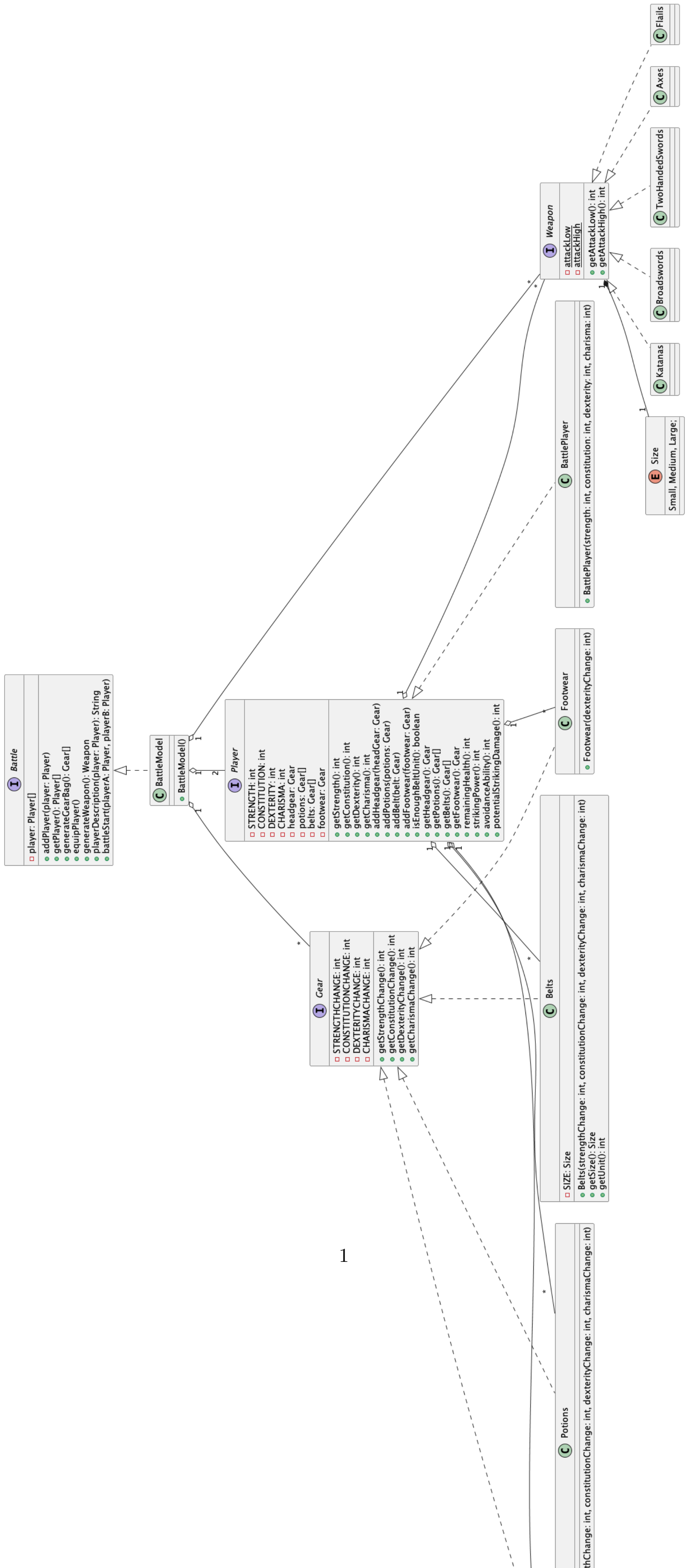
1. Original Design



2. BattlePlayer

Method	Input	Expected
Constructor with invalid values	BattlePlayer(0, 0, 0, 0)	IllegalArgumentException
Constructor with valid values	BattlePlayer(2, 2, 2, 2)	
test remainHealth	remainHealth()	
test strikingPower	strikingPower()	
test avoidanceAbility	avoidanceAbility()	
test potentialStrikingDamage	potentialStrikingDamage()	

3. Headgear

Method	Input	Expected
Constructor with valid values	Headgear(2)	

4. Potions

Method	Input	Expected
Constructor with invalid values	Potions(2, 2, 2, 2)	IllegalArgumentException
Constructor with valid values	Potions(2, 0, 0,0)	

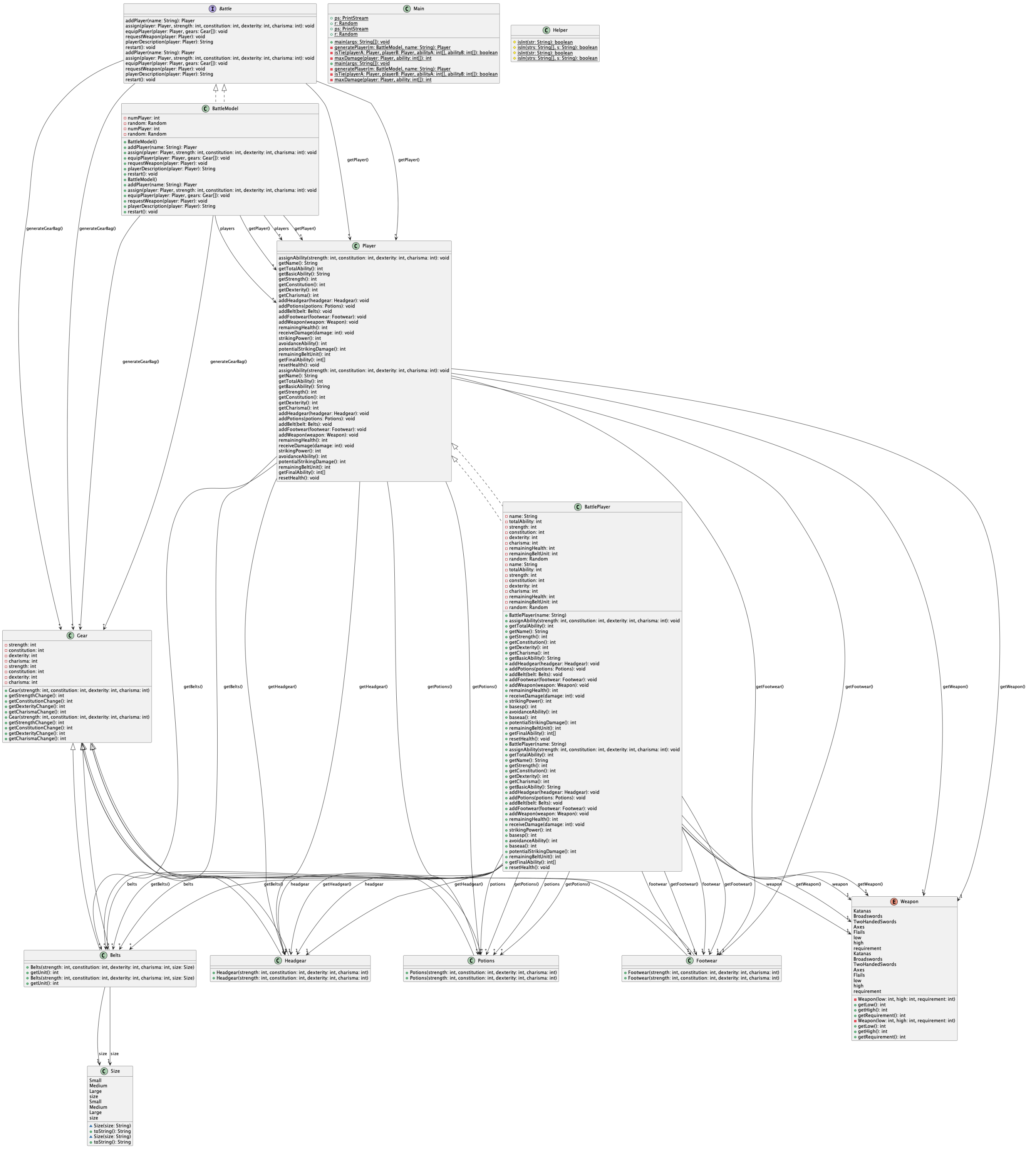
5. Footwear

Method	Input	Expected
Constructor with valid values	Footwear(2)	

6. BattleModel

Method	Input	Expected
Constructor	BattleModel()	

7. Final Design


Battle
addPlayer(name: String): Player
assign(player: Player, strength: int, constitution: int, dexterity: int, charisma: int): void
equipPlayer(player: Player, gears: Gear[]): void
requestWeapon(player: Player): void
playerDescription(player: Player): String
restart(): void
addPlayer(name: String): Player
assign(player: Player, strength: int, constitution: int, dexterity: int, charisma: int): void
equipPlayer(player: Player, gears: Gear[]): void
requestWeapon(player: Player): void
playerDescription(player: Player): String
restart(): void
Main
ps: PrintStream
r: Random
ps: PrintStream
r: Random
main(args: String[]): void
generatePlayer(m: BattleModel, name: String): Player
isTie(playerA: Player, playerB: Player, abilityA: int[], abilityB: int[]): boolean
maxDamage(player: Player, ability: int[]): int
main(args: String[]): void
generatePlayer(m: BattleModel, name: String): Player
isTie(playerA: Player, playerB: Player, abilityA: int[], abilityB: int[]): boolean
maxDamage(player: Player, ability: int[]): int
Helper
isInt(str: String): boolean
isIn(strs: String[], s: String): boolean
isInt(str: String): boolean
isIn(strs: String[], s: String): boolean
BattleModel
numPlayer: int
random: Random
numPlayer: int
random: Random
BattleModel()
addPlayer(name: String): Player
assign(player: Player, strength: int, constitution: int, dexterity: int, charisma: int): void
equipPlayer(player: Player, gears: Gear[]): void
requestWeapon(player: Player): void
playerDescription(player: Player): String
restart(): void
BattleModel()
addPlayer(name: String): Player
assign(player: Player, strength: int, constitution: int, dexterity: int, charisma: int): void
equipPlayer(player: Player, gears: Gear[]): void
requestWeapon(player: Player): void
playerDescription(player: Player): String
restart(): void
getPlayer()
getPlayer()
generateGearBag()
generateGearBag()
players
getPlayer()
players
getPlayer()
Player
assignAbility(strength: int, constitution: int, dexterity: int, charisma: int): void
getName(): String
getTotalAbility(): int
getBasicAbility(): String
getStrength(): int
getConstitution(): int
getDexterity(): int
getCharisma(): int
addHeadgear(headgear: Headgear): void
addPotions(potions: Potions): void
addBelt(belt: Belts): void
addFootwear(footwear: Footwear): void
addWeapon(weapon: Weapon): void
remainingHealth(): int
receiveDamage(damage: int): void
strikingPower(): int
avoidanceAbility(): int
potentialStrikingDamage(): int
remainingBeltUnit(): int
getFinalAbility(): int[]
resetHealth(): void
assignAbility(strength: int, constitution: int, dexterity: int, charisma: int): void
getName(): String
getTotalAbility(): int
getBasicAbility(): String
getStrength(): int
getConstitution(): int
getDexterity(): int
getCharisma(): int
addHeadgear(headgear: Headgear): void
addPotions(potions: Potions): void
addBelt(belt: Belts): void
addFootwear(footwear: Footwear): void
addWeapon(weapon: Weapon): void
remainingHealth(): int
receiveDamage(damage: int): void
strikingPower(): int
avoidanceAbility(): int
potentialStrikingDamage(): int
remainingBeltUnit(): int
getFinalAbility(): int[]
resetHealth(): void
generateGearBag()
generateGearBag()
BattlePlayer
name: String
totalAbility: int
strength: int
constitution: int
dexterity: int
charisma: int
remainingHealth: int
remainingBeltUnit: int
random: Random
name: String
totalAbility: int
strength: int
constitution: int
dexterity: int
charisma: int
remainingHealth: int
remainingBeltUnit: int
random: Random
BattlePlayer(name: String)
assignAbility(strength: int, constitution: int, dexterity: int, charisma: int): void
getTotalAbility(): int
getName(): String
getStrength(): int
getConstitution(): int
getDexterity(): int
getCharisma(): int
getBasicAbility(): String
addHeadgear(headgear: Headgear): void
addPotions(potions: Potions): void
addBelt(belt: Belts): void
addFootwear(footwear: Footwear): void
addWeapon(weapon: Weapon): void
remainingHealth(): int
receiveDamage(damage: int): void
strikingPower(): int
basesp(): int
avoidanceAbility(): int
baseaa(): int
potentialStrikingDamage(): int
remainingBeltUnit(): int
getFinalAbility(): int[]
resetHealth(): void
BattlePlayer(name: String)
assignAbility(strength: int, constitution: int, dexterity: int, charisma: int): void
getTotalAbility(): int
getName(): String
getStrength(): int
getConstitution(): int
getDexterity(): int
getCharisma(): int
getBasicAbility(): String
addHeadgear(headgear: Headgear): void
addPotions(potions: Potions): void
addBelt(belt: Belts): void
addFootwear(footwear: Footwear): void
addWeapon(weapon: Weapon): void
remainingHealth(): int
receiveDamage(damage: int): void
strikingPower(): int
basesp(): int
avoidanceAbility(): int
baseaa(): int
potentialStrikingDamage(): int
remainingBeltUnit(): int
getFinalAbility(): int[]
resetHealth(): void
Gear
strength: int
constitution: int
dexterity: int
charisma: int
strength: int
constitution: int
dexterity: int
charisma: int
Gear(strength: int, constitution: int, dexterity: int, charisma: int)
getStrengthChange(): int
getConstitutionChange(): int
getDexterityChange(): int
getCharismaChange(): int
Gear(strength: int, constitution: int, dexterity: int, charisma: int)
getStrengthChange(): int
getConstitutionChange(): int
getDexterityChange(): int
getCharismaChange(): int
getBelts()
getBelts()
getHeadgear()
getHeadgear()
getPotions()
getPotions()
getFootwear()
getFootwear()
getWeapon()
getWeapon()
belts
getBelts()
belts
getBelts()
headgear
getHeadgear()
headgear
getHeadgear()
potions
getPotions()
potions
getPotions()
footwear
getFootwear()
footwear
getFootwear()
weapon
getWeapon()
weapon
getWeapon()
Belts
Belts(strength: int, constitution: int, dexterity: int, charisma: int, size: Size)
getUnit(): int
Belts(strength: int, constitution: int, dexterity: int, charisma: int, size: Size)
getUnit(): int
Headgear
Headgear(strength: int, constitution: int, dexterity: int, charisma: int)
Headgear(strength: int, constitution: int, dexterity: int, charisma: int)
Potions
Potions(strength: int, constitution: int, dexterity: int, charisma: int)
Potions(strength: int, constitution: int, dexterity: int, charisma: int)
Footwear
Footwear(strength: int, constitution: int, dexterity: int, charisma: int)
Footwear(strength: int, constitution: int, dexterity: int, charisma: int)
Weapon
Katanas
Broadswords
TwoHandedSwords
Axes
Flails
low
high
requirement
Katanas
Broadswords
TwoHandedSwords
Axes
Flails
low
high
requirement
Weapon(low: int, high: int, requirement: int)
getLow(): int
getHigh(): int
getRequirement(): int
Weapon(low: int, high: int, requirement: int)
getLow(): int
getHigh(): int
getRequirement(): int
size
size
Size
Small
Medium
Large
size
Small
Medium
Large
size
Size(size: String)
toString(): String
Size(size: String)
toString(): String

8. GearTest

Method	Input	Expected
Constructor of a gear	strength of the gear	equal
Constructor of a gear	constitution of the gear	equal
Constructor of a gear	dexterity of the gear	equal
Constructor of a gear	charisma of the gear	equal
Constructor of a gear	size of the belt	equal

9. WeaponTest

Method	Input	Expected
Constructor of a weapon	low attack range of the weapon	equal
Constructor of a weapon	high attack range of the weapon	equal

10. BattleTest

Method	Input
Non-positive ability of a player	0 charisma
Adding gear doesn't change base ability	base constitution and constitution after adding headg
Correct total health	assertEquals(total, player.remainingHealth())
Correct striking power	assertEquals(total - 3, ((BattlePlayer) player).basesp
Correct avoidance	assertEquals(1, ((BattlePlayer) player).baseaa())
Correct potential damage	assertEquals(total - 3, player.potentialStrikingDamag
Negative damage change nothing	do -1 damage and check if change remain health
Correct updating health after damage	do 1 damage and check
Test entering arena with bare hand	check no gear and no weapon
Test one headgear	add two headgear
Test one footwear	add two footwear
Test belt unit	add 2 large belt and 1 medium belt