Suggestion for exercise based on D&D

Torben Mogensen

12. december 2016

Resumé

This exercise suggestion is about modelling a very simplified variant of the role-playing game Dungeons & Dragons™.

1 Characters

A *character* describes an adventurer. A character is defined by the following attributes:

- `body : int`, a read-only attribute that is a value between 3 and 18.
- `mind : int`, a read-only attribute that is a value between 3 and 18.
- `level : int`, a read-write attribute that is a non-negative integer.
- `maxHitpoints : int`, a read-write attribute that is a non-negative integer.
- `currentHitpoints : int`, a read-write attribute that is a non-negative integer less than or equal to `maxHitpoints`.

Define a class `Character` with these attributes. It should include a constructor that takes the values of the first two attributes. The remaining three attributes should be set to 0.

The class should implement the following methods:

- `levelUp : unit` that increases the level by one and increases the maximum hit points by the value of the `body` attribute. It also sets the current hit points to equal the maximum hit points.
- `takeDamage : int -> unit` that reduces the current hit points by the amount given, which must be a non-negative integer.
- `healDamage : int -> unit` that increases the current hit points by the amount given, which must be a non-negative integer.
- `isAlive : unit -> bool` that checks whether the current hit points is non-negative.

2 Character Classes

A *character class* describes a special kind of adventurer, so it is a subclass of *character*. You must implement the following character classes:

2.1 Fighter

A fighter is a character that has the following extra attributes and methods:

- A read-write attribute `weapon : Weapon` describing the weapon that the fighter uses, see Section **??**.
- A read-write attribute `armour : Armour` describing the weapon that the fighter is wearing, see Section **??**.

- A method `hit : Character -> unit`, indicating that the fighter tries to hit another character. If the fighter is dead, nothing happens.

  The chance of hitting the opposing character is (10×`level`+`mind`)%, where `level` and `mind` are attributes of the fighter.

  The amount of damage dealt (if successfully hitting the opponent) is `level` + weapon damage, where `level` is the `level` attribute of the fighter and weapon damage is found by calling the `damage` method of the fighter's weapon.

- An overridden method `takeDamage : int -> unit` that reduces the damage taken by the value found by calling the `protect` method of the figher's armour.

A fighter starts with cloth armour and a knife (see Sections **??** and **??**, but must have at least 11 in her `body` attribute. Override the constructor method to ensure this.

2.2 Mage

A *mage* is a magician that employs spells instead of weapons. It has the following additional attributes and methods:

- An attribute `currentMana : int` indicating the amount of mana points that the mage hsa available for spell casting. It is a non-negative integer.
- An attribute `maxMana : int` indicating the maximum amount of mana that points the mage can have.
- A method `cast : Spell -> Character -> unit` that casts a spell (see Section **??**) on the indicated character, subtracting the necessary amount from the mage's mana points. If the mage has less than the indicated mana left, or if the mage is dead, the spell is not cast, and the mana is unmodified.
- A method `rest : unit` that makes the mage rest, regaining his level in mana points, up to his maximum mana. A dead mage can not rest.

A mage starts (at level 0) with 0 mana points and must have at least 11 in `mind`. Override the constructor method to ensure this.

At every level increase, the maximum mana of the mage is set to `level`×`mind`. Override the `levelUp` method to ensure this.

2.3 Your Choice

Define a character class of your own (such as Priest or Rogue). It should use the `body` and `mind` attributes more or less equally.

3 Weapons, Armour and Spells

These are the tools that characters can employ.

3.1 Weapons

A `Weapon` has the following method:

- `damage : unit -> int` that returns the amount of damage that the weapons deals.

You must implement at least the following instances of the `Weapon` class:

- A `Knife` is a weapon that deals 1d8 points of damage, i.e, 1–8 points chosen randomly with equal probability.
- A `Mace` is a weapon that deals 2d4 points of damage, i.e, 1–4 points chosen randomly with equal probability plus another 1–4 points chosen randomly with equal probability.
- One more weapon of your design.

3.2 Armour

A suit of `Armour` has the following method:

- `protect : unit -> int` that returns the amount of damage by which the armour reduces the damage taken when hit.

You must implement at least the following instances of the `Armour` class:

- `Cloth` is a type of armour that subtracts 2 points of damage.
- `Chain` is a type of armour that subtracts 1d4 points of damage, i.e, 1–4 points chosen randomly with equal probability.
- One more type of armour of your design.

3.3 Spells

A mage can employ spells. A spell has the following attributes and methods:

- An attribute `manaCost : integer` is the number of mana points it costs to cast the spell.
- A method `effect : Character -> unit` is the effect that the spell has on the recipient character.

You must implement the following spells:

- `MagicMissile` costs 1 mana point and deals 2 points of damage to the recipient.
- `Heal` costs 2 mana points and heals 4 points of damage to the recipient. The recipient can be the mage herself.
- `Drain` costs 5 mana points and deals 1d6 damage to the recipient. The damage dealt will be added as mana points to the mana pool of the caster.
- And one spell of your own design, for example a spell that enchants the weapon of the recipient (who must be a fighter) to do more damage for a limited number of uses.

4 Random character creation

- Make a function `createCharacter : unit -> Character` that randomly creates a character by using the following steps:

  1. roll 3d6, i.e, find a value equal to adding three random numbers equally distributed in the 1–6 range.
  2. Set `body` to this value and `mind` to 21 minus `body`.
  3. Choose an appropriate character class.

  The character starts at level 0.

- Make a function `createParty : int -> int -> Character list` that creates a party of M level L characters, using the `createCharacter` function and the `levelUp` method of each character.

5 Simulate Battle

Simulate a battle between two parties. A battle consists of a number of rounds in which all characters in both parties take an action each. The order in which characters act is determined randomly in each round. An action is using a method such as `hit`, `cast` or `rest` implemented by the character class.

The battle continues until all characters in one of the parties are dead.