Design patterns (inf-335) final project presentation

Hero vs. Zombies
Nurmangaliyev Dias - 190103485

About the project

Game goal ?

Hero should kill all zombies and stay alive till the end of the game.

What player can do?

Decide how to upgrade the Hero at each Round.

GUI ?

No GUI.

Just logs.

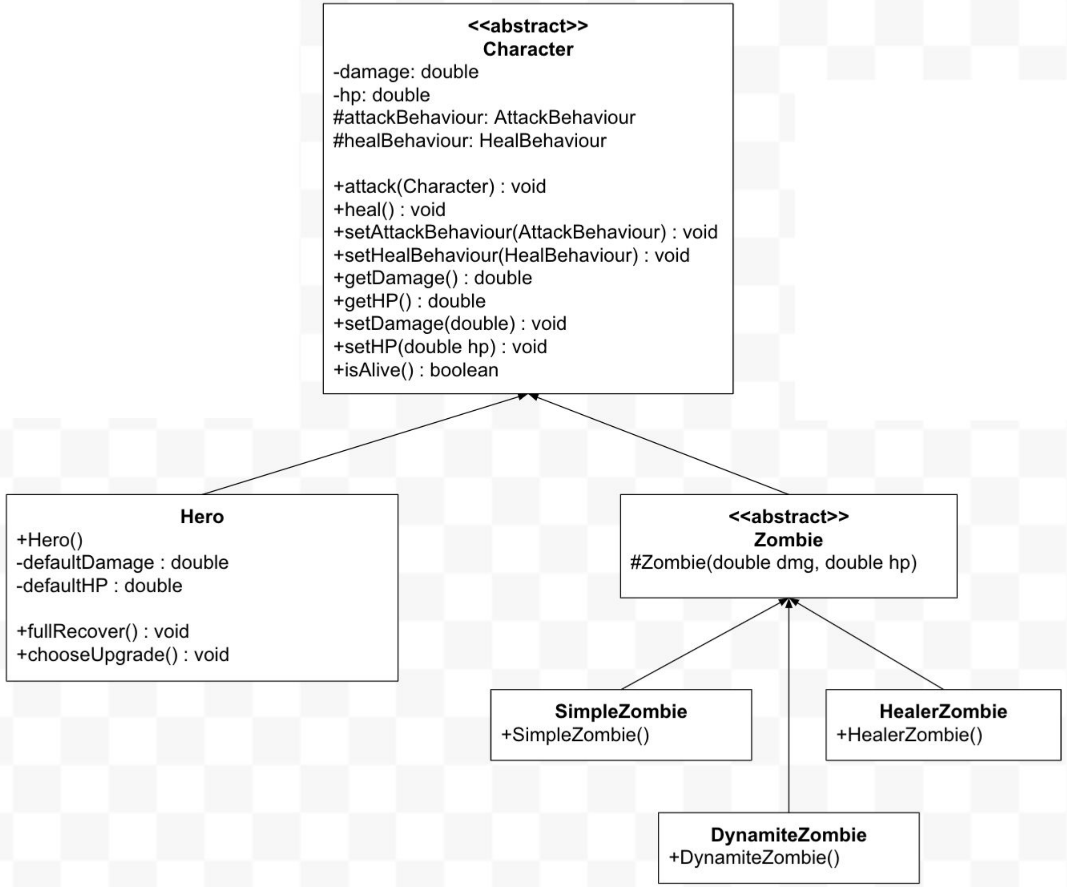
Stupidly simple.

Rounds and Battles

Hero must stay alive N **Rounds** to win the game.
Each round:

- M random **zombies** are **created**.
- Hero **fights** individually against each Zombie in **Battle**.
- Player can choose to **upgrade** either its attack or heal behaviours.
- Hero **recovers** HP to the default value for the next round.

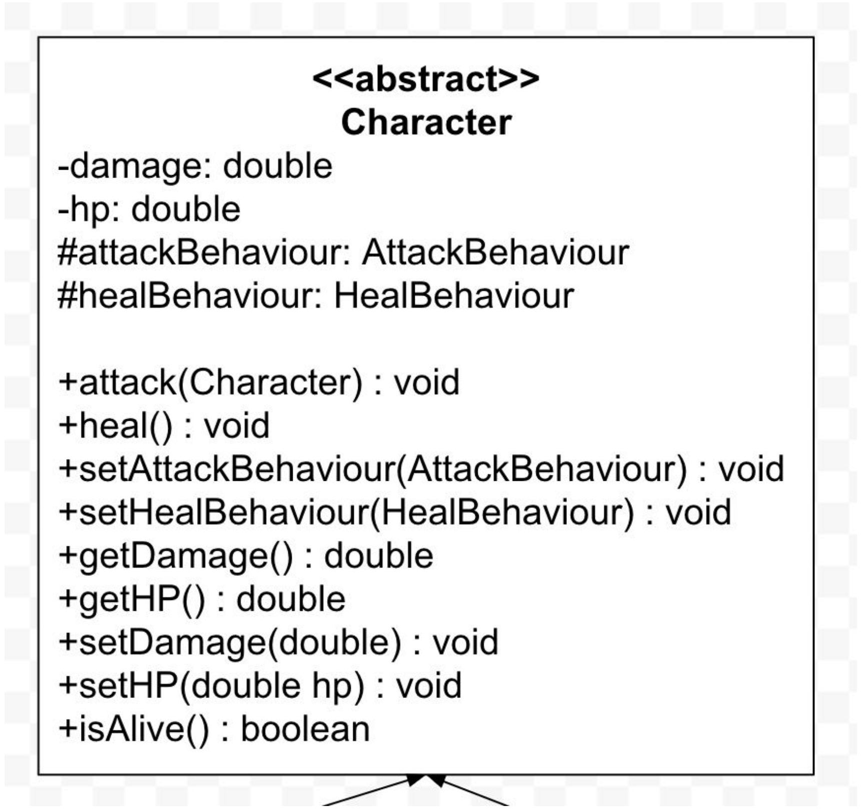
Character
Hero
Zombie(s)



Character

Both **Hero** and **Zombie** extends Character class.

Parameters:



```
<<abstract>>
Character
-damage: double
-hp: double
#attackBehaviour: AttackBehaviour
#healBehaviour: HealBehaviour

+attack(Character) : void
+heal() : void
+setAttackBehaviour(AttackBehaviour) : void
+setHealBehaviour(HealBehaviour) : void
+getDamage() : double
+getHP() : double
+setDamage(double) : void
+setHP(double hp) : void
+isAlive() : boolean
```

Character

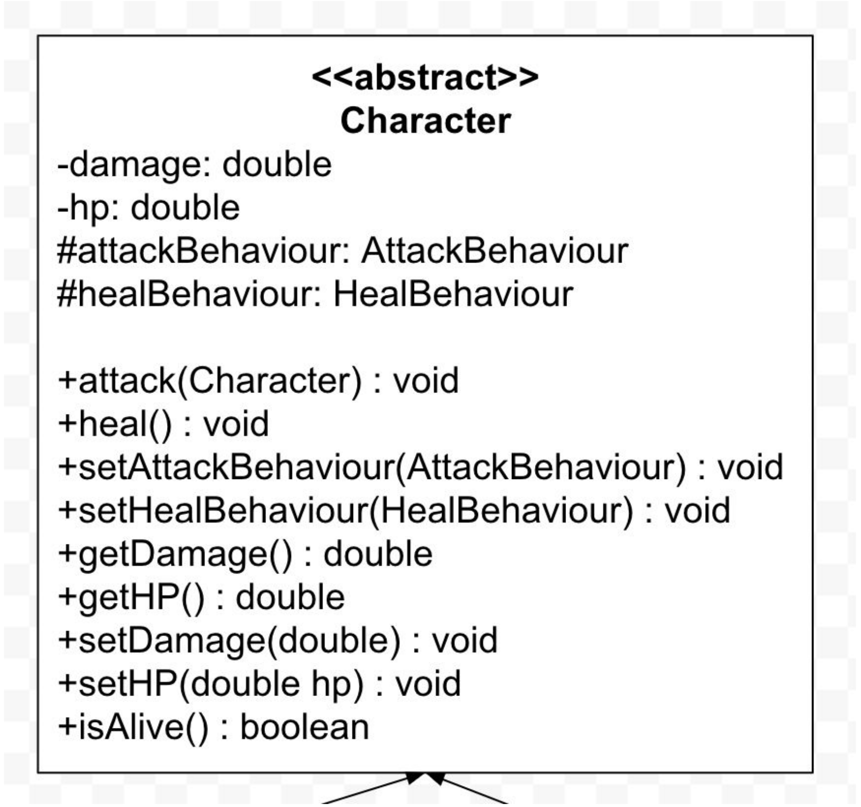
Both **Hero** and **Zombie** extends Character class.

Parameters:

- **basic**: damage and hp.
- **strategies**: attackBehaviour and healBehaviour **interfaces**.

Methods:

- **attack**() and **heal**().
- accessors and mutators.



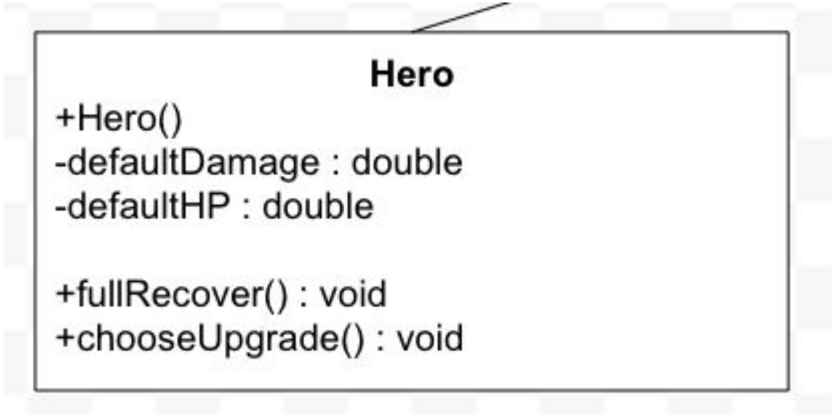
```
<<abstract>>
Character
-damage: double
-hp: double
#attackBehaviour: AttackBehaviour
#healBehaviour: HealBehaviour

+attack(Character) : void
+heal() : void
+setAttackBehaviour(AttackBehaviour) : void
+setHealBehaviour(HealBehaviour) : void
+getDamage() : double
+getHP() : double
+setDamage(double) : void
+setHP(double hp) : void
+isAlive() : boolean
```

Hero

Hero extends **Character**.

Methods:

- **fullRecover:** set HP to 100%.
- **chooseUpgrade:**
    - 1) wrap **attackBehaviour** with randomly created **weapon** decorator**.**
    - 2) increase the default HP.

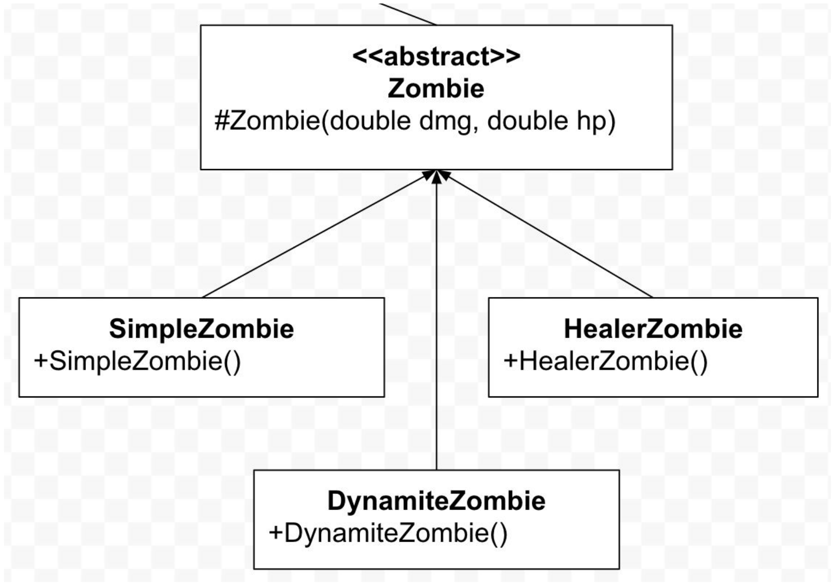


Zombie(s)

Zombie extends **Character**.

Provides the constructor for concrete subclasses.

Each **ConcreteZombie** has its own basic parameters and behaviours:



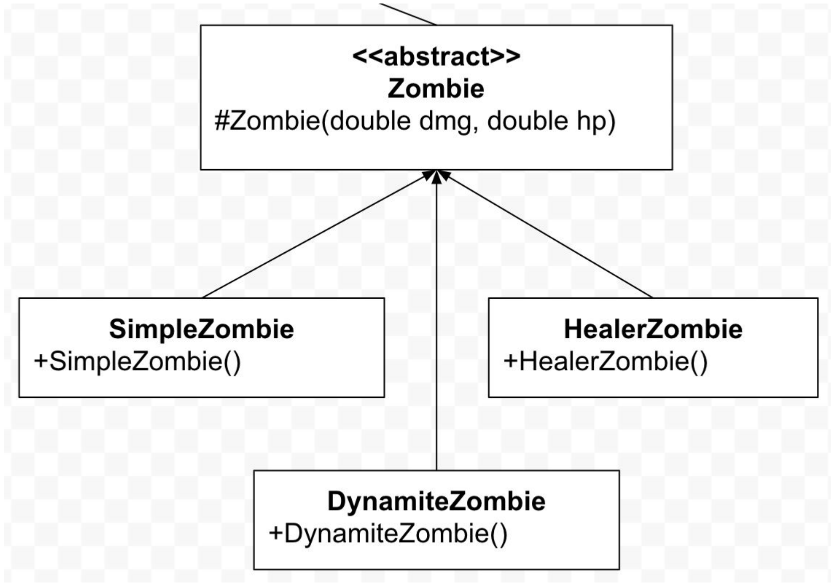
Zombie(s)

Zombie extends **Character**.

Provides the constructor for concrete subclasses.

Each **ConcreteZombie** has its own basic parameters and behaviours:

- **SimpleZombie:** has SimpleAttackBehaviour and NoHealBehaviour.
- **HealerZombie:** can heal itself with some probability.
- **DynamiteZombie:** uses DynamiteDecorator, can cause a high value damage with some probability, but only once.



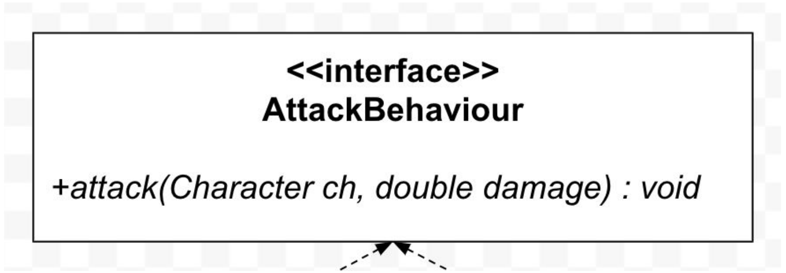
Patterns used

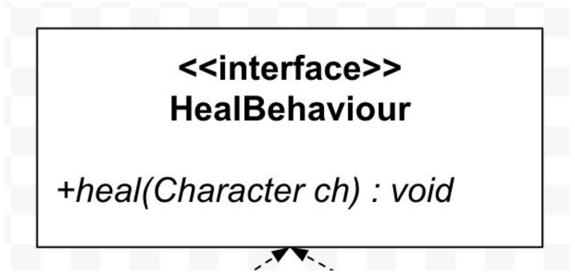
Strategies

AttackBehaviour and **HealBehaviour:** interfaces that is used as **strategies** for Character class.

They define the behaviour of the **context** - Character.

We can create concrete implementations and use them **interchangeably**.



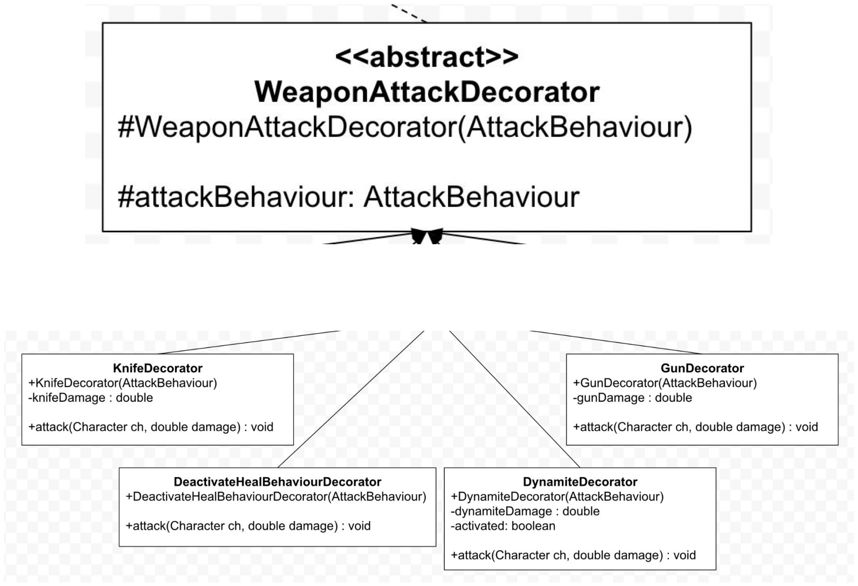


Decorator

WeaponAttackDecorator:

- implements AttackBehaviour.
- has an instance of another AttackBehaviour.

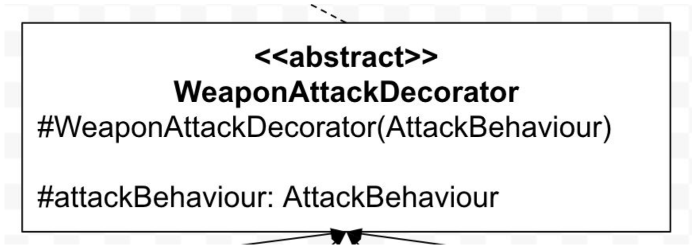
Concrete weapond decorators implement the **attack(Character ch, double damage)** method.

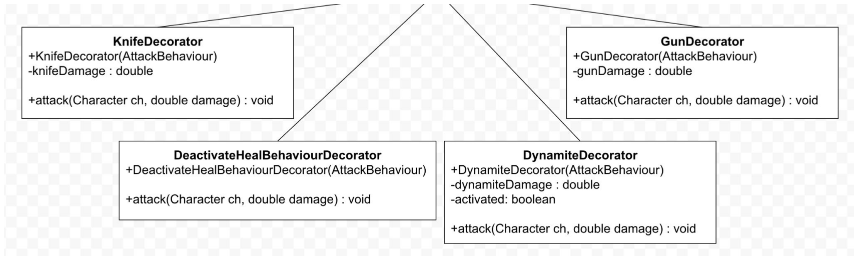


<<abstract>>
WeaponAttackDecorator
#WeaponAttackDecorator(AttackBehaviour)
#attackBehaviour: AttackBehaviour
KnifeDecorator
+KnifeDecorator(AttackBehaviour)
-knifeDamage : double
+attack(Character ch, double damage) : void
GunDecorator
+GunDecorator(AttackBehaviour)
-gunDamage : double
+attack(Character ch, double damage) : void
DeactivateHealBehaviourDecorator
+DeactivateHealBehaviourDecorator(AttackBehaviour)
+attack(Character ch, double damage) : void
DynamiteDecorator
+DynamiteDecorator(AttackBehaviour)
-dynamiteDamage : double
-activated: boolean
+attack(Character ch, double damage) : void

Decorators

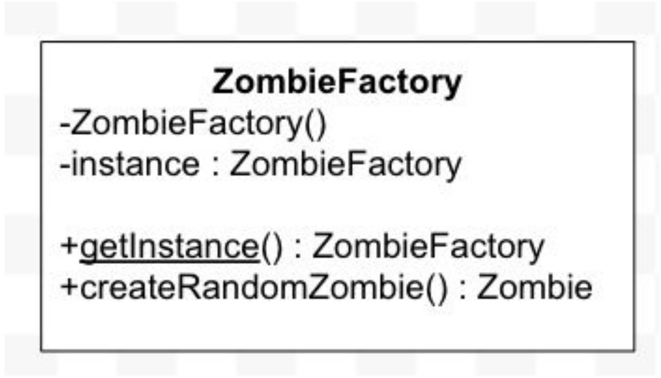
- **KnifeDecorator** - increments the damage for constant value.
- **GunDecorator** - multiplies the characters damage with probability of 20% for a single attack.
- **DynamiteDecorator** - damages the opponent with high value with probability 10%.
- **DeactivateHealBehaviourDecorator** - sets the HealBehaviour of the opponent to NoHealBehaviour.


<<abstract>>
WeaponAttackDecorator
#WeaponAttackDecorator(AttackBehaviour)
#attackBehaviour: AttackBehaviour


KnifeDecorator
+KnifeDecorator(AttackBehaviour)
-knifeDamage : double
+attack(Character ch, double damage) : void
GunDecorator
+GunDecorator(AttackBehaviour)
-gunDamage : double
+attack(Character ch, double damage) : void
DeactivateHealBehaviourDecorator
+DeactivateHealBehaviourDecorator(AttackBehaviour)
+attack(Character ch, double damage) : void
DynamiteDecorator
+DynamiteDecorator(AttackBehaviour)
-dynamiteDamage : double
-activated: boolean
+attack(Character ch, double damage) : void

Factory method & Singleton

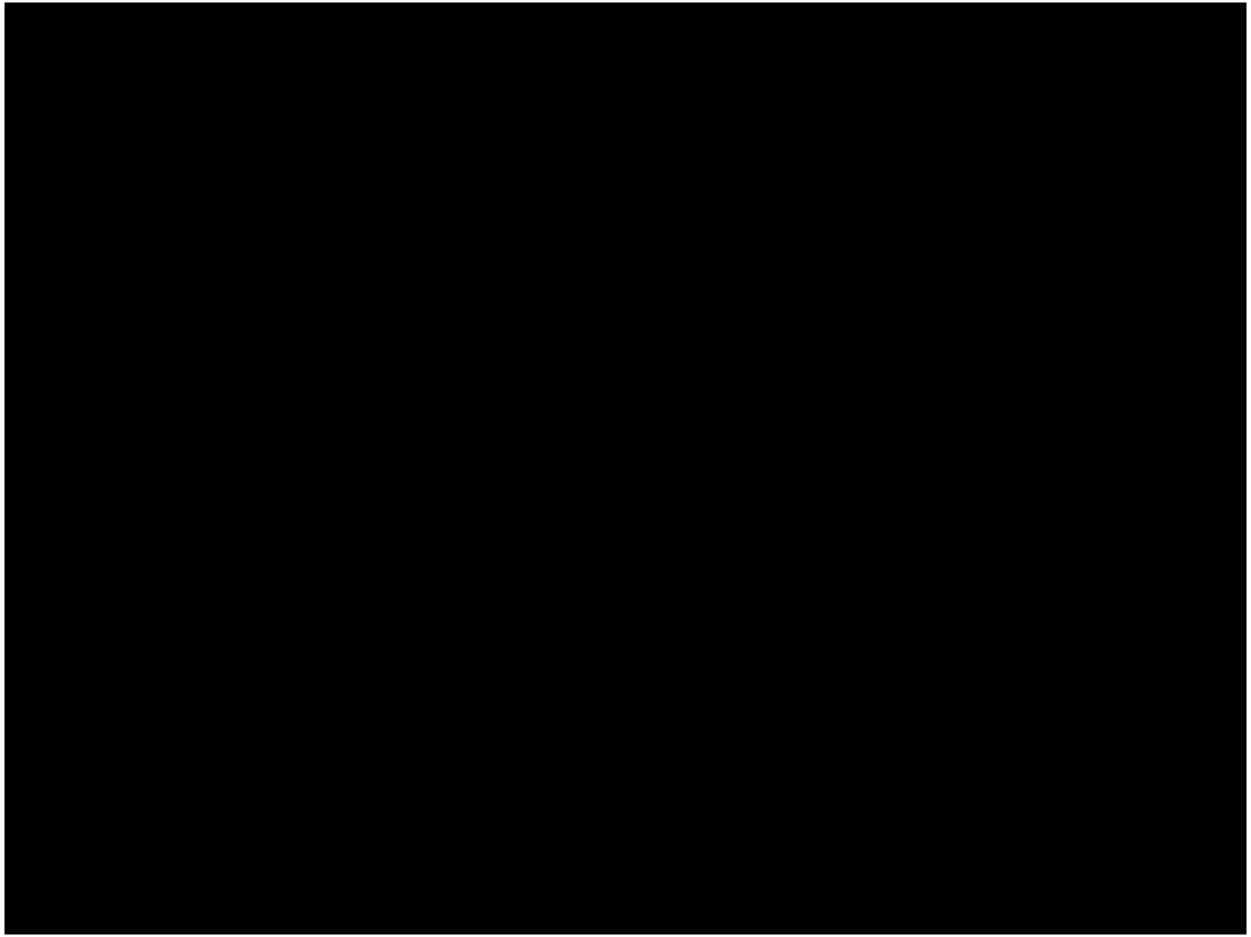
- ZombieFactory used as a **factory** to create random Zombie instances.
- The same instance is lazily created and returned every time we call **getInstance().**


ZombieFactory
-ZombieFactory()
-instance : ZombieFactory
+getInstance() : ZombieFactory
+createRandomZombie() : Zombie

Patterns used

1. Character has AttackBehaviour and HealBehaviour interfaces as a class parameter: **Strategy** pattern.
2. AttackBehaviour can be upgraded using WeaponAttackDecorator: **Decorator** pattern.
3. Creation of Zombies delegated to ZombieFactory class: **Factory method** pattern.
4. The same instance of ZombieFactory is used on every round: **Singleton** pattern.

Demo



Thanks for attention =)