attack: AttackAction actions: Actions getAllowableActions(actor:Actor, direction:String, map:GameMap) <<create>> UML Sequence Diagram for Dinosaur's drinking water add(new AttackAction(this)): void \_\_\_\_ <u>GameMap</u> <u>Ground</u> <u>Location</u> [Item item : actor.getInventory()] UML Sequence Diagram for Feeding actors :getAction(actor:Actor, map:GameMap): Action [item instanceof Feedable] locationOf(actor): Location\_\_\_\_ | | [this.hasCapability(DinosaurCapability.CARNIV $\phi$ RE)] && item.hasCapability(DinosaurCapability.CARNIVORE\_FEEDING locationOf(actor) <u>: MealKit</u> <u>item :PortableITem</u> actor : Actor add(feed: FeedAction): boolean **▼** - - - true getWater(): Location execute(actor:Actor, map:GameMap):String ((Dinosaur) target).addFood(item.addingFood):int if(water != null) [this.hasCapability(DinosaurCapability.HERBIVORE)] && item.hasCapability(DinosaurCapability.HERBIVORE\_FEEDING item.addingFood(); int if(water.getGround().hasCapability(Abilities.LAKE)) add(feed: FeedAction): boolean water.getGround(): Ground \_\_\_\_\_\_true [item.hasCapability(Abilities.MEAL\_KIT)] [target.has capability(DinosaurCapability.STEGOSAUR)] addFood<mark>(getAddFood(target)): int</mark> return max FL add(attack: AttackAction): boolean alt if(distance(current, water) > 1) [target.hascapability(DinosaurCapability.BRACHIOSAUR)] addFood(getAddFood(target)): int addFood(getAddFood(tar<mark>get)): void</mark> distance(current, water): int return max FL Loop
[Exit exit : current.getExits()] alt

if(nearest == null || distance(exit.getDestination(), water) < distance(nearest, water) [target.hasCapability(DinosaurCapability.ALLOSAUR)] exit.getDest|nation(): Location nearestLocation UML Sequence Diagram for Dinosaur's eating items [target.hasCapability(DinosaurCapability.PTERODACTYL)] addFood(getAddFood(target)): int addFood(getAddFood(target)): void actor:Actor :Location execute(actor:Actor, map:GameMap):String (!exitedUsed.getDestination().containsAnActor() && exitUsed.getDestination().canActorEnter(actor) ((PortableItem) item).addingFood(): int [target.hasCapabi|ity(pinosaurCapability.PTERODACTYL) && item.hasCapability(DinosaurCapability.CORPSE)] addFood(10): void decrementFoodLevel(10): void ((Dinosaur) actor).addFood(addFL): void fedDino\$aurPoints() void map.locationOf(actor).removeItem(item): void alt [corpse.addFood == 0] [distance(current, water) == 1 && currentLake.hasCapability(Abilities.HAS\_WATER)] **4**-----[pteroFood == False | [item instanceof ∯ru fedDinosaurPoints(): void UML Sequence Diagram for harvesting fruit on a bush or a tree **\***----tree: Tree <u>bush : Bush</u> execute(actor, map) GenerateRandom.randomPossibilities(HARVEST\_UPPER\_BOUND) return chance [(bush != null) && bush.hasCapability(Abilities.HAS\_FRUIT)] UML Sequence Diagram for producing ripe fruit on a tree fruits.get(HARVESTED\_FRUIT\_INDEX) tree : Tree <u>player : Player</u> ecoPoint : EcoPoint produceRipeFruit(treeLocation) [(tree !- null) && tree.hasCapability(Abilities.HAS\_FRUIT)] tree.toBeHarvested() GenerateRandom.randomPossibilities(UPPER\_BOUND\_PRODUCE) fruits.get(HARVESTED\_FRUIT\_INDEX) chance == SUCCESS\_PRODUCE] fruits.add(new Fruit("Fruit")) mapPlayer.trackPlayer() ripeFruitTreePoints() actor.addItemToInventory(this.fruit) UML Sequence Diagram for Dinosaur Death (from unconsciousness) player.getEcoPoints() fruitHarvestedPoints() <u>dinosaur : Dinosaur</u> <u>location : Location</u> playTurn(actions, lastAction, map) return menuDescription(actor) return "You search the tree or bush for fruit, but you can't find any ripe ones." [getFoodLevel() > 0] UML Sequence Diagram for Lake Life Cycle (Raining and Fish Born Mechanism) [getWaterLevel() <= 0 || getFoodLevel() <= 0] this.incrementUnconsciousCounter() <u>lake : Lake</u> <u>fish : ArrayList<Fish></u> locationOf(this) tick(lakeLocation) return location [this.getUnconsciousCounter() > 15 && waterLevel == 0] [lakeLocation.x()  $\geq$  0 && lakeLocation.y()  $\geq$  0] [this.hasCapability(DinosaurCapability.STEGOSAUR)] removeActor(this) | Weather.isRain()] addItem(new StegosaurCorpse("Stegosaur Corpse")) removeActor(this) addItem(new AllosaurCorpse("Allosaur Corpse")) [fish.size() < MAXIMUM\_FISH] randomPossibilities(UPPER\_BOUND\_BORN) [this instanceof Pterodactyl] removeActor(this) addItem(new PterodactylCorpse("Pterodactyl Corpse")) [chance < SUCCESS\_BORN] removeActor(this) addItem(new BrachiosaurCorpse("Brachiosaur Corpse")) [this.getUnconsciousCounter() >= 20] this hasCapability(DinousaurCapability.STEGOSAUR)] removeActor(this) [turns % 10 == 1] addItem(new StegosaurCorpse("Stegosaur Corpse")) isRain = False addItem(new AllosaurCorpse("Allosaur Corpse")) [this hasCapability(DinosaurCapability.PTERODACTYL)] firstGameMap(lakeLocation) removeActor(this) additem(new PterodactylCorpse("Pterodactyl Corpse")) [(akeLocation.x() == 0 && lakeLocation.y() == 0) && (turns = counter \* 10) && Application.mapping.firstGameMap(lakeLocation)] [this.getUnconsciousCounter() >= 15] this hasCapability(DinousaurCapability.BRACHIOSAUR)] removeActor(this) addItem(new BrachiosaurCorpse("Brachiosaur Corpse")) return Action UML Sequence Diagram for More Sophisticated Game Driver and Second Map <u>world :</u> AdvancedWorld <u>firstGameMap :</u> secondGameMap : <u>mapping:</u> <u>groundFactory :</u> <u>player : Actor</u> <u>: Stegosaur</u> <u>: Display</u> : Dirt <u>: Bush</u> <u>: Lake</u> <u>: Pterodactyl</u> <u>: Wall</u> : Floor : Tree : VendingMachine <u>GameMap</u> <u>GameMap</u> <u>Mapping</u> <u>FancyGroundFactory</u> main(String[] args) <<create>> <<create>> return Display return AdvancedWorld <<create>> <<create>> return Dirt <<create>> return Wall <<create>> return Floor <<create>> return Tree <<create>> return VendingMachine return Bush <<create>> return Lake return FancyGroundFactory <<create>> return firstGameMap addGameMap(firstGameMap) addGameMap(secondGameMap) getGameMaps().get(0) getGameMaps().get(1) return player addPlayer(player, firstGameMap.at(20, 4)) at(20, 6) <<create>> return Pterodactyl at(28, 19) return Stegosaur addActor(Stegosaur) return movesAndPoints [movesAndPoints != null] runChallenge(specMoves, specEcoPoints) 

UML Sequence Diagram for Dinosaur's allowable actions