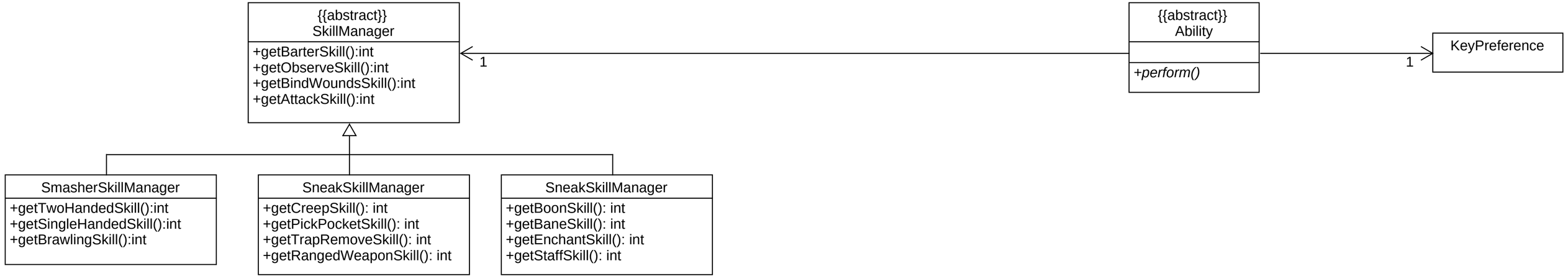
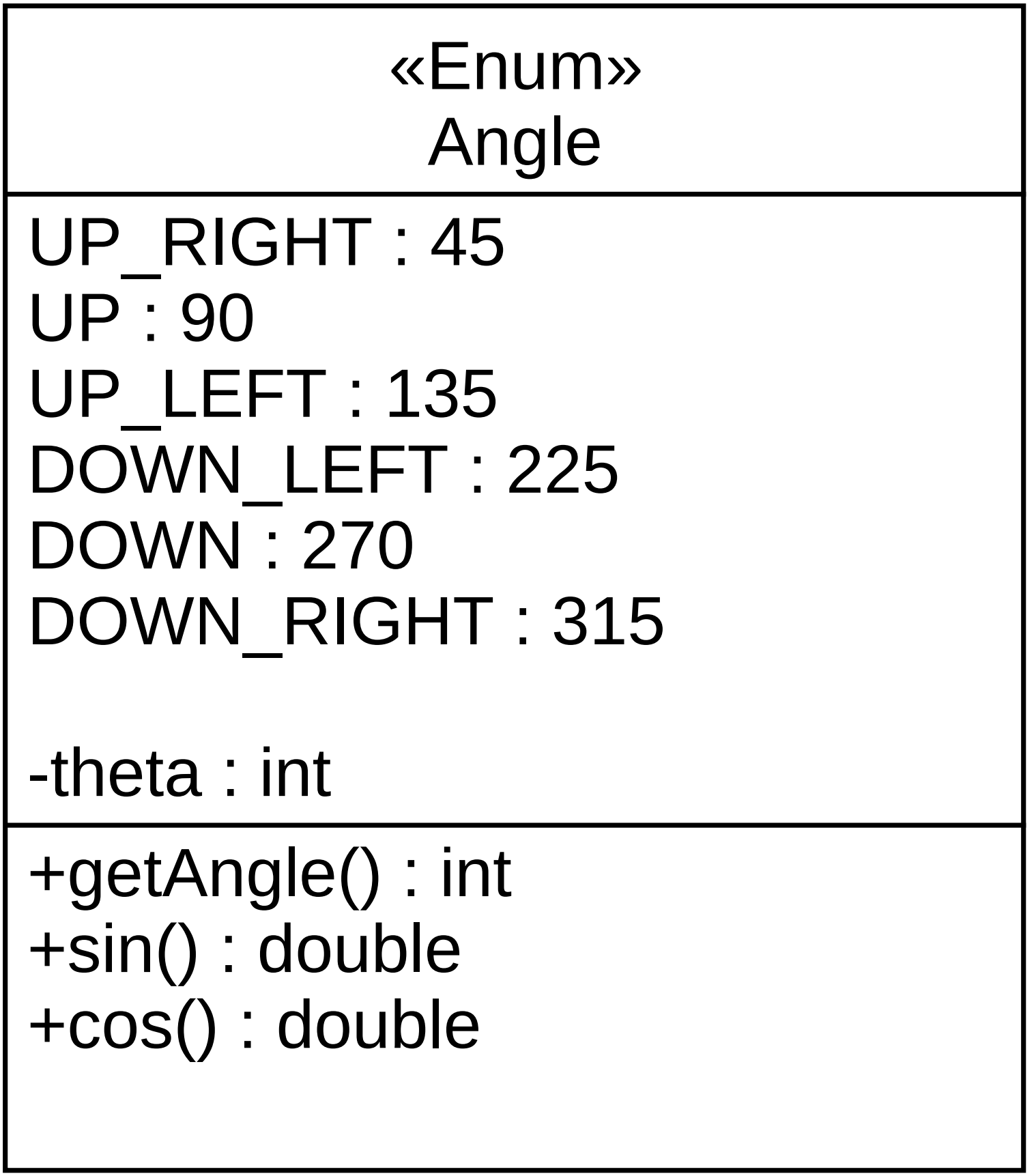
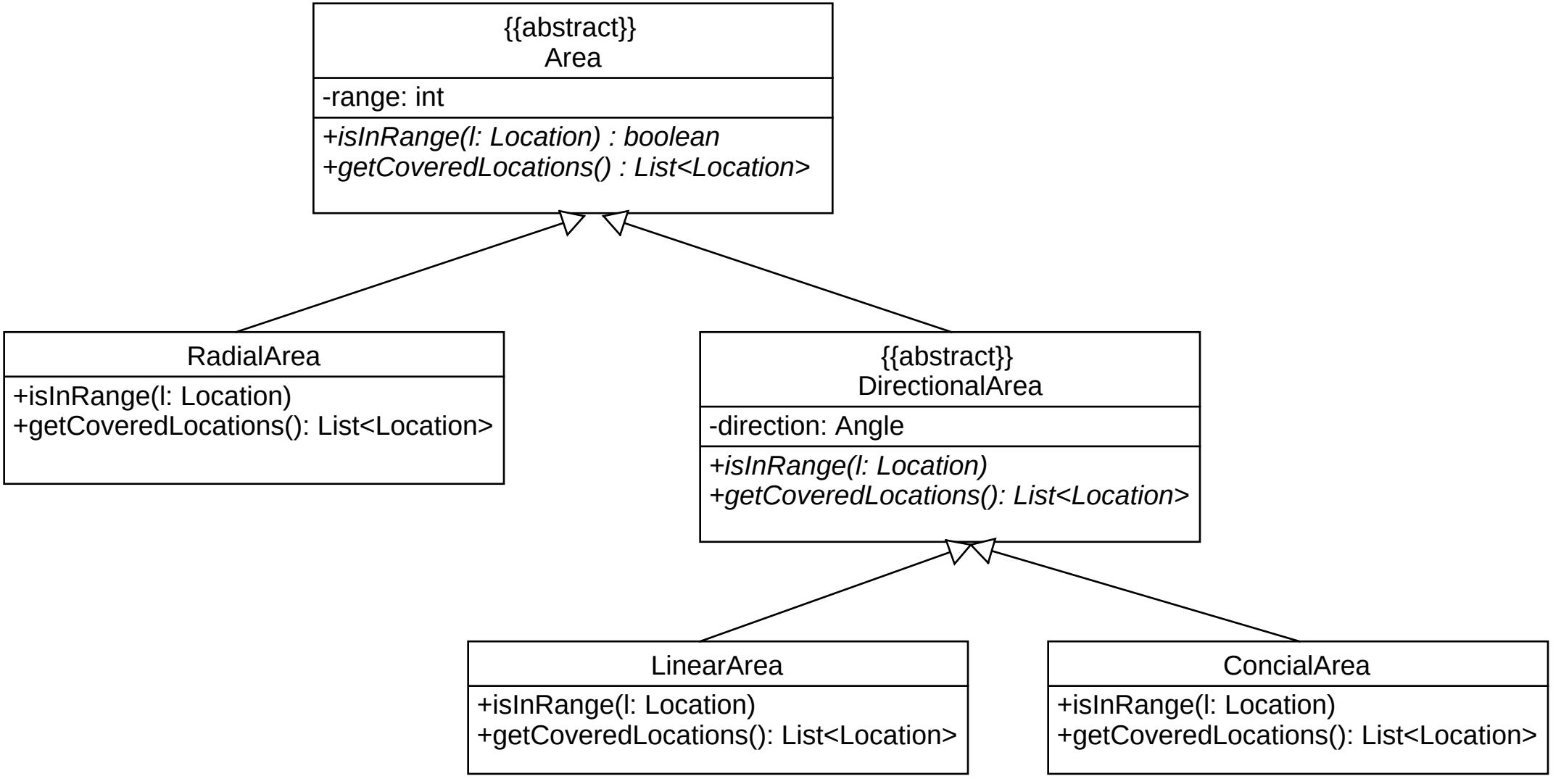
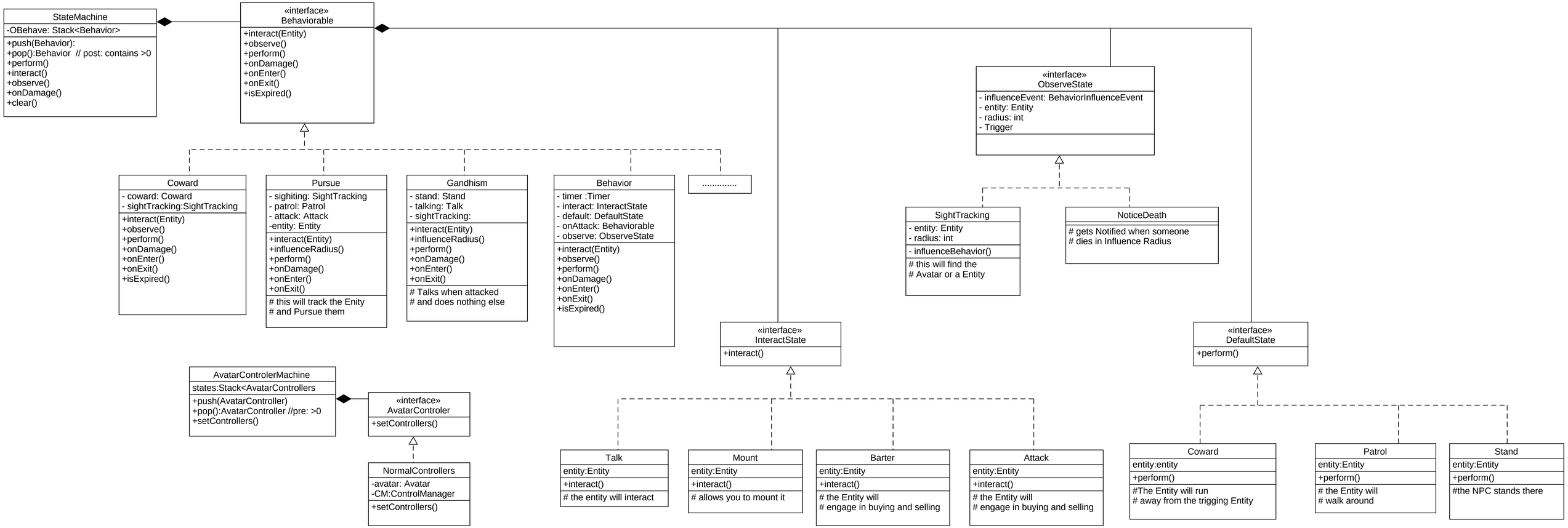
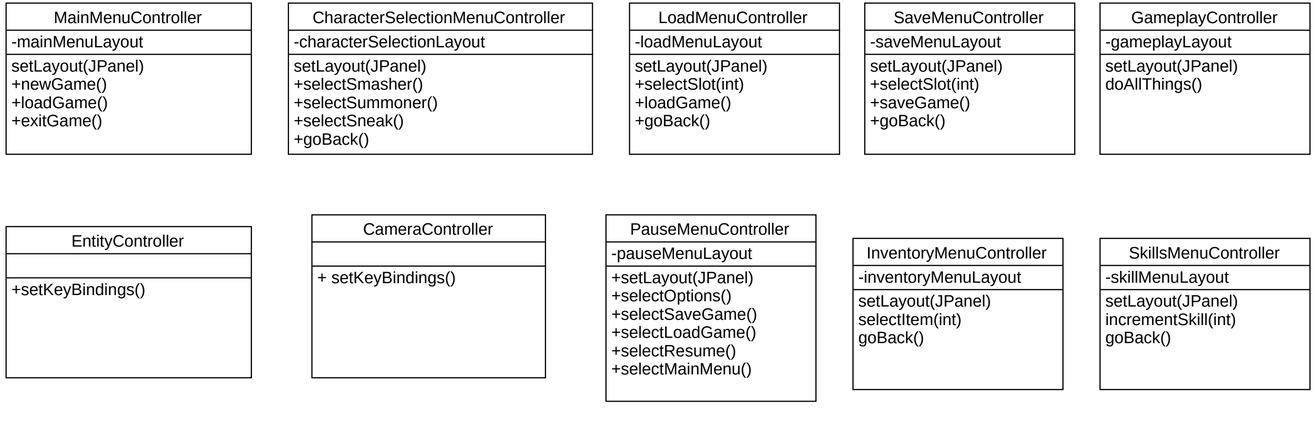
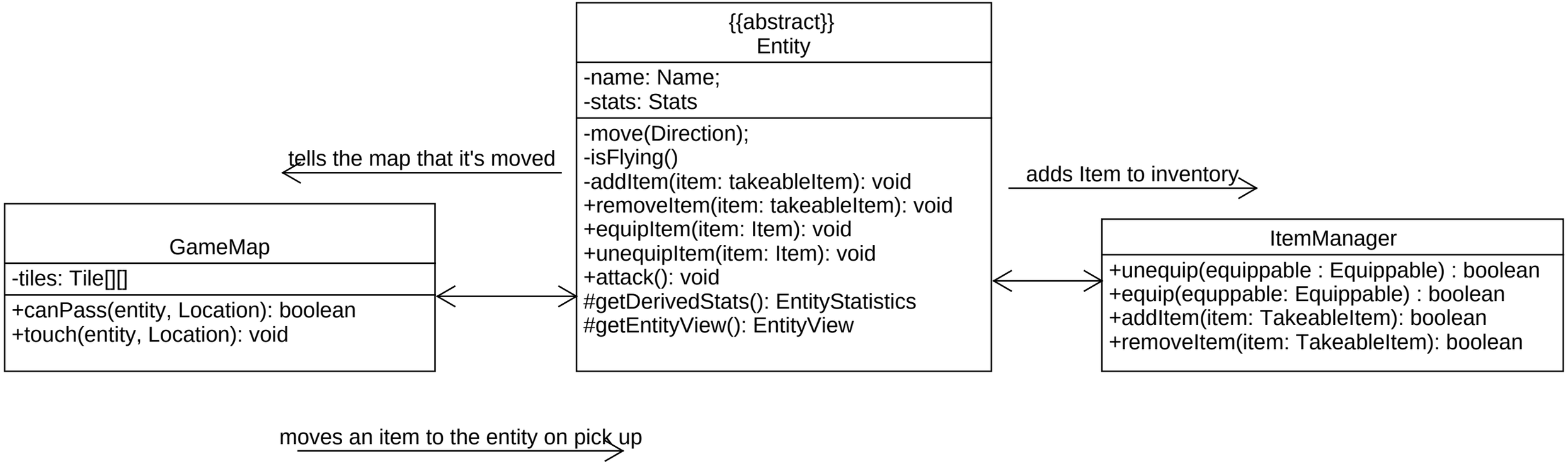

{{abstract}}
SkillManager
+getBarterSkill():int
+getObserveSkill():int
+getBindWoundsSkill():int
+getAttackSkill():int
1
{{abstract}}
Ability
+perform()
1
KeyPreference
SmasherSkillManager
+getTwoHandedSkill():int
+getSingleHandedSkill():int
+getBrawlingSkill():int
SneakSkillManager
+getCreepSkill(): int
+getPickPocketSkill(): int
+getTrapRemoveSkill(): int
+getRangedWeaponSkill(): int
SneakSkillManager
+getBoonSkill(): int
+getBaneSkill(): int
+getEnchantSkill(): int
+getStaffSkill(): int


«Enum»
Angle
UP_RIGHT : 45
UP : 90
UP_LEFT : 135
DOWN_LEFT : 225
DOWN : 270
DOWN_RIGHT : 315
-theta : int
+getAngle() : int
+sin() : double
+cos() : double


{{abstract}}
Area
-range: int
+isInRange(l: Location) : boolean
+getCoveredLocations() : List<Location>
RadialArea
+isInRange(l: Location)
+getCoveredLocations(): List<Location>
{{abstract}}
DirectionalArea
-direction: Angle
+isInRange(l: Location)
+getCoveredLocations(): List<Location>
LinearArea
+isInRange(l: Location)
+getCoveredLocations(): List<Location>
ConcialArea
+isInRange(l: Location)
+getCoveredLocations(): List<Location>


StateMachine
-OBehave: Stack<Behavior>
+push(Behavior):
+pop():Behavior // post: contains >0
+perform()
+interact()
+observe()
+onDamage()
+clear()
«interface»
Behaviorable
+interact(Entity)
+observe()
+perform()
+onDamage()
+onEnter()
+onExit()
+isExpired()
Coward
- coward: Coward
- sightTracking:SightTracking
+interact(Entity)
+observe()
+perform()
+onDamage()
+onEnter()
+onExit()
+isExpired()
Pursue
- sighiting: SightTracking
- patrol: Patrol
- attack: Attack
-entity: Entity
+interact(Entity)
+influenceRadius()
+perform()
+onDamage()
+onEnter()
+onExit()
this will track the Enity
and Pursue them
Gandhism
- stand: Stand
- talking: Talk
- sightTracking:
+interact(Entity)
+influenceRadius()
+perform()
+onDamage()
+onEnter()
+onExit()
Talks when attacked
and does nothing else
Behavior
- timer :Timer
- interact: InteractState
- default: DefaultState
- onAttack: Behaviorable
- observe: ObserveState
+interact(Entity)
+observe()
+perform()
+onDamage()
+onEnter()
+onExit()
+isExpired()
.....
«interface»
ObserveState
- influenceEvent: BehaviorInfluenceEvent
- entity: Entity
- radius: int
- Trigger
SightTracking
- entity: Entity
- radius: int
- influenceBehavior()
this will find the
Avatar or a Entity
NoticeDeath
gets Notified when someone
dies in Influence Radius
«interface»
InteractState
+interact()
«interface»
DefaultState
+perform()
AvatarControlerMachine
states:Stack<AvatarControllers
+push(AvatarController)
+pop():AvatarController //pre: >0
+setControllers()
«interface»
AvatarControler
+setControllers()
NormalControllers
-avatar: Avatar
-CM:ControlManager
+setControllers()
Talk
entity:Entity
+interact()
the entity will interact
Mount
entity:Entity
+interact()
allows you to mount it
Barter
entity:Entity
+interact()
the Entity will
engage in buying and selling
Attack
entity:Entity
+interact()
the Entity will
engage in buying and selling
Coward
entity:entity
+perform()
#The Entity will run
away from the trigging Entity
Patrol
entity:Entity
+perform()
the Entity will
walk around
Stand
entity:Entity
+perform()
#the NPC stands there


MainMenuController
-mainMenuLayout
setLayout(JPanel)
+newGame()
+loadGame()
+exitGame()
CharacterSelectionMenuController
-characterSelectionLayout
setLayout(JPanel)
+selectSmasher()
+selectSummoner()
+selectSneak()
+goBack()
LoadMenuController
-loadMenuLayout
setLayout(JPanel)
+selectSlot(int)
+loadGame()
+goBack()
SaveMenuController
-saveMenuLayout
setLayout(JPanel)
+selectSlot(int)
+saveGame()
+goBack()
GameplayController
-gameplayLayout
setLayout(JPanel)
doAllThings()
EntityController
+setKeyBindings()
CameraController
+ setKeyBindings()
PauseMenuController
-pauseMenuLayout
+setLayout(JPanel)
+selectOptions()
+selectSaveGame()
+selectLoadGame()
+selectResume()
+selectMainMenu()
InventoryMenuController
-inventoryMenuLayout
setLayout(JPanel)
selectItem(int)
goBack()
SkillsMenuController
-skillMenuLayout
setLayout(JPanel)
incrementSkill(int)
goBack()
TradeMenuController
-tradeLayout
+ sellItem(itemSlot: int)
+ buyItem(itemSlot: int)
+ goBack()
DialogController
-tradeLayout
+getText(): String


{{abstract}}
Entity
-name: Name;
-stats: Stats
-move(Direction);
-isFlying()
-addItem(item: takeableItem): void
+removeItem(item: takeableItem): void
+equipItem(item: Item): void
+unequipItem(item: Item): void
+attack(): void
#getDerivedStats(): EntityStatistics
#getEntityView(): EntityView
tells the map that it's moved
adds Item to inventory
GameMap
-tiles: Tile[][]
+canPass(entity, Location): boolean
+touch(entity, Location): void
ItemManager
+unequip(equippable : Equippable) : boolean
+equip(equppable: Equippable) : boolean
+addItem(item: TakeableItem): boolean
+removeItem(item: TakeableItem): boolean
moves an item to the entity on pick up

Entity

EntityManager

-partyNpcs: List<NPC>
-nonPartyNpcs: List<NPC>
-avatar: Avatar

+update(): void //calls update on all entities
+addPartyNpc(npc: NPC)
+removePartyNPC(npc: NPC)
+getEntityAtLocation(location: Location): Entity // returns NULL if not found

{{abstract}}
Entity

-name: Name;
-stats: Stats

-move(Direction);
-isFlying(): bool
+addItem(item: takeableItem): void
+removeItem(item: takeableItem): void
+equipItem(item: Item): void
+unequipItem(item: Item): void
+attack(): void
#getDerivedStats(): EntityStatistics
#getEntityView(): EntityView

{{abstract}}
Avatar

-name: Name;
-stats: Stats
-SkillManager: Skills
-abilities: List<Ability>

-isFlying(): bool
-move(Direction);
+addItem(item: takeableItem): void
+removeItem(item: takeableItem): void
+equipItem(item: Item): void
+unequipItem(item: Item): void
+getKeyListeners() : Listeners
#getDerivedStats(): EntityStatistics
#getSkillManager(): SkillManager
#getEntityView(): EntityView

NPC

-behavior: Behavior
-attackAbility: Ability

PET

-behavior: Behavior

Mount

-behavior: Behavior
-attackAbility: Ability
-rider: Avatar

-setRider(Avatar) //Mount contains rider, and rider is set by the mount in the interaction dialog.

Smasher

+attack(): void
#getSkillManager(): SmasherSkillManager
#getEntityView(): SmasherView

Summoner

+attack(): void //Handles staff skill
+bane(): void
+boon(): void
+enchantment():void
#getSkillManager(): SummonerSkillManager
#getEntityView(): SummonerView

Sneak

+attack(): void //Handles skill rangedAttack
+creep(): void
+pickPocket():void
+detectTrap():void
+removeTrap(Trap t):void //detect trap will call remove trap on traps it finds.
#getSkillManager(): SneakSkillManager
#getEntityView(): SneakView

Events

EventManager

-eventList: List<Event>

+update(): void
+addEvent(event: Event): void
#removeEvent(event: Event): boolean // Used to remove

{{abstract}}
Event

-duration: long (ms)

+Event(double duration)
+onBegin(): void //called by the event manager
+onExpire(): void //called by the event manager
+hasExpired(): boolean
+perform(): void

{{abstract}}
UnSourcedEvent

-target: Entity

+onBegin(): void
+onExpire(): void
+hasExpired(): boolean
+perform(): void

SourcedEvent

-target: Entity
-destination: Entity

+onBegin(): void
+onExpire(): void
+hasExpired(): boolean
+perform(): void

PrintEvent

-printDesintation: View?

+onBegin(): void
+onExpire(): void
+hasExpired(): boolean
+perform(): void

SkillModifierEvent

+onBegin(): void
+onExpire(): void
+hasExpired(): boolean
+perform(): void

StatisticModifierEvent

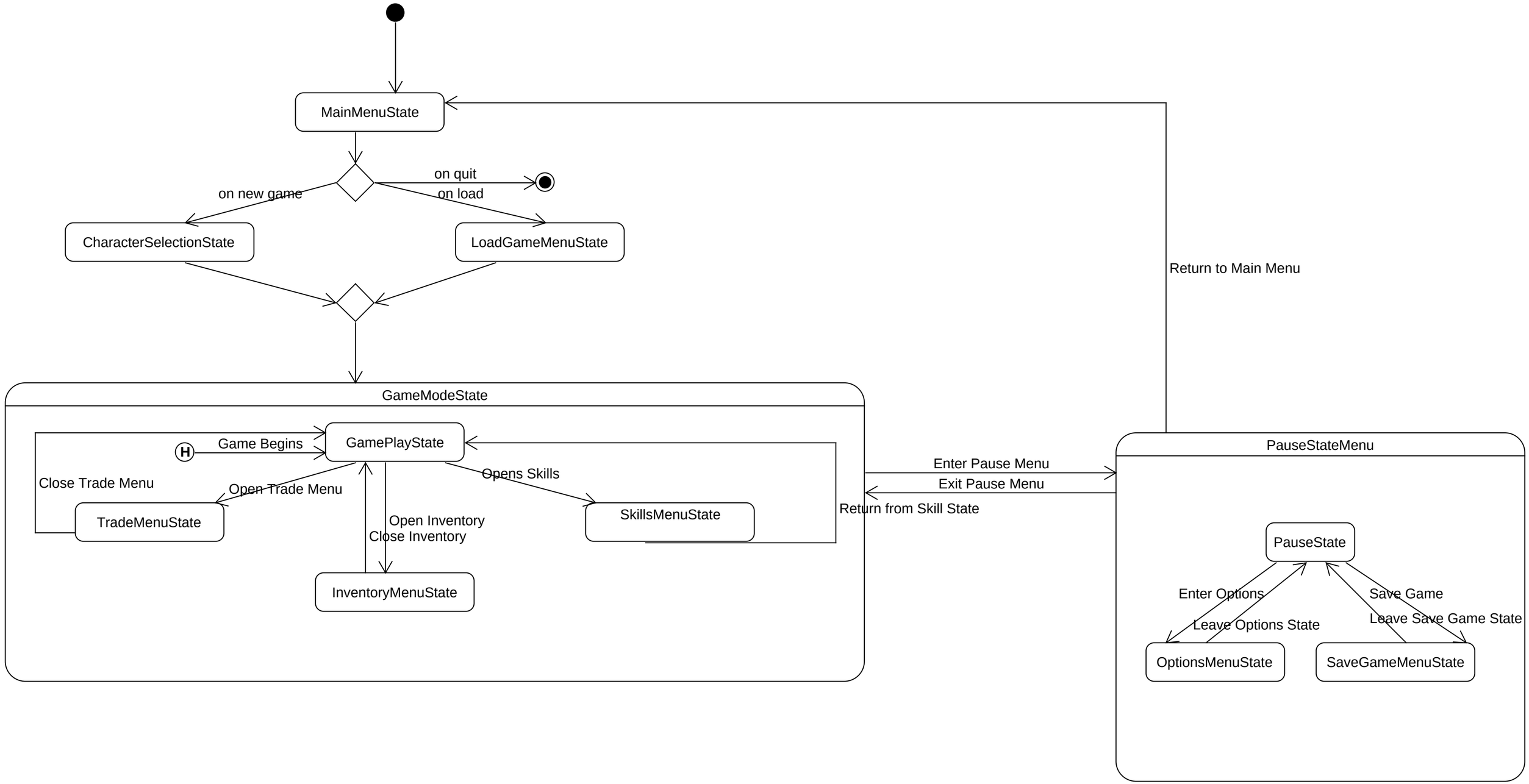
-statistics: EntityStatistics

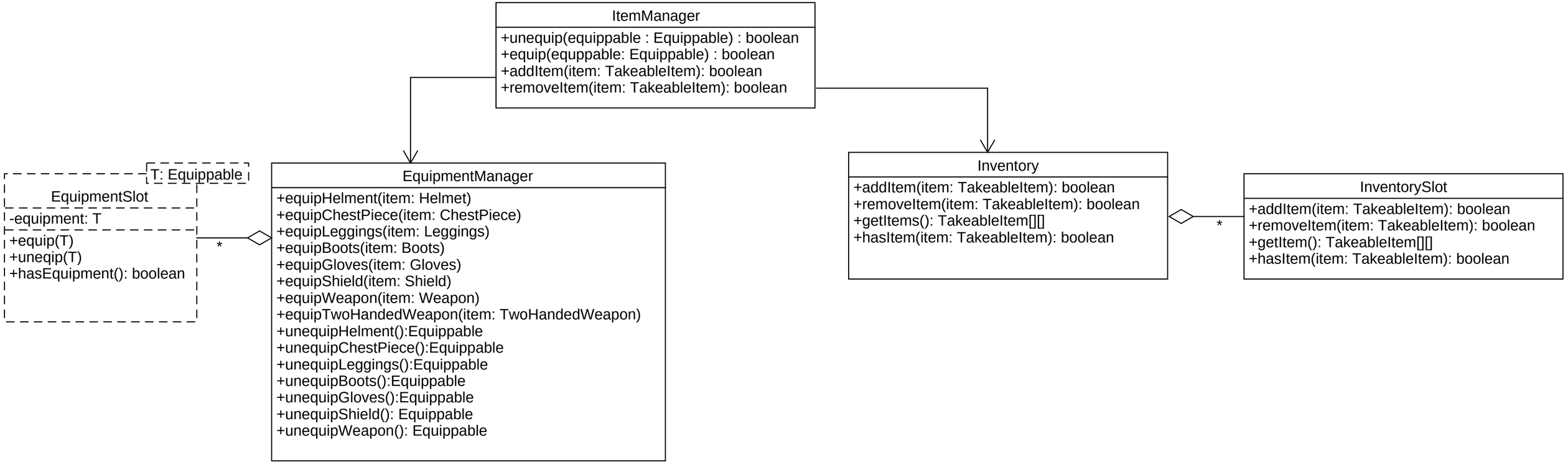
+onBegin(): void
+onExpire(): void
+hasExpired(): boolean
+perform(): void

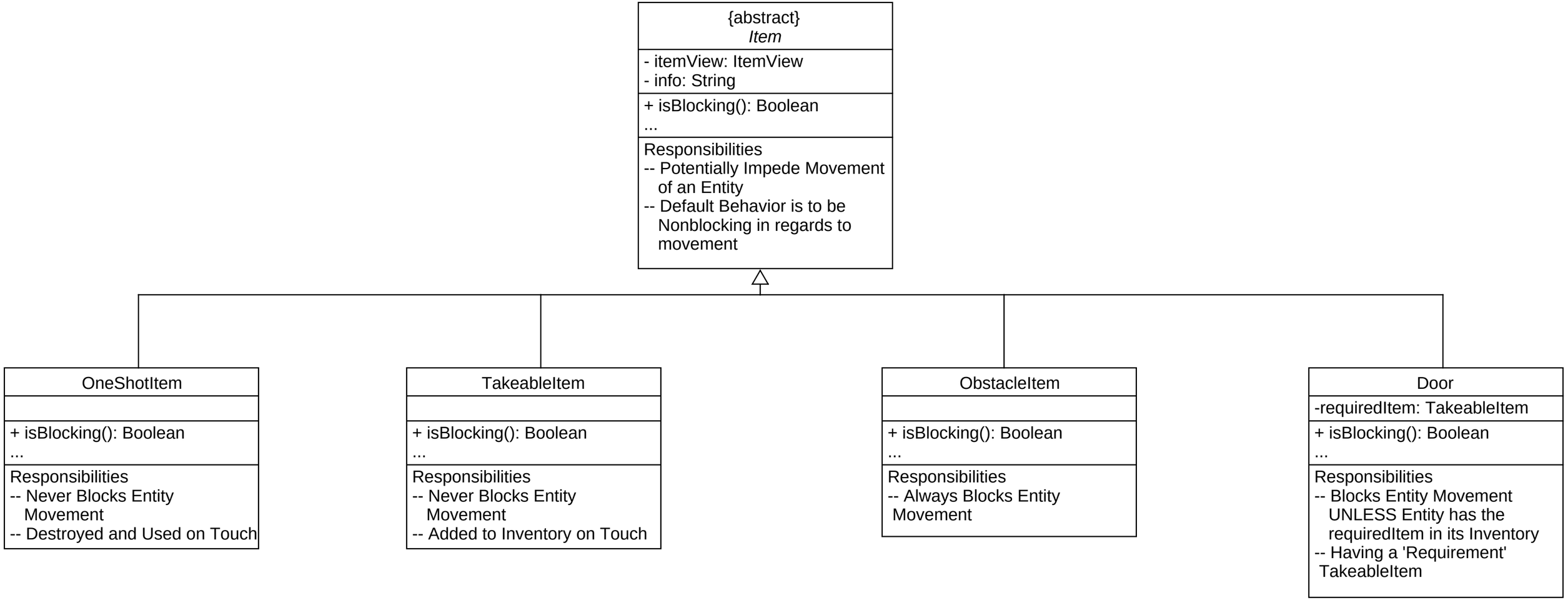
BehaviorModifierEvent

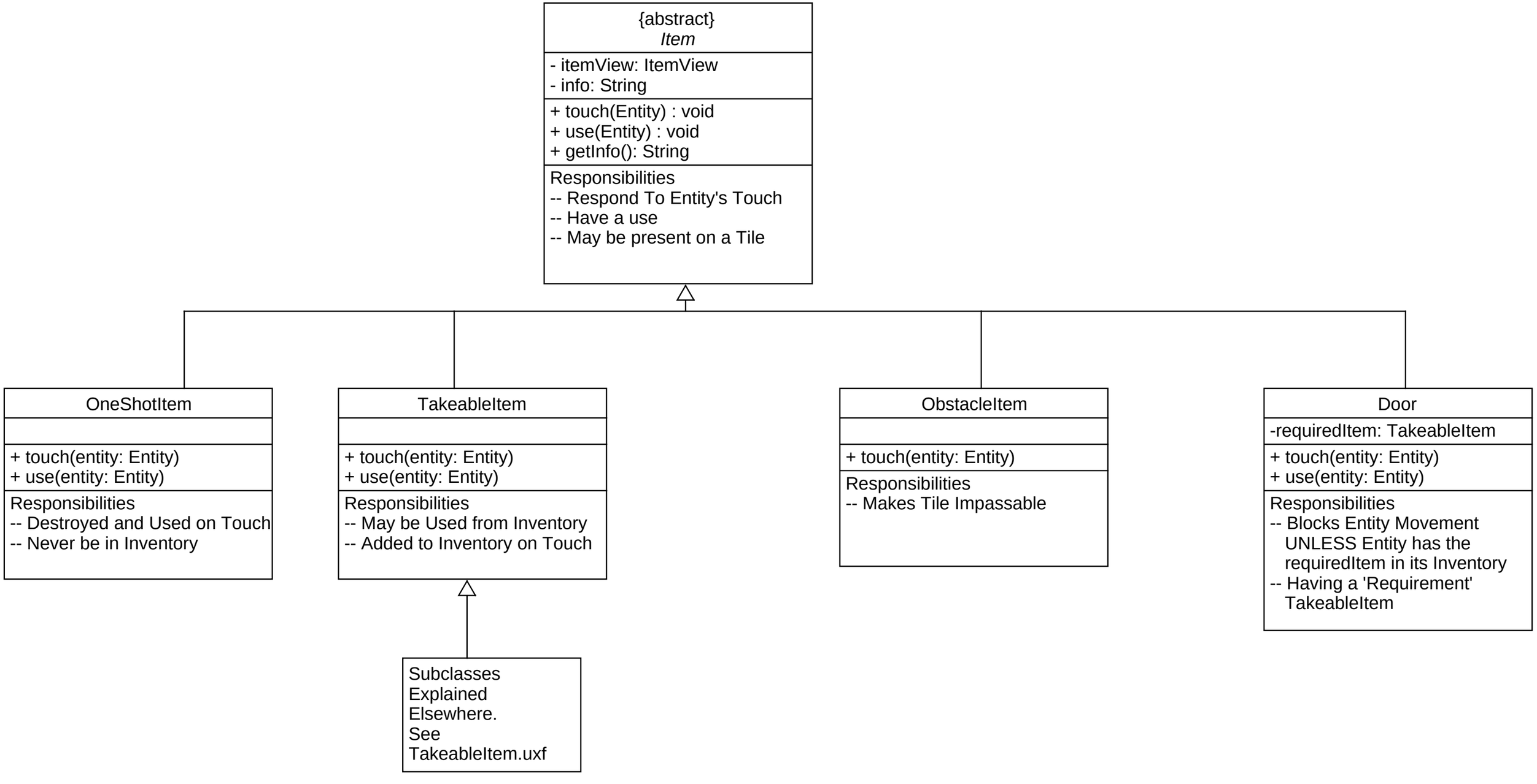
-behavior: Behavior

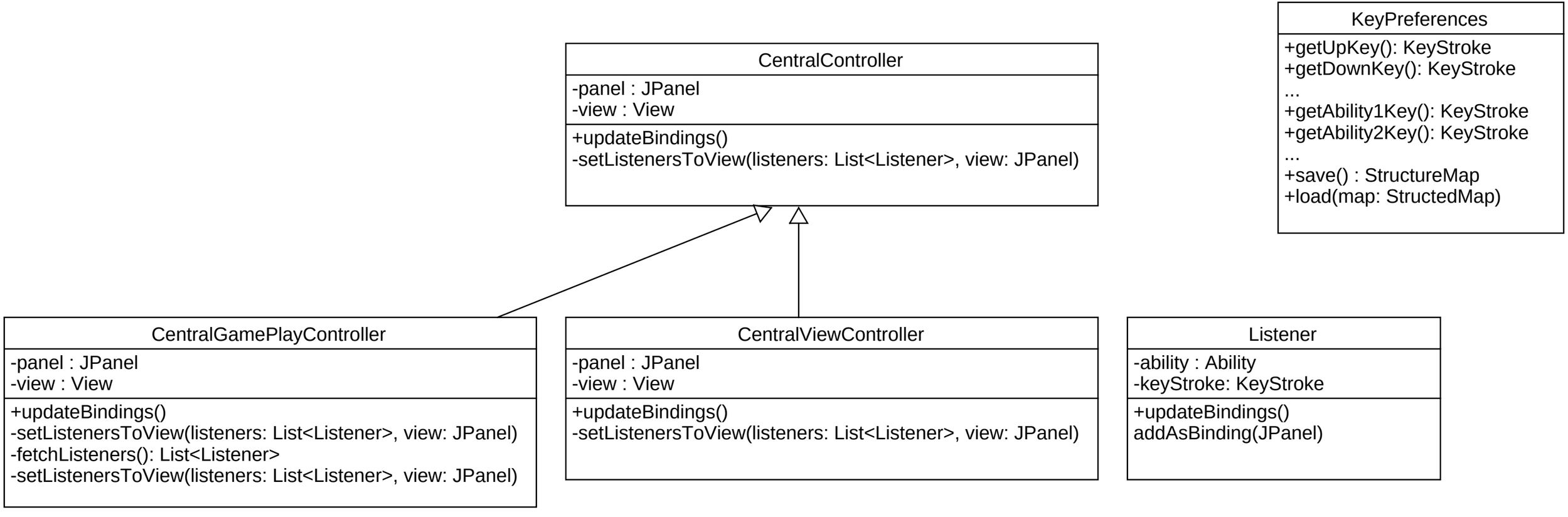
+onBegin(): void
+onExpire(): void
+hasExpired(): boolean
+perform(): void

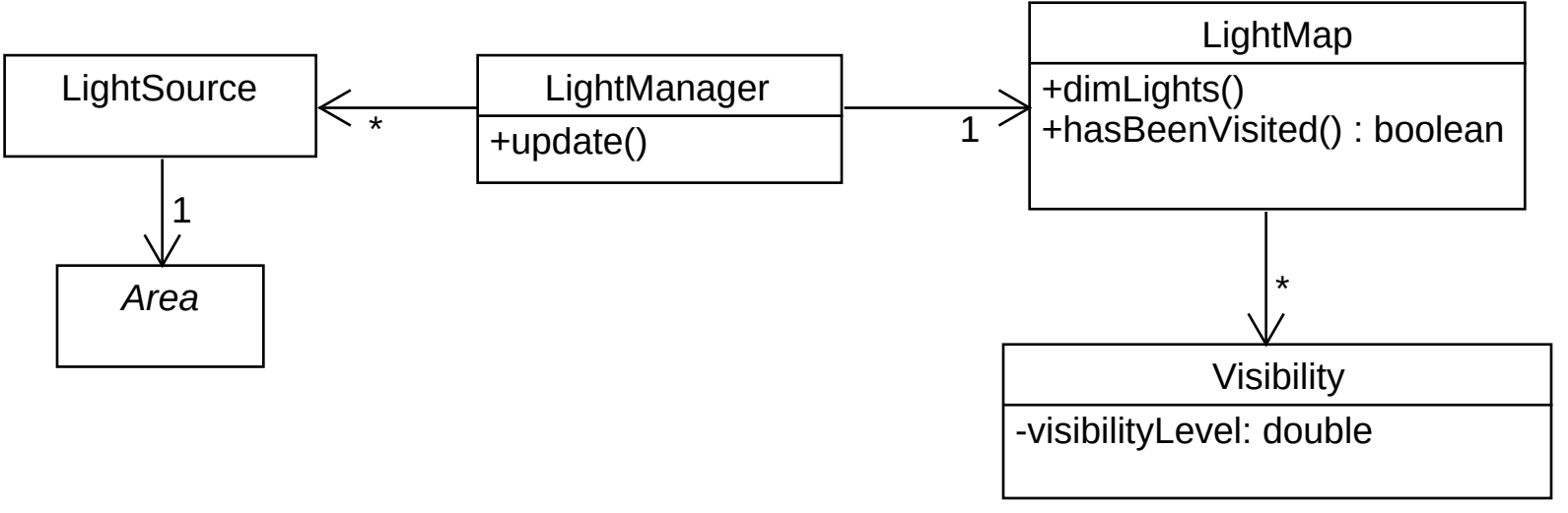

MainMenuState
on new game
on quit
on load
CharacterSelectionState
LoadGameMenuState
Return to Main Menu
GameModeState
H
Game Begins
GamePlayState
Close Trade Menu
Open Trade Menu
TradeMenuState
Opens Skills
SkillsMenuState
Open Inventory
Close Inventory
InventoryMenuState
Enter Pause Menu
Exit Pause Menu
Return from Skill State
PauseStateMenu
PauseState
Enter Options
Leave Options State
Save Game
Leave Save Game State
OptionsMenuState
SaveGameMenuState


ItemManager
+unequip(equippable : Equippable) : boolean
+equip(equppable: Equippable) : boolean
+addItem(item: TakeableItem): boolean
+removeItem(item: TakeableItem): boolean
T: Equippable
EquipmentSlot
-equipment: T
+equip(T)
+uneqip(T)
+hasEquipment(): boolean
*
EquipmentManager
+equipHelment(item: Helmet)
+equipChestPiece(item: ChestPiece)
+equipLeggings(item: Leggings)
+equipBoots(item: Boots)
+equipGloves(item: Gloves)
+equipShield(item: Shield)
+equipWeapon(item: Weapon)
+equipTwoHandedWeapon(item: TwoHandedWeapon)
+unequipHelment():Equippable
+unequipChestPiece():Equippable
+unequipLeggings():Equippable
+unequipBoots():Equippable
+unequipGloves():Equippable
+unequipShield(): Equippable
+unequipWeapon(): Equippable
Inventory
+addItem(item: TakeableItem): boolean
+removeItem(item: TakeableItem): boolean
+getItems(): TakeableItem[][]
+hasItem(item: TakeableItem): boolean
*
InventorySlot
+addItem(item: TakeableItem): boolean
+removeItem(item: TakeableItem): boolean
+getItem(): TakeableItem[][]
+hasItem(item: TakeableItem): boolean


{abstract}
Item
- itemView: ItemView
- info: String
+ isBlocking(): Boolean
...
Responsibilities
-- Potentially Impede Movement of an Entity
-- Default Behavior is to be Nonblocking in regards to movement
OneShotItem
+ isBlocking(): Boolean
...
Responsibilities
-- Never Blocks Entity Movement
-- Destroyed and Used on Touch
TakeableItem
+ isBlocking(): Boolean
...
Responsibilities
-- Never Blocks Entity Movement
-- Added to Inventory on Touch
ObstacleItem
+ isBlocking(): Boolean
...
Responsibilities
-- Always Blocks Entity Movement
Door
-requiredItem: TakeableItem
+ isBlocking(): Boolean
...
Responsibilities
-- Blocks Entity Movement UNLESS Entity has the requiredItem in its Inventory
-- Having a 'Requirement' TakeableItem


{abstract}
Item
- itemView: ItemView
- info: String
+ touch(Entity) : void
+ use(Entity) : void
+ getInfo(): String
Responsibilities
-- Respond To Entity's Touch
-- Have a use
-- May be present on a Tile
OneShotItem
+ touch(entity: Entity)
+ use(entity: Entity)
Responsibilities
-- Destroyed and Used on Touch
-- Never be in Inventory
TakeableItem
+ touch(entity: Entity)
+ use(entity: Entity)
Responsibilities
-- May be Used from Inventory
-- Added to Inventory on Touch
Subclasses Explained Elsewhere. See TakeableItem.uxf
ObstacleItem
+ touch(entity: Entity)
Responsibilities
-- Makes Tile Impassable
Door
-requiredItem: TakeableItem
+ touch(entity: Entity)
+ use(entity: Entity)
Responsibilities
-- Blocks Entity Movement UNLESS Entity has the requiredItem in its Inventory
-- Having a 'Requirement' TakeableItem


KeyPreferences
+getUpKey(): KeyStroke
+getDownKey(): KeyStroke
...
+getAbility1Key(): KeyStroke
+getAbility2Key(): KeyStroke
...
+save() : StructureMap
+load(map: StructedMap)
CentralController
-panel : JPanel
-view : View
+updateBindings()
-setListenersToView(listeners: List<Listener>, view: JPanel)
CentralGamePlayController
-panel : JPanel
-view : View
+updateBindings()
-setListenersToView(listeners: List<Listener>, view: JPanel)
-fetchListeners(): List<Listener>
-setListenersToView(listeners: List<Listener>, view: JPanel)
CentralViewController
-panel : JPanel
-view : View
+updateBindings()
-setListenersToView(listeners: List<Listener>, view: JPanel)
Listener
-ability : Ability
-keyStroke: KeyStroke
+updateBindings()
addAsBinding(JPanel)


LightSource
LightManager
+update()
LightMap
+dimLights()
+hasBeenVisited() : boolean
Area
Visibility
-visibilityLevel: double
1
*
1
*

Map

GameMap
-tiles: Tile[][]
+canPass(entity, Location): boolean
+touch(entity, Location): void

{{ Abstract }}
Tile
-tileView: TileView
+isPassable(Entity): boolean
+touch(Entity): void

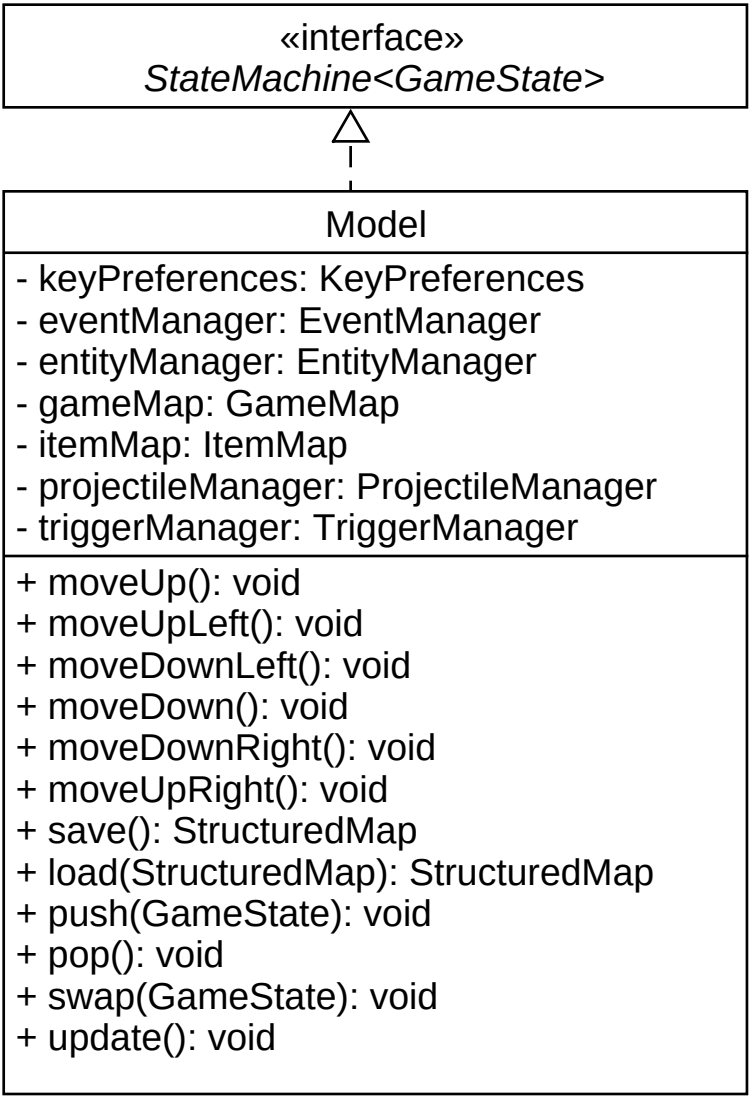
PassableTile
+isPassable(Entity): boolean
+touch(Entity): void

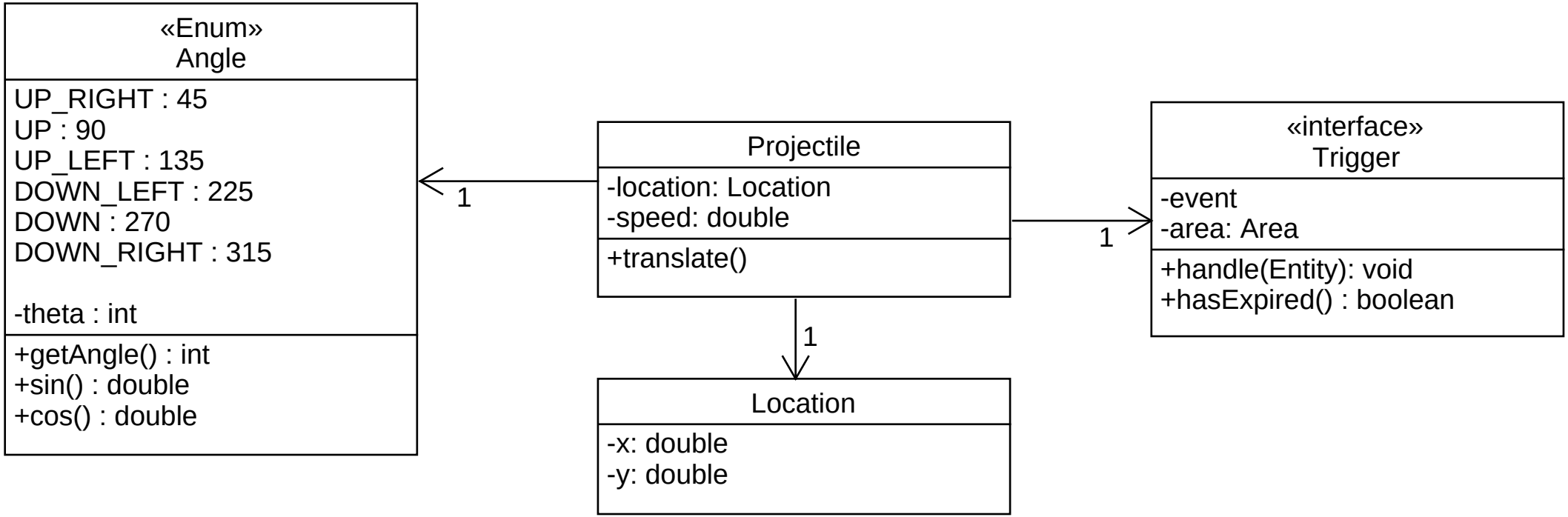
ImpassableTile
+isPassable(Entity): boolean
+touch(Entity): void

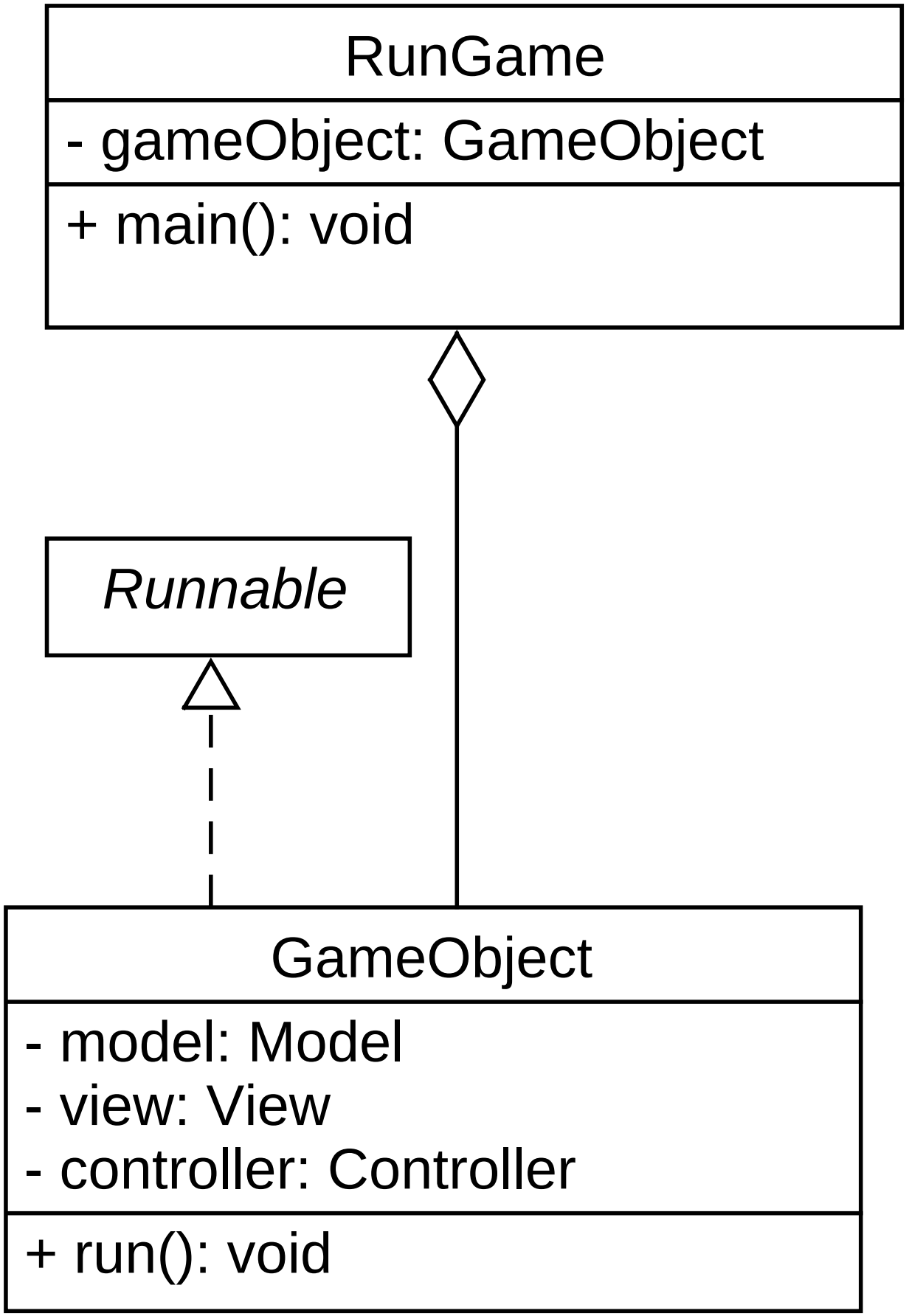
AirPassableTile
+isPassable(Entity): boolean
+touch(Entity): void

ItemMap
-items: Collection<Location, Item>
+add(item, location)
+touch(entity, location): void // adds item if it can and removes it from map

Goes off to Item


«interface»
StateMachine<GameState>
Model
- keyPreferences: KeyPreferences
- eventManager: EventManager
- entityManager: EntityManager
- gameMap: GameMap
- itemMap: ItemMap
- projectileManager: ProjectileManager
- triggerManager: TriggerManager
+ moveUp(): void
+ moveUpLeft(): void
+ moveDownLeft(): void
+ moveDown(): void
+ moveDownRight(): void
+ moveUpRight(): void
+ save(): StructuredMap
+ load(StructuredMap): StructuredMap
+ push(GameState): void
+ pop(): void
+ swap(GameState): void
+ update(): void


«Enum»
Angle
UP_RIGHT : 45
UP : 90
UP_LEFT : 135
DOWN_LEFT : 225
DOWN : 270
DOWN_RIGHT : 315
-theta : int
+getAngle() : int
+sin() : double
+cos() : double
1
Projectile
-location: Location
-speed: double
+translate()
1
1
«interface»
Trigger
-event
-area: Area
+handle(Entity): void
+hasExpired() : boolean
Location
-x: double
-y: double


RunGame
- gameObject: GameObject
+ main(): void
Runnable
GameObject
- model: Model
- view: View
- controller: Controller
+ run(): void

State Machinery

StateMachine
T extends State
-states:stack<T>
+push(T): void
+swap(T): void
+pop(): void

«Interface»
State
+onEnter(): void
+onPause(): void
+onResume(): void
+onExit(): void

GameStates

{{abstract}}
GameState
+onEnter(): void //initialize and attach a view layout object
+onPause(): void
+onResume(): void
+onExit(): void
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

MainMenuState
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

CharacterSelectionState
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

LoadGameMenuState
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

SaveGameMenuState
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

OptionsMenuState
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

GamePlayState
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

PauseMenuState
+onEnter(): void //overrides normal view behavior to overlay instead
+onExit(): void //removes popup layer
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

InventoryMenuState
+onEnter(): void //overrides normal view behavior to overlay instead
+onExit(): void //removes popup layer
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

TradeMenuState
+onEnter(): void //overrides normal view behavior to overlay instead
+onExit(): void //removes popup layer
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

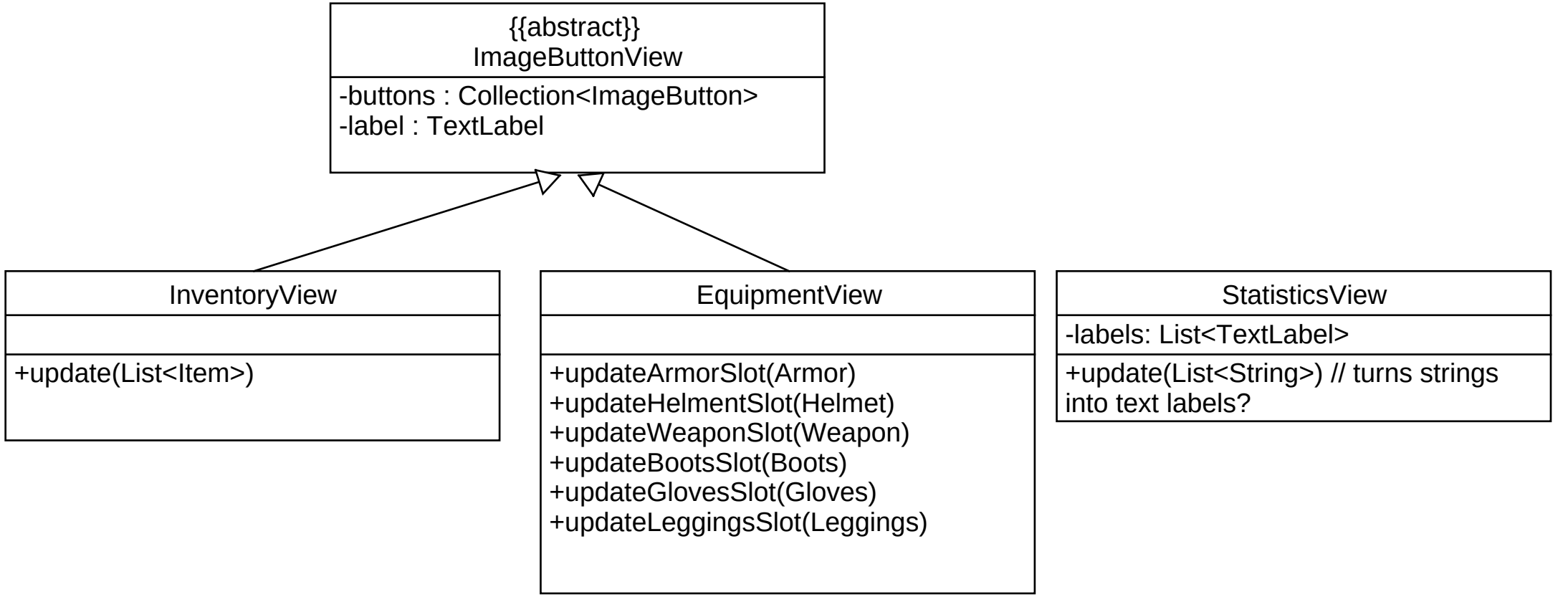
SkillsMenuState
+onEnter(): void //overrides normal view behavior to overlay instead
+onExit(): void //removes popup layer
#getViewLayout(): ViewLayout / (JPanel?), {{Abstract}}
#getController(): (Controller??) {{Abstract}}

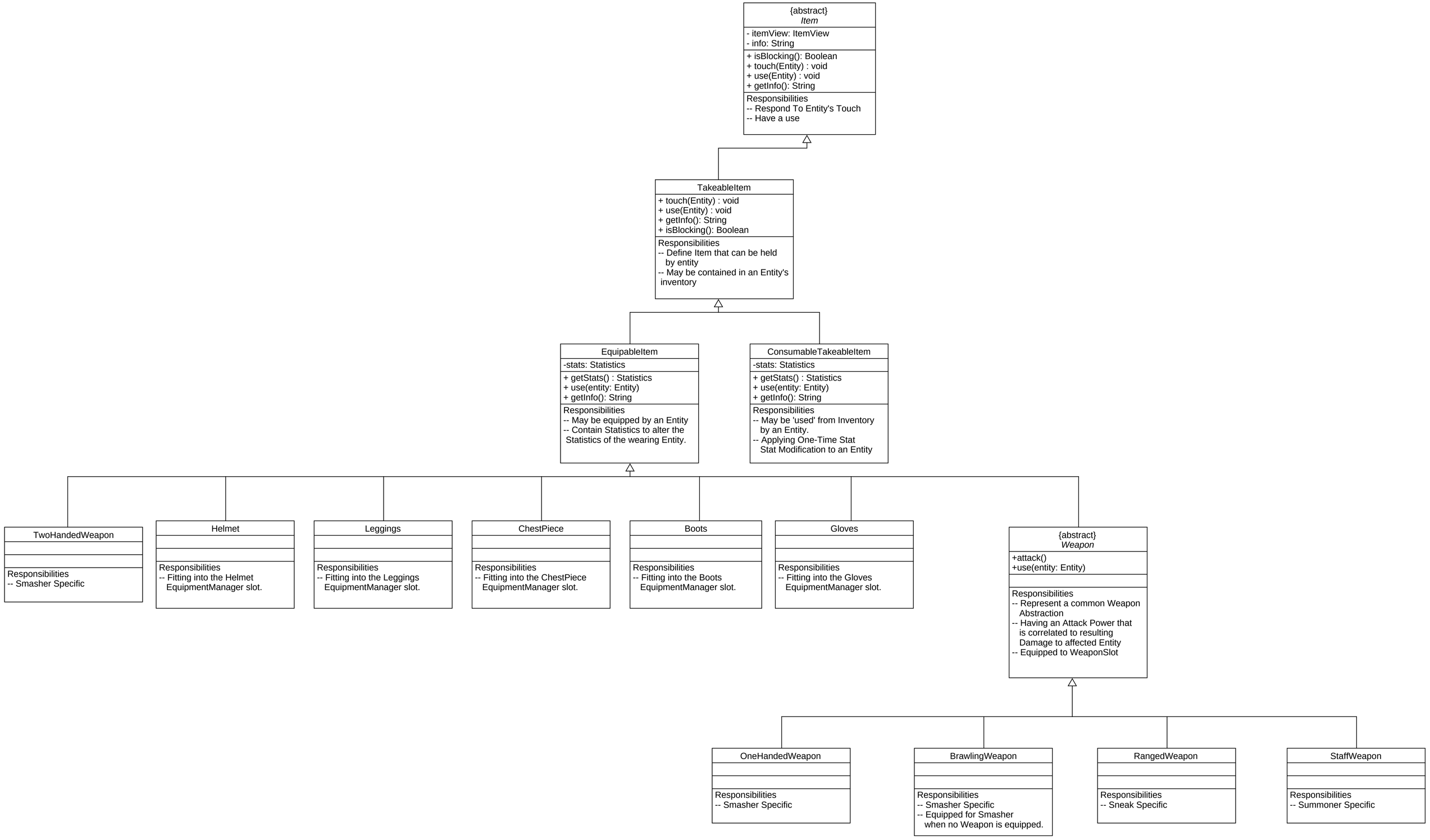
NPCStates

{{abstract}}
NPCState
+onEnter(): void
+onPause(): void
+onResume(): void
+onExit(): void
+Entity.interact(Entity) //explicit interaction
+Entity.onEnterObservationArea(Entity) //implicit interaction. the npc “sees” us

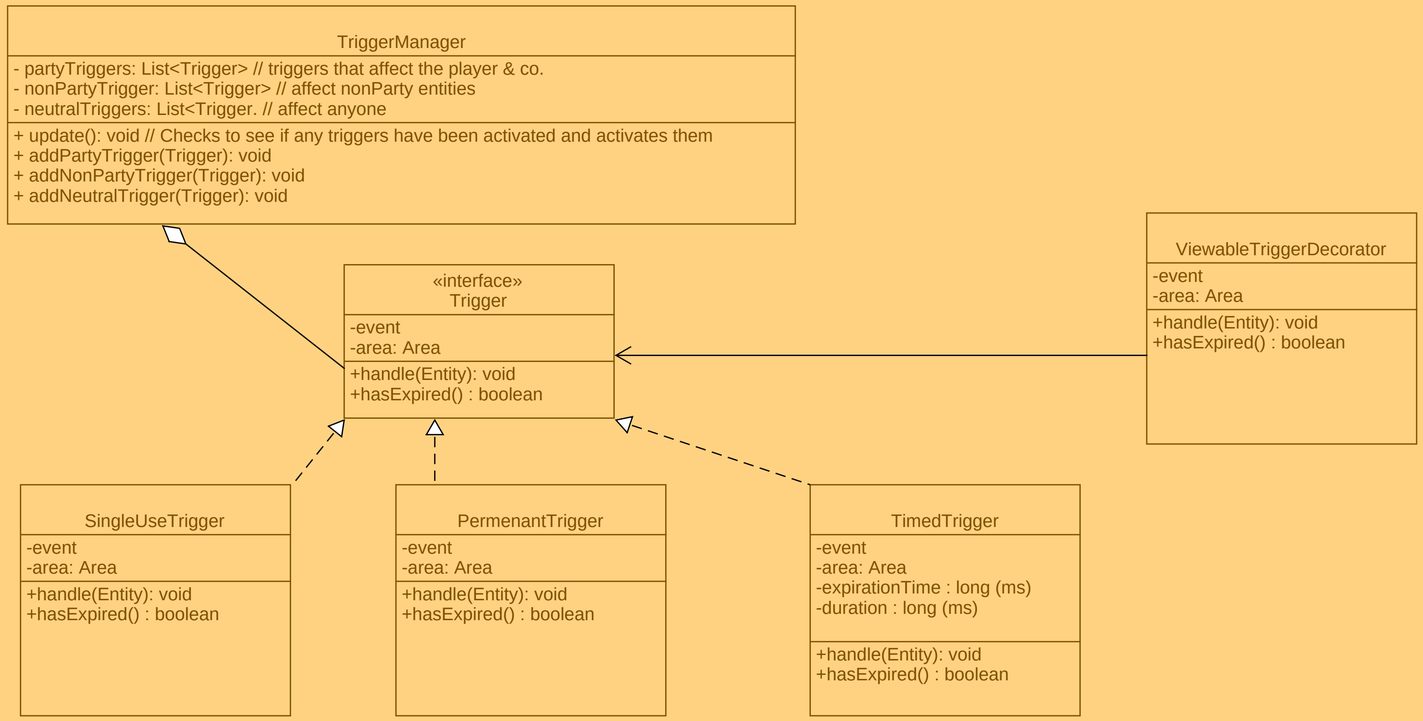
AvatarStates

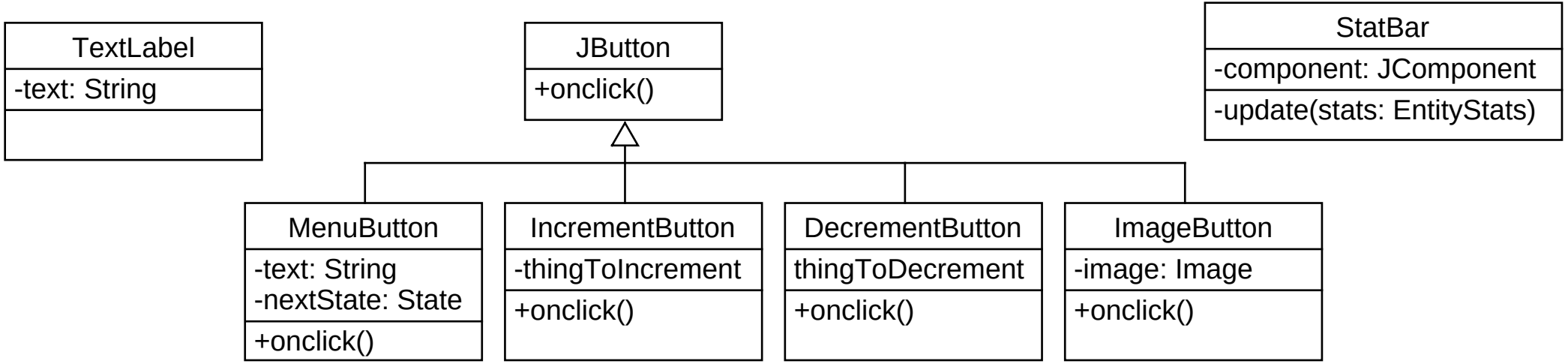
{{abstract}}
AvatarState
+onEnter(): void
+onPause(): void
+onResume(): void
+onExit(): void
+Avatar.interact(Entity) //explicit interaction

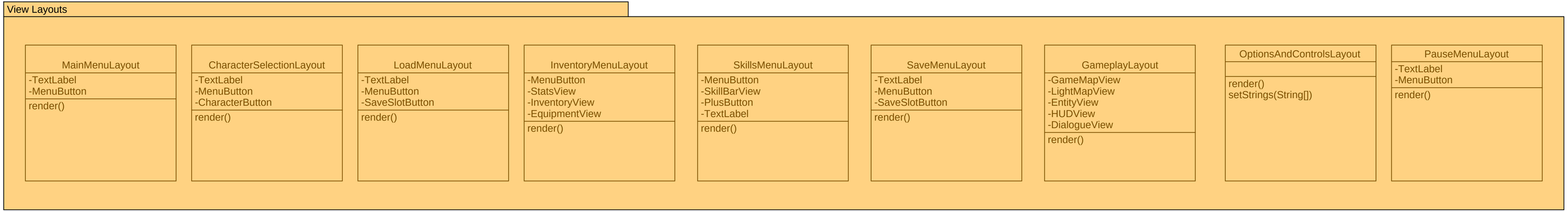

{{abstract}}
ImageButtonView
-buttons : Collection<ImageButton>
-label : TextLabel
InventoryView
+update(List<Item>)
EquipmentView
+updateArmorSlot(Armor)
+updateHelmentSlot(Helmet)
+updateWeaponSlot(Weapon)
+updateBootsSlot(Boots)
+updateGlovesSlot(Gloves)
+updateLeggingsSlot(Leggings)
StatisticsView
-labels: List<TextLabel>
+update(List<String>) // turns strings into text labels?

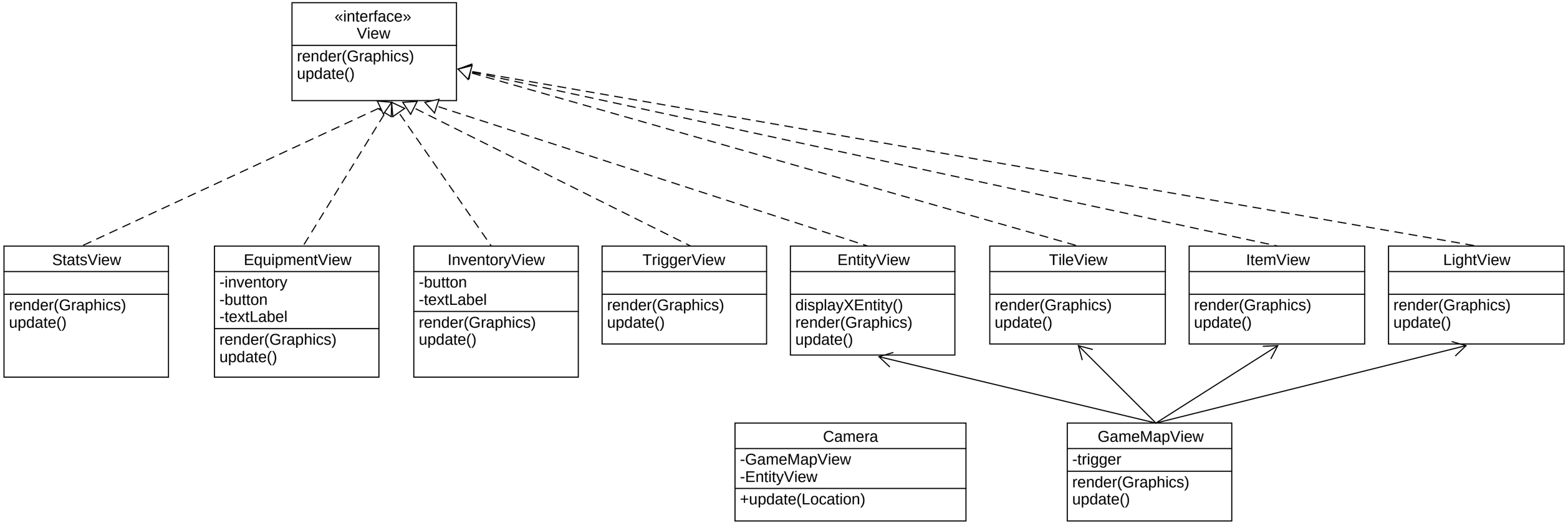

{abstract}
Item
- itemView: ItemView
- info: String
+ isBlocking(): Boolean
+ touch(Entity) : void
+ use(Entity) : void
+ getInfo(): String
Responsibilities
-- Respond To Entity's Touch
-- Have a use
TakeableItem
+ touch(Entity) : void
+ use(Entity) : void
+ getInfo(): String
+ isBlocking(): Boolean
Responsibilities
-- Define Item that can be held by entity
-- May be contained in an Entity's inventory
EquipableItem
-stats: Statistics
+ getStats() : Statistics
+ use(entity: Entity)
+ getInfo(): String
Responsibilities
-- May be equipped by an Entity
-- Contain Statistics to alter the Statistics of the wearing Entity.
ConsumableTakeableItem
-stats: Statistics
+ getStats() : Statistics
+ use(entity: Entity)
+ getInfo(): String
Responsibilities
-- May be 'used' from Inventory by an Entity.
-- Applying One-Time Stat Stat Modification to an Entity
TwoHandedWeapon
Responsibilities
-- Smasher Specific
Helmet
Responsibilities
-- Fitting into the Helmet EquipmentManager slot.
Leggings
Responsibilities
-- Fitting into the Leggings EquipmentManager slot.
ChestPiece
Responsibilities
-- Fitting into the ChestPiece EquipmentManager slot.
Boots
Responsibilities
-- Fitting into the Boots EquipmentManager slot.
Gloves
Responsibilities
-- Fitting into the Gloves EquipmentManager slot.
{abstract}
Weapon
+attack()
+use(entity: Entity)
Responsibilities
-- Represent a common Weapon Abstraction
-- Having an Attack Power that is correlated to resulting Damage to affected Entity
-- Equipped to WeaponSlot
OneHandedWeapon
Responsibilities
-- Smasher Specific
BrawlingWeapon
Responsibilities
-- Smasher Specific
-- Equipped for Smasher when no Weapon is equipped.
RangedWeapon
Responsibilities
-- Sneak Specific
StaffWeapon
Responsibilities
-- Summoner Specific

Trigger


TriggerManager
- partyTriggers: List<Trigger> // triggers that affect the player & co.
- nonPartyTrigger: List<Trigger> // affect nonParty entities
- neutralTriggers: List<Trigger. // affect anyone
+ update(): void // Checks to see if any triggers have been activated and activates them
+ addPartyTrigger(Trigger): void
+ addNonPartyTrigger(Trigger): void
+ addNeutralTrigger(Trigger): void
«interface»
Trigger
-event
-area: Area
+handle(Entity): void
+hasExpired() : boolean
ViewableTriggerDecorator
-event
-area: Area
+handle(Entity): void
+hasExpired() : boolean
SingleUseTrigger
-event
-area: Area
+handle(Entity): void
+hasExpired() : boolean
PermenantTrigger
-event
-area: Area
+handle(Entity): void
+hasExpired() : boolean
TimedTrigger
-event
-area: Area
-expirationTime : long (ms)
-duration : long (ms)
+handle(Entity): void
+hasExpired() : boolean


TextLabel
-text: String
JButton
+onclick()
StatBar
-component: JComponent
-update(stats: EntityStats)
MenuButton
-text: String
-nextState: State
+onclick()
IncrementButton
-thingToIncrement
+onclick()
DecrementButton
thingToDecrement
+onclick()
ImageButton
-image: Image
+onclick()


View Layouts
MainMenuLayout
-TextLabel
-MenuButton
render()
CharacterSelectionLayout
-TextLabel
-MenuButton
-CharacterButton
render()
LoadMenuLayout
-TextLabel
-MenuButton
-SaveSlotButton
render()
InventoryMenuLayout
-MenuButton
-StatsView
-InventoryView
-EquipmentView
render()
SkillsMenuLayout
-MenuButton
-SkillBarView
-PlusButton
-TextLabel
render()
SaveMenuLayout
-TextLabel
-MenuButton
-SaveSlotButton
render()
GameplayLayout
-GameMapView
-LightMapView
-EntityView
-HUDView
-DialogueView
render()
OptionsAndControlsLayout
render()
setStrings(String[])
PauseMenuLayout
-TextLabel
-MenuButton
render()


«interface»
View
render(Graphics)
update()
StatsView
render(Graphics)
update()
EquipmentView
-inventory
-button
-textLabel
render(Graphics)
update()
InventoryView
-button
-textLabel
render(Graphics)
update()
TriggerView
render(Graphics)
update()
EntityView
displayXEntity()
render(Graphics)
update()
TileView
render(Graphics)
update()
ItemView
render(Graphics)
update()
LightView
render(Graphics)
update()
Camera
-GameMapView
-EntityView
+update(Location)
GameMapView
-trigger
render(Graphics)
update()