Implementation and Testing Evidence
Amy Morrison
Cohort 18

I.T 1

```
public class Enemy implements IDamageable, IAttack {

    private String name;
    private int hp;
    private int hitValue;

    public Enemy(String name, int hp, int hitValue){
        this.name = name;
        this.hp = hp;
        this.hitValue = hitValue;
    }

    public String getName() { return this.name; }

    public int getHP() { return this.hp; }

    public void setHP(int hp) { this.hp = hp; }

    public int getHitValue() { return this.hitValue; }


    public void inflictDamage(IDamageable character) {
            character.takeDamage(this.getHitValue());
        }

        public void takeDamage(int damage) { setHP(getHP() - damage); }
    }
```

I.T 2

```java
public class Bedroom extends Room {

    private final int room_number;
    private final BedroomType type;
    private final double roomCharge;

    public Bedroom(int room_number, BedroomType type) {

        super(type.getCapacity());
        this.room_number = room_number;
        this.type = type;
        this.roomCharge = type.getCharge();

    }

    public int getRoomNumber() {
        return this.room_number;
    }

    public double getNightRate() {
        return this.roomCharge;
    }

    public BedroomType getRoomType(){
        return this.type;
    }
```

```java
public class DiningRoom extends Room{
    public DiningRoom(int capacity) {
        super(capacity);
    }
}
```

```java
 public class ConferenceRoom extends Room {

      final String name;
      final double dailyRate;

      public ConferenceRoom(String name, int room_capacity, double dailyRate) {
          super(room_capacity);
          this.name = name;
          this.dailyRate = dailyRate;
      }

      public String getName() {
          return this.name;
      }

      public double getCharge() {
          return this.dailyRate;
      }

      public void pretty_print(){
          System.out.println(this.name  + " " + getCapacity() + " " + this.dailyRate);
      }
 }
```

I.T 3

```ruby
def SqlRunner.run(sql, values = [])
  db = PG.connect({dbname: 'defqlm9nkpe56a', host: 'ec2-54-83-46-116.compute-1.amazonaws.com',
    port: 5432, user: 'mtluwkblumblcm', password: '6e1df5cb36949157230b874f9cbcce292bfd2ea36b8d1261ec439d4e1dcb39ff'})
  db.prepare("query", sql)
  result = db.exec_prepared("query", values)
  db.close()
  return result
end
```

```ruby
def SqlRunner.map_object(array, classname)
  return array.map {|item| classname.new(item)}
end
```

```ruby
def Category.all()
    sql = "SELECT * FROM categories;"
    merchants = SqlRunner.run_sql_and_map(sql, Category)
end
```

```
    108: binding.pry
 => 109: nil

[[1] pry(main)> Category.all()
 => [#<Category:0x00007fb84420c138 @id=48, @luxury="f", @name="rent">,
  #<Category:0x00007fb84420c070 @id=49, @luxury="f", @name="electricity bill">,
  #<Category:0x00007fb84420c228 @id=50, @luxury="f", @name="gas bill">,
  #<Category:0x00007fb84420d628 @id=51, @luxury="f", @name="phone bill">,
  #<Category:0x00007fb84420fd88 @id=52, @luxury="f", @name="gym membership">,
  #<Category:0x00007fb844207ef8 @id=53, @luxury="f", @name="groceries">,
  #<Category:0x00007fb844207e30 @id=54, @luxury="t", @name="eating out">,
  #<Category:0x00007fb844207d68 @id=55, @luxury="t", @name="coffee">,
  #<Category:0x00007fb844207c00 @id=56, @luxury="t", @name="alcohol">,
  #<Category:0x00007fb844207a98 @id=57, @luxury="t", @name="socialising">,
  #<Category:0x00007fb844207958 @id=58, @luxury="t", @name="presents">,
  #<Category:0x00007fb844207890 @id=59, @luxury="t", @name="credit card bill">,
  #<Category:0x00007fb8442077c8 @id=60, @luxury="t", @name="shopping">,
  #<Category:0x00007fb8442076d8 @id=61, @luxury="t", @name="travel">,
  #<Category:0x00007fb844207610 @id=62, @luxury="t", @name="lazy travelling">]
[[2] pry(main)> Category.find(54)
 => #<Category:0x00007fb8441a4150 @id=54, @luxury="t", @name="eating out">
```

I.T 4

```ruby
def Category.most_spent_on()
  sql = "SELECT SUM(value),category_id FROM transactions GROUP BY category_id ORDER BY sum DESC;"
  result = SqlRunner.run(sql)
  return result[0]['category_id'].to_i
end
```

```
[1] pry(#<Sinatra::Application>)> Category.most_spent_on()
=> 1
```

I.T 5

```ruby
numbers = [1, 10, 40, 30, 60, 100]

def cube_numbers(numbers)
  return numbers.map{|number|number**3}
end

p "Call cube_numbers on array: #{numbers}: #{cube_numbers(numbers)}"
```

```
[➜  pda_work git:(master) ✗ ruby array.rb
"Call cube_numbers on array: [1, 10, 40, 30, 60, 100]: [1, 1000, 64000, 27000, 216000, 1000000]"
```

I.T 6

```ruby
albania = {
  name: "Albania",
  capital: "Tirana",
  landlocked: "false",
  religions: ["muslim", "christian"]
}

def is_landlocked(country)
  return country[:landlocked]
end

p "Is country landlocked? #{is_landlocked(albania)}"
```

```
[➜  pda_work git:(master) ✗ ruby hash.rb
"Is country landlocked? false"
```

I.T 7

```java
package com.codeclan.amymorrison.shakespeare.instruments;


import java.util.ArrayList;
import java.util.HashMap;

public class Exchange {

    private double till;
    private ArrayList<iSaleable> stock;

    public Exchange(){
        this.till = 0;
        stock = new ArrayList<>();
    }

    public void addStock(iSaleable stockItem){
        stock.add(stockItem);
    }

    public int getStockCount() {
        return this.stock.size();
    }

    public double getPotentialGrossProfit() {
        double sum = 0.00;
        for (StockItem item : stock) {
            sum += item.calculateMarkup ( );
        }
        return sum;
    }


}
```

```java
public interface iSaleable {

    double calculateMarkup();

    double getSellingPrice();

    double getBuyingPrice();
}
```

```
public class Microphone implements ISaleable {

    private AccessoryType accessoryType;

    public Microphone(double buyingPrice, double sellingPrice, boolean used, String description) {
        this.buyingPrice = buyingPrice;
        this.sellingPrice = sellingPrice;
        this.used = used;
        this.description = description;
        this.accessoryType = AccessoryType.MICROPHONE;
    }

    public String getAccessoryType() {
        return this.accessoryType.getName();
    }

    public double getBuyingPrice() {
        return this.buyingPrice;
    }

    public double getSellingPrice(){
        return this.sellingPrice;
    }

    public boolean isUsed() {
        return this.used;
    }

    public double calculateMarkup() {
        return this.sellingPrice - this.buyingPrice;
    }
}
```