Practice Problem for Section 8

Subscribe for email updates. PINNED

No tags yet. + Add Tag

Sort replies by: Oldest first Newest first Most popular

Pavel Lepin COMMUNITY TA · 11 days ago

Practice Problem for Section 8

A Little Adventure

To practice with the Visitor pattern and OO/functional decomposition a bit, we'll do something similar to the homework problem.

You're given a Standard ML program implementing a small bit of a very simple role-playing engine. This is not an actual game, just something that pits a character (either a knight or a wizard) against a series of various challenges. Since knights and wizards have very different approaches to solving the problems facing them, we need to implement small pieces of logic changing the state of the game depending on the combination of character's type and the type of the challenge. Looks like a perfect match for a Visitor pattern!

Your task is to inspect the provided Standard ML definitions, and reimplement the same logic in Ruby using a principled object-oriented approach. A template file for your solution is also provided, as well as some code responsible for progressing the overall world state.

Note that the double dynamic dispatch is asymmetric in this case.

1 · flag

Pavel Lepin COMMUNITY TA · 11 days ago

`section-8-provided.sml`:

```
(* Provided code for section 8 practice problem (Standard ML) *)

datatype character =
    knight of int * int (* a knight has hitpoints and armor points *)
  | wizard of int * int (* a wizard has hitpoints and mana points *)
    (* whenever a character's hitpoints reach zero -- or become negative --
```

```
       he expires and shuffles off this mortal coil *)

datatype encounter =
    floor_trap of int (* a floor trap hurts anyone walking over it, reducing their hit
points *)
  | monster of int * int (* a monster has attack strength and hitpoints *)
  | potion of int * int (* potions may restore some hitpoints and some mana points (if
 applicable) *)
  | armor of int (* armor pieces may boost a characters armor points (if applicable) *
)

type dungeon = encounter list

fun is_dead character =
    case character of
        knight (hp, _) => hp <= 0
        (* this is something of a dirty hack, as it simplifies the encounter mechanics
 below
           at the cost of wizard state being slightly crazy after a lethal encounter w
ith a
           monster *)
      | wizard (hp, mp) => hp <= 0 orelse mp < 0

fun damage_knight dam (hp, ap) =
    case ap of
        0 => (hp - dam, 0)
      | _ => if dam > ap then damage_knight (dam - ap) (hp, 0) else (hp, ap - dam)

fun play_out_encounter character encounter =
    case (character, encounter) of
        (* knights just walk over traps, grimly accepting their fate *)
        (knight state, floor_trap dam) => knight (damage_knight dam state)
        (* knights take damage from monsters, as their armor hinders their mobility,
           but they are strong enough to take out any monster with a single blow after
wards *)
      | (knight state, monster (dam, _)) => knight (damage_knight dam state)
        (* knights can be healed by potions, but they have no use for mana *)
      | (knight (hp, ap), potion (hp', _)) => knight (hp + hp', ap)
        (* knights just love shiny armor, as it improves their survivability
           and makes them look cool! *)
      | (knight (hp, ap), armor ap') => knight (hp, ap + ap')
        (* wizards can levitate, so floor traps can't harm them... as long as they can
 spend a
           single mana point on the spell *)
      | (wizard (hp, mp), floor_trap dam) => if mp > 0 then wizard (hp, mp - 1) else w
```

```
izard (hp - dam, mp)
        (* wizards can hurl powerful fireballs from great distances... unfortunately,
they
           need mana points equal to the damage dealt to do that, and if a monster get
s close,
           they're toast, as their martial skills are nonexistent *)
      | (wizard (hp, mp), monster (_, hp')) => wizard (hp, mp - hp')
        (* wizards love potions, as they help them all around! *)
      | (wizard (hp, mp), potion (hp', mp')) => wizard (hp + hp', mp + mp')
        (* wizards couldn't care less for armor, as it does them absolutely no good *)
      | (wizard state, armor _) => wizard state

fun resolve_encounter character encounter =
    if is_dead character
        then character (* dead characters have already done all their adventuring... *
)
        else play_out_encounter character encounter

(* produces no side effects, but might be useful for testing *)
val compute_final_outcome = List.foldl (fn (x, y) => resolve_encounter y x)

fun print_char character =
    case character of
        knight (hp, ap) => print ("HP: " ^ Int.toString hp ^ " AP: " ^ Int.toString ap
 ^ "\n")
      | wizard (hp, mp) => print ("HP: " ^ Int.toString hp ^ " MP: " ^ Int.toString mp
 ^ "\n")

fun print_enc encounter =
    case encounter of
        floor_trap dam => print ("A deadly floor trap dealing " ^ Int.toString dam ^
            " point(s) of damage lies ahead!\n")
      | monster (dam, hp) => print ("A horrible monster lurks in the shadows ahead. It
 can attack for " ^
            Int.toString dam ^ " point(s) of damage and has " ^ Int.toString hp ^ " hi
tpoint(s).\n")
      | potion (hp, mp) => print ("There is a potion here that can restore " ^ Int.toS
tring hp ^
            " hitpoint(s) and " ^ Int.toString mp ^ " mana point(s).\n")
      | armor ap => print ("A shiny piece of armor, rated for " ^ Int.toString ap ^
            " AP, is gathering dust in an alcove!\n")

(* tells a story of a given hero trying to storm a dungeon represented as a list of
   sequential encounters *)
fun play_out_adventure character dungeon =
```

```
    if is_dead character
        then (print "Alas, the hero is dead.\nThe adventure ends here.\n"; character)
        else (print_char character;
            case dungeon of
                [] => (print "The hero emerges victorious!\nTheir adventures are over.
..\nFOR NOW.\n"; character)
              | (encounter :: rest_of_the_dungeon) => (print_enc encounter;
                    play_out_adventure (resolve_encounter character encounter) rest_of
_the_dungeon))

(* some heroes and dungeons to try out for your enjoyment *)

val sir_foldalot = knight (15, 3)
val knight_of_lambda_calculus = knight (10, 10)
val sir_pinin_for_the_fjords = knight (0, 15)
val alonzo_the_wise = wizard (3, 50)
val dhuwe_the_unready = wizard (8, 5)

val dungeon_of_mupl = [
    monster (1, 1),
    floor_trap 3,
    monster (5, 3),
    potion (5, 5),
    monster (1, 15),
    armor 10,
    floor_trap 5,
    monster (10, 10)
    ]
val the_dark_castle_of_proglang = [
    potion (3, 3),
    monster (1, 1),
    monster (2, 2),
    monster (4, 4),
    floor_trap 3,
    potion (3, 3),
    monster (4, 4),
    monster (8, 8),
    armor 5,
    monster (3, 5),
    monster (6, 6),
    floor_trap 5
    ]
```

+ Comment

 Pavel Lepin COMMUNITY TA · 11 days ago

`section-8-provided.rb`:

```
## Provided code for section 8 practice problem

## Targets 1.9.3

class Encounter
end

class Output
end

class Stdout < Output
  def print str
    puts str
  end
end

class Null < Output
  def print str
  end
end

class Adventure
  def initialize(out, character, dungeon)
    @out = out
    @init_character = character
    @dungeon = dungeon
  end

  def play_out
    reset

    @dungeon.each do |encounter|
      if @character.is_dead?
        break
      end
      @out.print @character.to_s
      @out.print encounter.to_s
      @character.resolve_encounter encounter
    end
```

```
    if !@character.is_dead?
      @out.print @character.to_s
      @out.print "The hero emerges victorious!\nTheir adventures are over...\nFOR NOW.
"
    else
      @out.print "Alas, the hero is dead.\nThe adventure ends here."
    end

    @character
  end

  private

  def reset
    @character = @init_character
  end
end
```

↑ 0 ↓ · flag

+ Comment

 Pavel Lepin COMMUNITY TA · 11 days ago

`section-8-provided.rb`:

```
## Solution template for section 8 practice problem

## Targets 1.9.3

require_relative './section-8-provided'

class Character
  def initialize hp
    @hp = hp
  end

  def resolve_encounter enc
    if !is_dead?
      play_out_encounter enc
    end
  end

  def is_dead?
```

```
    @hp <= 0
  end

  private

  def play_out_encounter enc
    ## YOUR CODE HERE
  end
end

class Knight < Character
  def initialize(hp, ap)
    super hp
    @ap = ap
  end

  def to_s
    "HP: " + @hp.to_s + " AP: " + @ap.to_s
  end

  ## YOUR CODE HERE
end

class Wizard < Character
  def initialize(hp, mp)
    super hp
    @mp = mp
  end

  def to_s
    "HP: " + @hp.to_s + " MP: " + @mp.to_s
  end

  ## YOUR CODE HERE
end

class FloorTrap < Encounter
  attr_reader :dam

  def initialize dam
    @dam = dam
  end

  def to_s
    "A deadly floor trap dealing " + @dam.to_s + " point(s) of damage lies ahead!"
```

```
  end

  ## YOUR CODE HERE
end

class Monster < Encounter
  attr_reader :dam, :hp

  def initialize(dam, hp)
    @dam = dam
    @hp = hp
  end

  def to_s
    "A horrible monster lurks in the shadows ahead. It can attack for " +
        @dam.to_s + " point(s) of damage and has " +
        @hp.to_s + " hitpoint(s)."
  end

  ## YOUR CODE HERE
end

class Potion < Encounter
  attr_reader :hp, :mp

  def initialize(hp, mp)
    @hp = hp
    @mp = mp
  end

  def to_s
    "There is a potion here that can restore " + @hp.to_s +
        " hitpoint(s) and " + @mp.to_s + " mana point(s)."
  end

  ## YOUR CODE HERE
end

class Armor < Encounter
  attr_reader :ap

  def initialize ap
    @ap = ap
  end
```

```
  def to_s
    "A shiny piece of armor, rated for " + @ap.to_s +
        " AP, is gathering dust in an alcove!"
  end

  ## YOUR CODE HERE
end

if __FILE__ == $0
  Adventure.new(Stdout.new, Knight.new(15, 3),
    [Monster.new(1, 1),
    FloorTrap.new(3),
    Monster.new(5, 3),
    Potion.new(5, 5),
    Monster.new(1, 15),
    Armor.new(10),
    FloorTrap.new(5),
    Monster.new(10, 10)]).play_out
end
```

↑ 0 ↓ · flag

+ Comment

New post

To ensure a positive and productive discussion, please read our forum posting policies before posting.

B | I | Link | <code> | Pic | Math | Edit: Rich | Preview

Make this post anonymous to other students

Subscribe to this thread at the same time

Add post