

Evidence for Implementation and Testing Unit

Ewa Lipinska

Cohort E20

I.T 1- Demonstrate one example of encapsulation that you have written in a program.

```
11 public abstract class Player {
12     private String name;
13     private int hp;
14     private int level;
15     private ArrayList<Treasure> pack;
16     private Room currentRoom;
17
18     public Player(String name) {
19         this.hp = 25;
20         this.level = 1;
21         this.pack = new ArrayList<>();
22         this.name = name;
23         this.currentRoom = null;
24     }
25
26     public String getName() {
27         return name;
28     }
```

I.T 2 - Example the use of inheritance in a program.

Parent class:

```
22 lines (13 sloc) | 378 Bytes
1 package Npcs.Healers;
2
3 import CombatItems.HealingTool;
4 import Npcs.Npc;
5
6 public abstract class Healer extends Npc implements IHeal {
7
8     private HealingTool healingtool;
9
10    public Healer (String name) {
11        super(name);
12        this.healingtool = HealingTool.getRandomHealingTool();
13    }
14
15
16    public HealingTool getHealingtool() {
17        return healingtool;
18    }
19
20
21 }
```

A class that inherits from the previous class:

```
package Npcs.Healers;

import Players.Player;

public class Cleric extends Healer {

    public Cleric(String name) {
        super(name);
    }

    public String heal(Player player1) {
        String result = "It looks like you can't afford cleric's services!\n";
        if (player1.canPayForHealing()) {
            result = "The cleric looks you over but sees no wounds to heal.\n";

            int playerHp = player1.getHp();
            if (playerHp < 25) {
                if (25 - playerHp < 5) {
                    playerHp = 25;
                } else {
                    player1.setHp(playerHp + getHealingtool().getHealingPower());
                }
                result = "The cleric heals you with " + getHealingtool().getName() + ". Your current hp: " + player1.getHp() + "\n";
            }
        }
        return result;
    }
}
```

An object of the subclass:

```
@Before
public void before() {
    cleric1 = new Cleric("Dulgren");
    player1 = new Barbarian("Brutus", Weapon.CLUB);
}
```

Method that uses information from the parent class:

```
public String heal(Player player1) {
    String result = "It looks like you can't afford cleric's services!\n";
    if (player1.canPayForHealing()) {
        result = "The cleric looks you over but sees no wounds to heal.\n";

        int playerHp = player1.getHp();
        if (playerHp < 25) {
            if (25 - playerHp < 5) {
                playerHp = 25;
            } else {
                player1.setHp(playerHp + getHealingtool().getHealingPower());
            }
            result = "The cleric heals you with " + getHealingtool().getName() + ". Your current hp: " + player1.getHp() + "\n";
        }
    }
    return result;
}
```

I.T 3 - Example of searching

```
cinema_boo...  sqlrunner.rb  customer.rb  screening.rb  film.rb  ticket.rb  console.rb
56 def booked_films
57   sql = "SELECT films.* FROM films
58         INNER JOIN screenings ON films.id =
59         screenings.film_id
60         INNER JOIN tickets ON screenings.id =
61         tickets.screening_id
62         WHERE tickets.customer_id = $1;"
63   values = [@id]
64   films_array = SqlRunner.run(sql, values)
65   return Film.map_items(films_array)
66 end
```

```
140 puts "Customer 4's booked films"
141 p customer4.booked_films()
142 puts ""
```

```
weekend_homework — Ewa@EVE — ..kend_homework — -zsh — 75x23
→ weekend_homework git:(master) × ruby console.rb
Customer 4's booked films
[#{Film:0x007f9a729bd688 @title="Thor Ragnarok", @id=7}, #{Film:0x007f9a729bd2f0 @title="The Shape of Water", @id=9}, #{Film:0x007f9a729bd228 @title="Hunt for the Wilderpeople", @id=8}]
→ weekend_homework git:(master) ×
```

I.T 4 – Example of sorting

```
main.rb
1 def dictionary_sort(array)
2   recursive_sort(array, [])
3 end
4
5 def recursive_sort(unsorted, sorted)
6   if unsorted.length <= 0
7     return sorted
8   end
9   last_element = unsorted.pop
10  still_to_sort = []
11
12  unsorted.each do |tested_element|
13    if last_element.downcase > tested_element.downcase
14      still_to_sort.push(last_element)
15      last_element = tested_element
16    else
17      still_to_sort.push(tested_element)
18    end
19  end
20  sorted.push(last_element)
21  recursive_sort(still_to_sort, sorted)
22 end
23
24 puts dictionary_sort(["b", "d", "hello", "0", "october", "pink", "car"])
```

```
ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-linux]
>
b
car
d
hello
0
october
pink
=> nil
>
```

I.T 5 - Example of an array, a function that uses an array and the result

```
fish.rb  fish_spec.rb  bear.rb x  river.rb  river_spec.rb  bear_spec.rb

1 class Bear
2   attr_reader :name, :stomach
3
4   def initialize(input_name)
5     @name = input_name
6     @stomach = []
7   end
8
9   def eat_a_fish(fish)
10    @stomach << fish
11  end
12
13  def is_starving?
14    @stomach.empty?
15  end
16
17  def hunt_for_fish(river)
18    unless river.fish.empty?
19      @stomach.push(river.fish.shift)
20    end
21  end
22 end

23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

bear_spec.rb

28
29 def test_bear_is_starving_true
30   assert_equal(true, @bear.is_starving?)
31 end
32
33 def test_bear_is_starving_false
34   @bear.eat_a_fish(@fish1)
35   assert_equal(false, @bear.is_starving?)
36 end
37
38 def test_hunt_for_fish_bear_eats_3_fish_5_fish_left_in_river
39   @bear.hunt_for_fish(@river)
40   assert_equal(false, @bear.is_starving?)
41   assert_equal(1, @bear.stomach.count)
42   assert_equal("Salmon", @bear.stomach[0].species)
43   @bear.hunt_for_fish(@river)
44   @bear.hunt_for_fish(@river)
45   assert_equal(3, @bear.stomach.count)
46   assert_equal("Trout", @bear.stomach[1].species)
47   assert_equal("Salmon", @bear.stomach[2].species)
48   assert_equal(5, @river.fish_count())
49 end

console.rb

1 fish.new("Herring"))
2
3
4
5
6
7
8
9 puts "hunt_for_fish bear eats 3 fish"
10 @bear.hunt_for_fish(@river)
11 @bear.hunt_for_fish(@river)
12 @bear.hunt_for_fish(@river)
13
14 p @bear.stomach
15
```

I.T 6 - Example of a hash, a function that uses a hash and the result

```
bounty.rb  space_cowboys.png  bounties.sql  console.rb

1 require('pg')
2
3 class Bounty
4   attr_accessor :name, :species, :bounty_value, :danger_level
5   attr_reader :id
6
7   def initialize(options)
8     @name = options['name']
9     @species = options['species']
10    @bounty_value = options['bounty_value'].to_i
11    @danger_level = options['danger_level']
12    @id = options['id'].to_i
13  end
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

console.rb

1 require('pry')
2 require_relative('models/bounty.rb')
3
4 options_hash1 = {'name'=> 'CSP4', 'species'=> 'android',
5   'bounty_value'=> 4500, 'danger_level'=>'medium'}
6
7 options_hash2 = {'name'=> 'Jack Dolan', 'species'=> 'human',
8   'bounty_value'=> 30000, 'danger_level'=>'ermagerdyderd'}
9
10 options_hash3 = {'name'=> 'Vak Irruct', 'species'=>
11   'klington', 'bounty_value'=> 10000, 'danger_level'=>'high'}
12
13
14 # Bounty.delete_all
15 #
16 p bounty1 = Bounty.new(options_hash1)
17 puts ""
18 p bounty2 = Bounty.new(options_hash2)
19 puts ""
20 p bounty3 = Bounty.new(options_hash3)
21
22 #
23 # bounty1.save()
24 # bounty2.save()
25 # bounty3.save()
26
```

I.T 7 - Example of polymorphism in a program

```
public ArrayList<Player> getPlayerOccupants() {  
    return playerOccupants;  
}
```

```
1  package Players.Fighters;  
2  
3  import CombatItems.Weapon;  
4  import Players.Player;  
5  import Surprises.Enemy;  
6  import Surprises.ISurprise;  
7  
8  import java.util.Random;  
9  
10 public abstract class Fighter extends Player implements IFight {  
11  
12     private Weapon weapon;  
13  
14     public Fighter(String name, Weapon weapon) {  
15         super(name);  
16         this.weapon = weapon;  
17     }  
18
```