Design Rationale

Author: Runzhe Hua & Yue Ling
Date: 10/5/2020

Problem Statement

Our initial Zombie game is missing a lot of desired functionality, which is boring and needs new features.

Solution & Explanations

There are lots of Behaviors and Actions need in our extended system. They are inherited from the interface `Behavior` and the abstract class `Action`. All these Behaviors are dependent on an Action.

We need add a new class called `Farmer` which inherited from `Human`. `Farmer` has more `Behavior`s such as `SownBehaviour`, `HarvestBehaviour` and `FertilizeBehaviour`. That will create `Crop` and `Food`. `Crop` is inherited from `Ground` like `Tree`. And the food is an `Item` can be used in `EatAction`. `Human` will call the `RisingFromDeath` every palyTurn, which will check whether the `Human` is died and start a counter. Create a new `Zombie` in the same place after 5-10 turns.

We also need to add some features to Zombies. We will add bite to the `getIntrinsicWeapon()` `method` in the Zombies class. Because bite is an attack by `Intrinsic Weapon. Using rand.nextBoolean() to` make sure there are 50% probability of using this instead of their normal attack. In the `AttackAction` class, we will add code to implement the lower chance of hitting and using the heal method after hitting. Moreover, we added `PickItemBehavior` and `PickItemAction` class to let zombie pick weapons to make more damages.

Sometimes the Zombies will do nothing and saying "Braaaaains" which possibility is 10%. We will add code about these in the beginning of the `playTurn()`method.

Each Zombies have counters about their existing legs and arms. And check the number of them in the `playTurn()` method to implement the function of "Beating up the Zombies" If some part of the Zombie dropped, it will become a `PortableItem`. Player can use it by crafting weapons.

A `RisingFromDead` class associated with `Human` and `Zombie` clears a human object and create a new zombie object.

The `CraftingWeapons` is a class inherited from `Action` which can create several `WeponItem`.

Decisions

1. In UML diagrams, we created lots of one-to-one relationships, which means each behavior is mandatory and only for one actor, one action is for one behavior, one zombie can only pick one item(weapon), etc. The reason is to limit the function. (later coding will test if "1" is a valid limit for harvesting crop and producing food)
2. For relationships around crop, we decided to use associations instead of dependencies (in

version_0.2), making crop an attribute (expected arrayList) in behavior classes, which reduces dependencies.

3. The farmer class we made is inherited from human class, same as player class. So, farmer and player share the same characteristics and abilities as Human, which also keeps our code DRY.

4. When we adding bite attack for Zombie, we chose making it into getIntrinsicWeapon() method in Zombie class rather than creating a new biteBehavior class. The reason is that we put it inside a method to decrease the scope without making dependent classes.