower: Double
- defenseStrength: Double

+ Item(ItemType, String, String, double, double)
+ getName(): String
+ getAdjective(): String
+ getType(): String
+ getAttackPower() : Double
+ combineItems(Item item): Item
+ getDefenseStrength(): Double
+ getName(): String
+ comineItems(Item): Item --> Abstract
+ combineNames(Item): String
# combineAttributes(Item, GearType, GearFactory): Item

**<<Enum>>**
**GearType**

+ HEADGEAR
+ HANDGEAR
+ FOOTGEAR

**<<Enum>>**
**AvailableItems**

+ DRAGON_SCALE_BOOTS
+ SILVER_HUNTER_SHOES
..................

+ chooseItem(): Item

Extends

**<<Class>> HeadGear**

+ HeadGear(String, String, Double)
+ combineItems(Item): HeadGear

**<<Class>> HandGear**

+ HandGear(String, String, Double)
+ combineItems(Item): HandGear

**<<Class>> FootGear**

+ FootGear(String, String, Double, Double)
+ combineItems(Item): Footgear

**<<Class>>**
**Battle**

- player1: Player
- player2: Player
- items: List<Item>

+ Battle(Player, Player, List<Item>
+ initiateCombat(): String
+ selectBestItem(Player): Item
+ hasAvailableSlot(Player, Item): boolean
+ determineWinner(): String
+ printCurrentStatus(Player)
+ calculateDamage(Player, Player): Double
+ getItems(): List<Item>

## Test Plan:

| Test Method | Description | Input Assignment | Expected Output |
|---|---|---|---|
| testAddGearsOfSameType() | Test combining gears correctly | Two gears of same type | Defense and Attack should be adjusted, total name is created - as per the requirement |
| testAddDifferentGearTypes() | Tests combining gears incorrectly | Two gears of different type | IllegalArgumentException |
| createGearsWithNullOrEmpty() | Tests incorrect creation of gear | Null adjective / type | IllegalArgumentException |
| testGearAssignmentToPlayer() | Tests the gear addition to the player | Success case | Player1 is assigned to the specified gear |
| testCreatePlayerWithNullArgument() / Empty String | Tests incorrect creation of player | Null player name | IllegalArgumentException |
| testAddExcessHeadGear() / HandGears / FootGears | Tests gear limit | Add gears more than the limit | IllegalStateException |
| testPlayerToString() | Tests the details of the player | Get the player details | Prints the details of the player. |