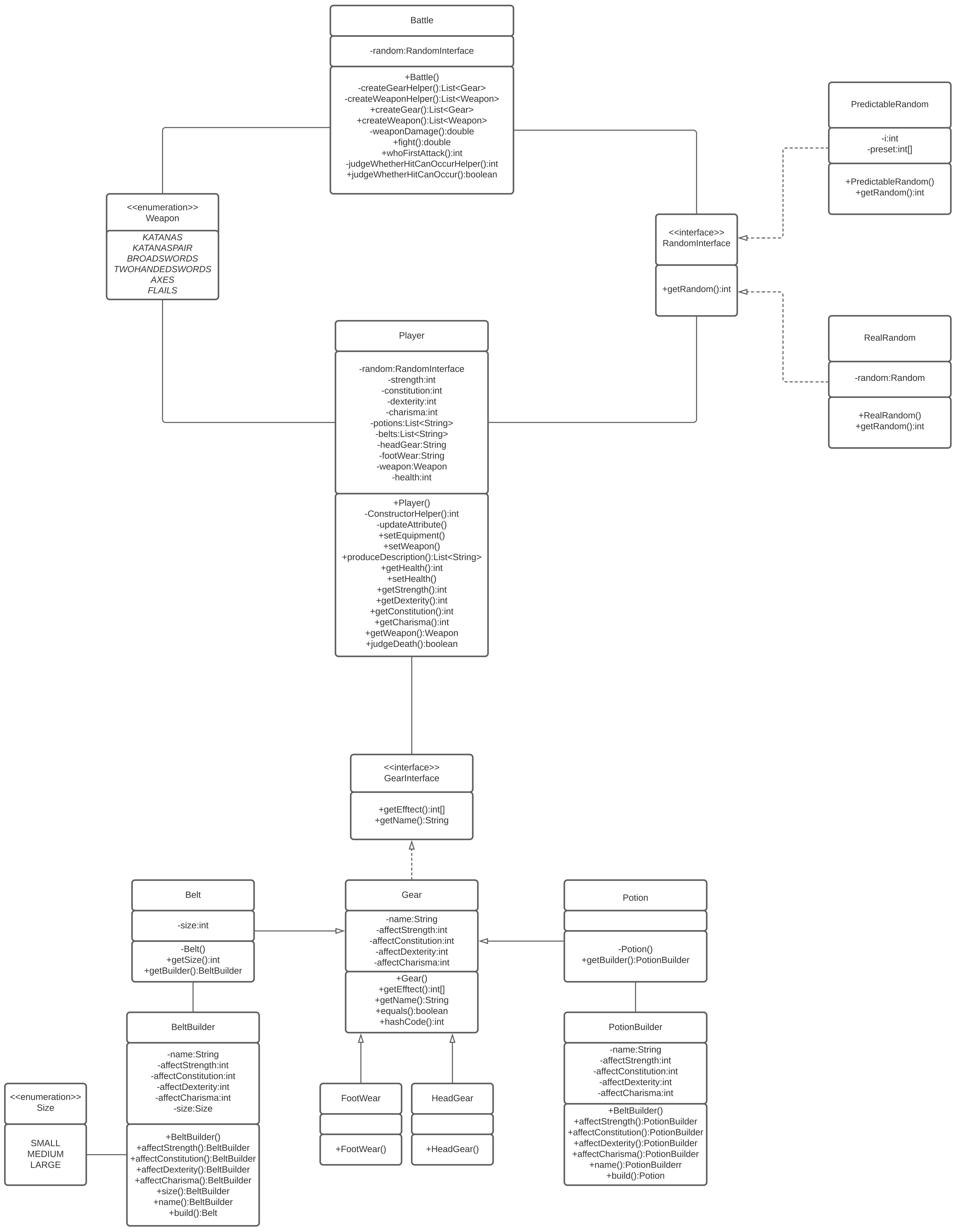
Battle
-random:RandomInterface
+Battle()
-createGearHelper():List<Gear>
-createWeaponHelper():List<Weapon>
+createGear():List<Gear>
+createWeapon():List<Weapon>
-weaponDamage():double
+fight():double
+whoFirstAttack():int
-judgeWhetherHitCanOccurHelper():int
+judgeWhetherHitCanOccur():boolean
PredictableRandom
-i:int
-preset:int[]
+PredictableRandom()
+getRandom():int
<<enumeration>>
Weapon
KATANAS
KATANASPAIR
BROADSWORDS
TWOHANDEDSWORDS
AXES
FLAILS
<<interface>>
RandomInterface
+getRandom():int
RealRandom
-random:Random
+RealRandom()
+getRandom():int
Player
-random:RandomInterface
-strength:int
-constitution:int
-dexterity:int
-charisma:int
-potions:List<String>
-belts:List<String>
-headGear:String
-footWear:String
-weapon:Weapon
-health:int
+Player()
-ConstructorHelper():int
-updateAttribute()
+setEquipment()
+setWeapon()
+produceDescription():List<String>
+getHealth():int
+setHealth()
+getStrength():int
+getDexterity():int
+getConstitution():int
+getCharisma():int
+getWeapon():Weapon
+judgeDeath():boolean
<<interface>>
GearInterface
+getEfftect():int[]
+getName():String
Belt
-size:int
-Belt()
+getSize():int
+getBuilder():BeltBuilder
Gear
-name:String
-affectStrength:int
-affectConstitution:int
-affectDexterity:int
-affectCharisma:int
+Gear()
+getEfftect():int[]
+getName():String
+equals():boolean
+hashCode():int
Potion
-Potion()
+getBuilder():PotionBuilder
BeltBuilder
-name:String
-affectStrength:int
-affectConstitution:int
-affectDexterity:int
-affectCharisma:int
-size:Size
+BeltBuilder()
+affectStrength():BeltBuilder
+affectConstitution():BeltBuilder
+affectDexterity():BeltBuilder
+affectCharisma():BeltBuilder
+size():BeltBuilder
+name():BeltBuilder
+build():Belt
PotionBuilder
-name:String
-affectStrength:int
-affectConstitution:int
-affectDexterity:int
-affectCharisma:int
+BeltBuilder()
+affectStrength():PotionBuilder
+affectConstitution():PotionBuilder
+affectDexterity():PotionBuilder
+affectCharisma():PotionBuilder
+name():PotionBuilderr
+build():Potion
<<enumeration>>
Size
SMALL
MEDIUM
LARGE
FootWear
+FootWear()
HeadGear
+HeadGear()

Battle Test

test what	input	expected value
construction with input null	Battle(null)	IllegalArgumentException
create gear	Battle.createGear()	a list of gear we set before
create weapon	Battle.createWeapon()	a list of weapon we set before
one player causes really damage to the other with weapon	2 players (player1 player2)	the damage caused by player1
Test one player causes really damage to the other without weapon.	2 players (player1 with weapon player2)	the damage caused by player1
Test one player' striking power is less than the other's avoidance ability.	2 players (player1 player2)	0
Test one player' striking power is larger than the other's avoidance ability.but the actual damage <= 0 with weapon	2 players (player1 with weapon player2)	0
Test one player' striking power is larger than the other's avoidance ability.but the actual damage <= 1 with no weapon	2 players (player1 player3)	1
Test who first attack. One player's charisma is larger than the other player.	2 players (player1 with higher, player2)	1
Test who first attack. One player's charisma is equal to the other player.	2 players (player1 player2)	random chosen from 1, 2
Test whether a hit can occur.	2 players (player1 player2)	decided by strikingPower and avoidance

player test

test what	input	expected value
construction with input null	player(null)	IllegalArgumentException
Test players to enter the arena with only their basic abilities and their bare hands.	player.getWeapon()	null
Test players to enter the arena with weapon.	player.getWeapon()	Weapon.AXES
Test players to enter the arena with weapon which is one Katana.	player.getWeapon()	Weapon.KATANAS
Test players to enter the arena with weapon which are two KATANAS.	player.getWeapon()	Weapon.KATANASPAIR
Test players to enter the arena with equipment.	player.setEquipment(res)	the updated 4 abilities
Test produce description for the player.	player.produceDescription()	"29", "30", "30", "29", "headgear1", "potion1", "potion2", "potion3",
Test get the health of the player.	player.getHealth()	118
Test the health after getting damage and reset the health to the beginning health.	player.setHealth(5),player.getHealth()	67
Test the health after getting damage and reset the health to the beginning health.	player.setHealth(),player.getHealth()	72
Test get the strength of the player.	player.getStrength()	18
Test get the dexterity of the player.	player.getDexterity()	18
Test get the constitution of the player.	player.getConstitution()	18
Test get the charisma of the player.	player.getCharisma()	18
Test get the weapon of the player.	player.getWeapon()	null
Test whether the player is alive.	player.judgeDeath()	FALSE

Belt test

test what	input	expected value
Test input without size.	Belt.getBuilder().name("belt").affectStrength(4).affectConstitution(1).build()	IllegalArgumentException
Test input without affecting any attribute.	belt=Belt.getBuilder().name("belt").size(Size.SMALL).build()	IllegalArgumentException
Test input affecting three attribute.	Belt.getBuilder().name("belt").size(Size.LARGE).affectStrength(1).affectConstitution	IllegalArgumentException
Test input affecting one attribute.	Belt.getBuilder().name("belt").size(Size.LARGE).affectStrength(1).affectDexterity(0).b	IllegalArgumentException
Test input affecting four attribute.	Belt.getBuilder().name("belt").size(Size.LARGE).affectStrength(1).affectConstitution	IllegalArgumentException
Test get the size of the belt.	belt.getSize()	1

footwear test

test what	input	expected value
Test input without effect.	footWear = new FootWear("footwear", 0)	IllegalArgumentException
Test only affect dexterity.	footWear.getEffect()	int[]{0, 0, 3, 0}

gear test

test what	input	expected value
Test the name of input is null.	gear = new Gear(null, 1, 1, 1, 1)	IllegalArgumentException
Test get the effect of the gear.	gear.getEffect()	int[]{1, 1, 1, 1}
Test get the name of the gear.	gear.getName()	"gear"

headgear test

test what	input	expected value
Test input without effect.	headGear = new HeadGear("headGear", 0)	IllegalArgumentException
Test only affect constitution.	headGear.getEffect()	int[]{0, 1, 0, 0}

potion test

test what	input	expected value
Test successfully create a potion by getting its effect on ability.	potion.getEffect()	int[]{1, 1, 1, 1}
Test input without affecting any attribute.	Potion.getBuilder().name("potion").builder()	IllegalArgumentException
Test input affecting one attribute.	Potion.getBuilder().name("potion").affectConstitution(1).builder()	IllegalArgumentException
Test input affecting two attribute.	Potion.getBuilder().name("potion").affectConstitution(1).affectDexterity(1).builder()	IllegalArgumentException

Test input affecting three attribute.	Potion.getBuilder().name("potion").affectConstitution(1).affectDexterity(1).affectChar	IllegalArgumentException

randominterface test

test what	input	expected value
Test whether there is actual randomness.	int i = random.getRandom(1, 5); int i1 = random.getRandom(1, 5)	assertTrue(i != i1)

Test whether created values are all in the range.	random = this.random.getRandom(1, 10)	assertTrue(random >= 1)
		assertTrue(random <= 10)