Voortgezet Programmeren (FEB23007-12)

5. Exercise

Deadline for submission: 2013-02-10 23:59 CET

Instructions

Include in each source file (in class documentation, @author) your names and student numbers. Submit the exercise as a zip file containing _only_ the source files in root of the zip. Submit via Blackboard. Note that incorrectly submitted or non-compiling exercises are automatically awarded 0 points. Remember to document your code with Javadoc annotations.

Exercise

Simulation is a powerful technique for performing what-if analyses. In this exercise, we will be analyzing the dynamics of a zombie invasion. Let us consider a world in which two main types of creatures exist: zombies and humans. Zombies are determined to take over the world by converting humans into zombies, one by one. Most humans are oblivious to this threat. However, their survival instinct kicks in when they realize that zombies are taking over their precious world – whenever there are more zombies than humans, humans can turn into zombie slayers, i.e., humans determined to eliminate the threat of the invading zombies. This process continues until only one main type of creatures populates the world, i.e., zombies or humans.

Write a Java program that simulates this scenario. For making the program in an object-oriented manner, you should implement (at least) the following classes:

- `Creature`: an abstract class for modeling any type of creature. This class should have a constructor that takes the name of the creature as input. Additionally, this class should have an accessor method for the name of the creature. Third, the class needs to have a protected method `say()` that takes a phrase as input and prints the name of the creature, followed by the phrase. Fourth, the `Creature` class should have a public method `act()` that specifies the behavior of the creature, given a `World` (specified below) as input. As this behavior differs per type of creature, this method should be abstract, such that it can be defined for different types of creatures separately. Finally, the `Creature` class should have a public method `die`, modeling the way the different types of creatures die.

- `Zombie`: an extension of `Creature`, specifying zombie-specific behavior. When zombies die, they make a horrible sound. When a `Zombie` performs an action in the `World`, it retrieves all humans from the `World`, and randomly kills one `Human`. This `Human` ceases to exist in the `World` and gets replaced by a zombified version, i.e., a new `Zombie` is spawned in the `World`. A `Zombie` says explicitly what it is doing.

- `Human`: an extension of `Creature`, specifying human-specific behavior. A dying human produces a very loud scream. When a `Human` performs an action in the `World`, it first checks whether there are more zombies than humans. If the humans form a minority, a `Human` becomes a `ZombieSlayer`. A `Human` says explicitly what it is doing.

- `ZombieSlayer`: an extension of `Human`, specifying slayer-specific behavior. When a `ZombieSlayer` performs an action in the `World`, it selects a random number of zombies and kills them. A slayer can kill a maximum number of zombies at once, which may be hard-coded in a public static final variable. A `ZombieSlayer` says explicitly what it is doing.

- `World`: a class for modeling the world inhabited by our creatures. Its constructor should be used to initialize the number of zombies, the number of humans, and the number of zombie slayers. It should also contain methods for retrieving all creatures, all humans, and all zombies. Additionally, it should have methods for spawning and removing creatures. Finally, it should implement a method for getting a random integer. You can use `Random` for this.
- `Engine`: a class that performs a simulation of a zombie invasion of a `World`, which is specified as an input in the constructor. This class should have a method that runs the simulation until either the population of humans or the population of zombies is reduced to 0. Each iteration, a random creature is selected to perform an action. Print the outcome of the simulation.
- `Main`: a main class that creates a world with 1500 humans, 10 zombies, and 0 zombie slayers, an engine and subsequently runs the simulation.

In addition, you have to provide unit testing classes for `Zombie`, `ZombieSlayer`, `Human`, `World`, and `Engine`.