

Evidence for Implementation and Testing Unit.

Jaime Lopez
Cohort E20

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

```
1  public class Customer {  
2  
3      private String name;  
4      private double wallet;  
5      private Table table;  
6  
7      public Customer(String name, double wallet){  
8          this.name = name;  
9          this.wallet = wallet;  
10         this.table = null;  
11     }  
12  
13     public String getName() { return this.name; }  
16  
17     public double getWallet() { return this.wallet; }  
20  
21     public void pay(double cost) { wallet -= cost; }  
24  
25     public Order placeOrder(){  
26         Order order = new Order( quantity: 1, MenuItem.LETTUCE);  
27         order.setTable(this.table);  
28         return order;  
29     }  
30  
31     public Table getTable() {  
32         return table;  
33     }  
34 }
```

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

```
public abstract class Character implements IMovable, ITargetable, ICollectionist, IFoundable {
    private String name;
    private int maxhp;
    private int hp;
    private int maxStamina;
    private int stamina;
    private Room currentRoom;
    private IWieldable primaryTool;
    private ArrayList<Treasure> treasures;
    private boolean dead;

    public Character(String name) {
        this.name = name;
        this.maxhp = 10;
        this.hp = maxhp;
        this.maxStamina = 0;
        this.stamina = maxStamina;
        this.treasures = new ArrayList<>();
        this.dead = false;
    }

    public Character(String name, int maxhp, int maxStamina, Room currentRoom) {
        this.name = name;
        this.maxhp = maxhp;
        this.maxStamina = maxStamina;
        this.hp = maxhp;
        this.stamina = maxStamina;
        this.currentRoom = currentRoom;
        this.treasures = new ArrayList<>();
        this.dead = false;
    }

    public String getName() { return name; }

    public int getHp() { return hp; }
```

The screenshot displays an IDE interface with three main components:

- Project Explorer (Left):** Shows a package structure including `non_player_character`, `player_character` (with sub-packages `fighter` and `spellcaster`), `Player`, `Character`, `collectables`, `dungeon`, `engine`, `META-INF`, `runner`, `resources`, `test`, and `java`.
- Code Editor (Center):** Shows the `@Before` method in `PlayerTest`:

```
@Before
public void setup() {
    player = new Player( name: "Gandalf");
    entryRoom = new EntryRoom( name: "Entry", description: "An Entry");
    endRoom = new EndRoom( name: "End", description: "An End");
    player2 = new Player( name: "Frodo", maxhp: 100, maxStamina: 50, entryRoom);
    entryRoom.setNorth(endRoom);
    chest = new CoinChest( quantity: 100, CoinType.GOLD);
    foe = new NonPlayerCharacter( name: "Giant Spider");
    key = new Key( name: "Golden", endRoom);
    hpotion = new Potion( name: "Red", poisonous: false, power: 5);
    sword = new Weapon( name: "Long Sword", value: 5, damage: 10);
    dagger = new Weapon( name: "Dagger", value: 5, damage: 5);
}
```
- Test Results (Bottom):** Shows the execution of `PlayerTest` with the following output:

```
run: PlayerTest.hasName x
>> Tests passed: 1 of 1 test - 20 ms
PlayerTest 20 ms /Library/Java/JavaVirtualMachines/jdk-9.0.1.jdk/Contents/Home/bin/java ...
hasName 20 ms
Process finished with exit code 0
```

```

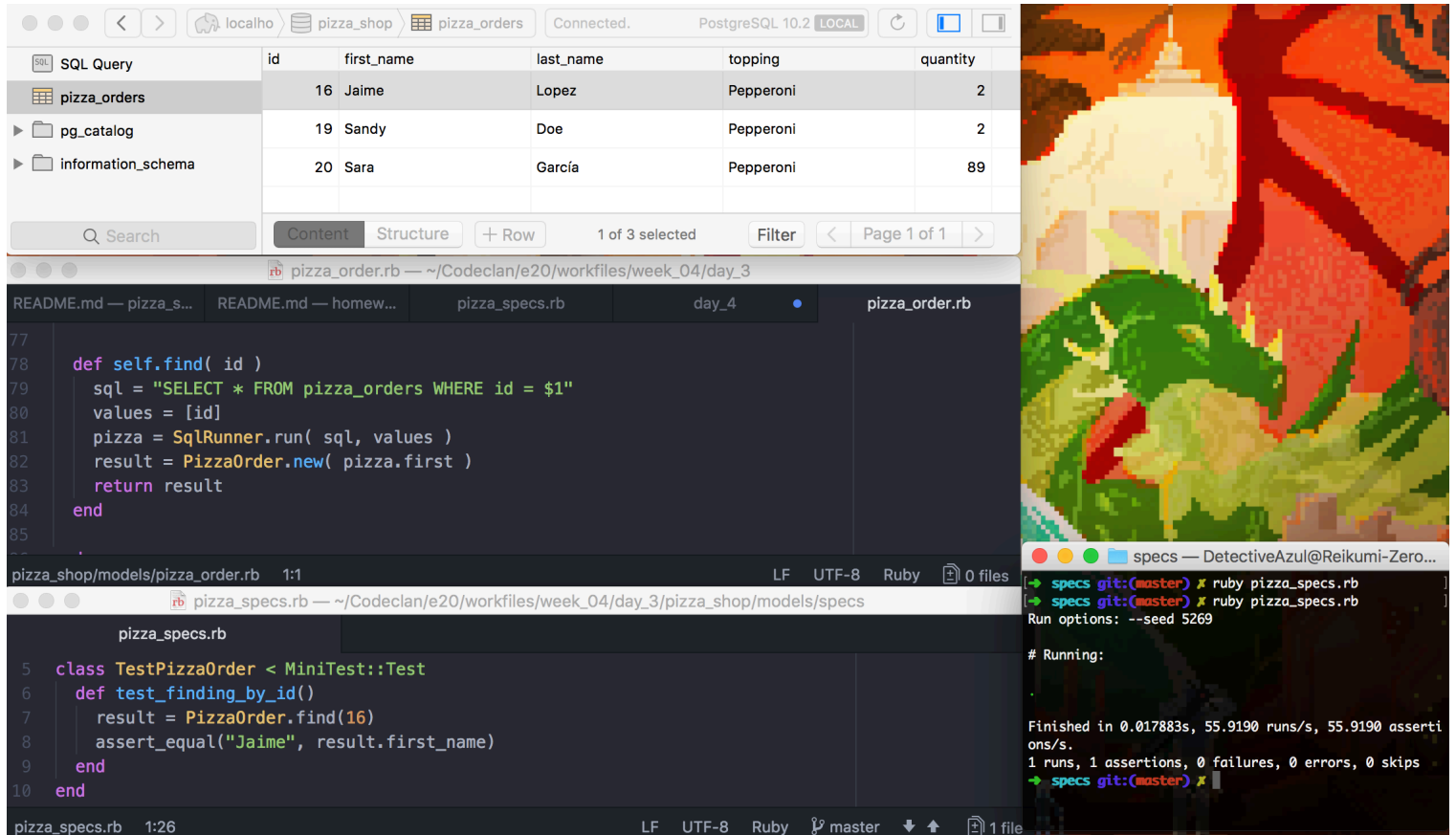
public class Player extends Character {

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

    public Player(String name, int maxhp, int maxStamina, Room currentRoom) {
        super(name, maxhp, maxStamina, currentRoom);
    }
}

```

I.T 3 - Example of searching



The screenshot displays a development environment with three main components:

- Database View (Top):** A table named `pizza_orders` with columns `id`, `first_name`, `last_name`, `topping`, and `quantity`. The table contains three rows:

id	first_name	last_name	topping	quantity
16	Jaime	Lopez	Pepperoni	2
19	Sandy	Doe	Pepperoni	2
20	Sara	García	Pepperoni	89
- Ruby Class (Middle):** The `pizza_order.rb` file shows a `find` method:


```

def self.find( id )
  sql = "SELECT * FROM pizza_orders WHERE id = $1"
  values = [id]
  pizza = SqlRunner.run( sql, values )
  result = PizzaOrder.new( pizza.first )
  return result
end

```
- Test Suite (Bottom):** The `pizza_specs.rb` file shows a test case:

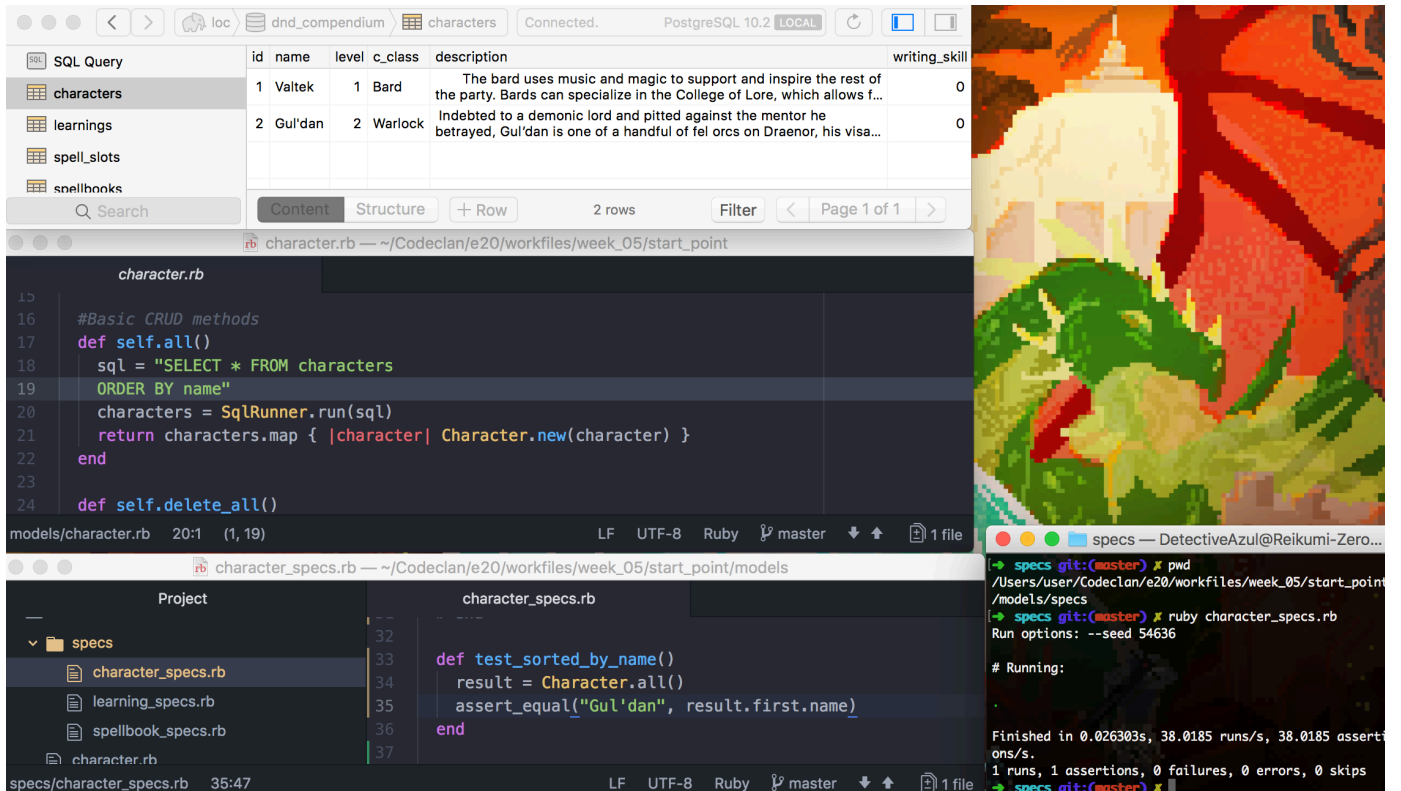

```

class TestPizzaOrder < MiniTest::Test
  def test_finding_by_id()
    result = PizzaOrder.find(16)
    assert_equal("Jaime", result.first_name)
  end
end

```

On the right side, there is a vertical image of a pizza with toppings. Below it, a terminal window shows the command `ruby pizza_specs.rb` being executed, resulting in a successful test run with 1 assertion and 0 failures.

I.T 4 - Example of sorting



The screenshot displays a development environment with three main windows:

- Database Window (Top):** Shows a PostgreSQL 10.2 connection. The 'characters' table is selected, displaying two rows of data. The table structure includes columns: id, name, level, c_class, description, and writing_skill.
- Code Editor (Middle):** Displays the `character.rb` file. It contains a `self.all()` method that executes a SQL query to fetch all characters, ordered by name, and returns them as a collection of `Character` objects.
- Test Runner (Bottom):** Shows the `character_specs.rb` file with a test `test_sorted_by_name()` that asserts the first character's name is 'Gul'dan'. The right sidebar shows the test results, indicating the test passed successfully.

id	name	level	c_class	description	writing_skill
1	Valtek	1	Bard	The bard uses music and magic to support and inspire the rest of the party. Bards can specialize in the College of Lore, which allows f...	0
2	Gul'dan	2	Warlock	Indebted to a demonic lord and pitted against the mentor he betrayed, Gul'dan is one of a handful of fel orcs on Draenor, his visa...	0

```
def self.all()
  sql = "SELECT * FROM characters
  ORDER BY name"
  characters = SqlRunner.run(sql)
  return characters.map { |character| Character.new(character) }
end
```

```
def test_sorted_by_name()
  result = Character.all()
  assert_equal("Gul'dan", result.first.name)
end
```

Finished in 0.026303s, 38.0185 runs/s, 38.0185 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips

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

```
array_specs.rb
1 require('minitest/autorun')
2 require('minitest/rb')
3 require_relative('array')
4
5 class TestArray < MiniTest::Test
6   def setup()
7     @array01 = ["Sandy", "Pawel", "Keith", "Sian"]
8   end
9
array.rb
1 def push_to_beginning(array, element)
2   array.unshift(element)
3   return array
4 end
5
array_specs.rb
9
10 def test_push_to_beginning()
11   result = push_to_beginning(@array01, "Jaime")
12   assert_equal("Jaime", @array01.first)
13 end
14 end
15
array_specs.rb 1:1 LF UTF-8 Ruby 0
PDA files — DetectiveAzul@Reikumi-Zero — ../les/PDA files
Remote Disc
[→ PDA files] ruby array_specs.rb
Run options: --seed 21629
# Running: msi
Finished in 0.001180s, 847.4576 runs/s, 847.4576 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

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

```
friends_spec.rb
29
30 @person3 = {
31   name: "Val",
32   age: 18,
33   monies: 20,
34   friends: ["Rick", "Jay"],
35   favourites: {
36     tv_show: "Pokemon",
37     things_to_eat: ["ratatouille", "stew"]
38   }
39 }
40
```

```
friends.rb
4
5 def get_favourite_tv_show(person)
6   return person[:favourites][:tv_show]
7 end
8
```

```
rb — ~/Codeclan/e20/workfiles/week_01/day_4/homework/starting_point
untitled friends_spec.rb
74 # 2. For a given person, return their favourite tv show
75 # (e.g. the function favourite_tv_show(@person2) should return the string
76 def test_getting_favourite_tv_show
77   result = get_favourite_tv_show(@person3)
78   assert_equal("Pokemon", result)
79 end
80 # 3. For a given person, check if they like a particular food
```

```
specs git:(master) ruby friends_spec.rb
Run options: --seed 25632

# Running:

.

Finished in 0.001099s, 909.9181 runs/s, 909.9181 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
→ specs git:(master) ✕
```

I.T 7 - Example of polymorphism in a program

