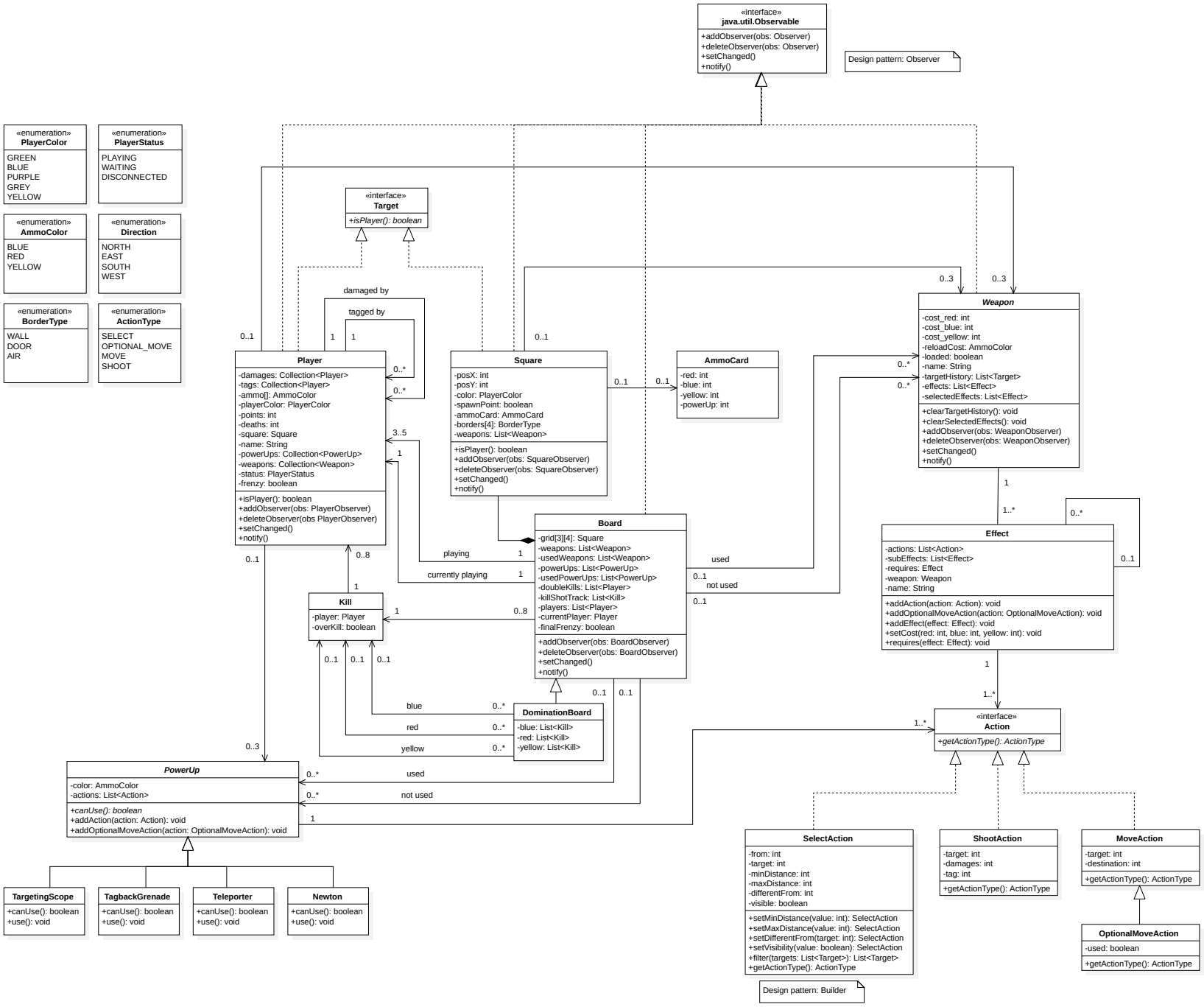
«interface»
java.util.Observable
+addObserver(obs: Observer)
+deleteObserver(obs: Observer)
+setChanged()
+notify()
Design pattern: Observer
«enumeration»
PlayerColor
GREEN
BLUE
PURPLE
GREY
YELLOW
«enumeration»
PlayerStatus
PLAYING
WAITING
DISCONNECTED
«enumeration»
AmmoColor
BLUE
RED
YELLOW
«enumeration»
Direction
NORTH
EAST
SOUTH
WEST
«enumeration»
BorderType
WALL
DOOR
AIR
«enumeration»
ActionType
SELECT
OPTIONAL_MOVE
MOVE
SHOOT
«interface»
Target
+isPlayer(): boolean
damaged by
tagged by
Player
-damages: Collection<Player>
-tags: Collection<Player>
-ammo[]: AmmoColor
-playerColor: PlayerColor
-points: int
-deaths: int
-square: Square
-name: String
-powerUps: Collection<PowerUp>
-weapons: Collection<Weapon>
-status: PlayerStatus
-frenzy: boolean
+isPlayer(): boolean
+addObserver(obs: PlayerObserver)
+deleteObserver(obs PlayerObserver)
+setChanged()
+notify()
Square
-posX: int
-posY: int
-color: PlayerColor
-spawnPoint: boolean
-ammoCard: AmmoCard
-borders[4]: BorderType
-weapons: List<Weapon>
+isPlayer(): boolean
+addObserver(obs: SquareObserver)
+deleteObserver(obs: SquareObserver)
+setChanged()
+notify()
AmmoCard
-red: int
-blue: int
-yellow: int
-powerUp: int
Weapon
-cost_red: int
-cost_blue: int
-cost_yellow: int
-reloadCost: AmmoColor
-loaded: boolean
-name: String
-targetHistory: List<Target>
-effects: List<Effect>
-selectedEffects: List<Effect>
+clearTargetHistory(): void
+clearSelectedEffects(): void
+addObserver(obs: WeaponObserver)
+deleteObserver(obs: WeaponObserver)
+setChanged()
+notify()
Board
-grid[3][4]: Square
-weapons: List<Weapon>
-usedWeapons: List<Weapon>
-powerUps: List<PowerUp>
-usedPowerUps: List<PowerUp>
-doubleKills: List<Player>
-killShotTrack: List<Kill>
-players: List<Player>
-currentPlayer: Player
-finalFrenzy: boolean
+addObserver(obs: BoardObserver)
+deleteObserver(obs: BoardObserver)
+setChanged()
+notify()
Effect
-actions: List<Action>
-subEffects: List<Effect>
-requires: Effect
-weapon: Weapon
-name: String
+addAction(action: Action): void
+addOptionalMoveAction(action: OptionalMoveAction): void
+addEffect(effect: Effect): void
+setCost(red: int, blue: int, yellow: int): void
+requires(effect: Effect): void
Kill
-player: Player
-overKill: boolean
DominationBoard
-blue: List<Kill>
-red: List<Kill>
-yellow: List<Kill>
«interface»
Action
+getActionType(): ActionType
PowerUp
-color: AmmoColor
-actions: List<Action>
+canUse(): boolean
+addAction(action: Action): void
+addOptionalMoveAction(action: OptionalMoveAction): void
TargetingScope
+canUse(): boolean
+use(): void
TagbackGrenade
+canUse(): boolean
+use(): void
Teleporter
+canUse(): boolean
+use(): void
Newton
+canUse(): boolean
+use(): void
SelectAction
-from: int
-target: int
-minDistance: int
-maxDistance: int
-differentFrom: int
-visible: boolean
+setMinDistance(value: int): SelectAction
+setMaxDistance(value: int): SelectAction
+setDifferentFrom(target: int): SelectAction
+setVisibility(value: boolean): SelectAction
+filter(targets: List<Target>): List<Target>
+getActionType(): ActionType
Design pattern: Builder
ShootAction
-target: int
-damages: int
-tag: int
+getActionType(): ActionType
MoveAction
-target: int
-destination: int
+getActionType(): ActionType
OptionalMoveAction
-used: boolean
+getActionType(): ActionType
playing
currently playing
used
not used
blue
red
yellow
0..1
0..*
1
0..3
3..5
0..8
1..*