Claire Connachan
CodeClan Cohort - E20

Evidence for Implementation and Testing Unit

I.T 1 - Week 9. Take a screenshot of an example of encapsulation in a program.

```
5   public abstract class Character {
6
7       protected String name;
8       protected int hp;
9
10      public Character(String name){
11          this.name = name;
12          this.hp = 100;
13      }
14
15      public String getName() {
16          return this.name;
17      }
18
19      public int getHp() {
20          return this.hp;
21      }
22
23      public void setHp(int hp) {
24          this.hp = hp;
25      }
26
```

!.T 2 - Week 9. Take a screenshot of the use of inheritance in a program.

```java
package Characters.Heroes.Fighters;

import Characters.Character;

public class Knight extends Fighter {

    public Knight(String name){
        super(name);
        this.weapon = Weapon.SWORD;
        this.defence = Defence.SHIELD;
    }

    public int stab(){
        return 20;
    }

    //move2
    //For the knight the stab has 100% chance of hitting.
    public void signatureMove(Character characterToAttack){
       {characterToAttack.takeDamage(this.stab());}
    }


}
```

```java
import Characters.Creatures.Creature;
import Characters.Creatures.CreatureType;
import Characters.Heroes.Fighters.Defence;
import Characters.Heroes.Fighters.Knight;
import Characters.Heroes.Fighters.Weapon;
import Characters.Heroes.Treasure;
import org.junit.Before;
import org.junit.Test;

import static junit.framework.TestCase.assertEquals;

public class KnightTest {
    private Knight knight1;
    private Weapon sword1;
    private Defence defence1;
    private Creature dragon1;
    private Treasure treasure1;

    @Before
    public void before(){
        knight1 = new Knight("Mike");
        sword1 = Weapon.SWORD;
        defence1 = Defence.SHIELD;
        dragon1 = new Creature(CreatureType.DRAGON);
        treasure1 = Treasure.DIAMOND;
    }

```

```java
package Characters.Heroes.Fighters;

import Characters.Character;
import Characters.Heroes.Hero;

public abstract class Fighter extends Hero {

    protected Weapon weapon;
    protected Defence defence;

    public Fighter(String name){
        super(name);
        this.weapon = null;
        this.defence = null;

    }

    public Weapon getWeapon() {
        return weapon;
    }

    public void setWeapon(Weapon weapon) {
        this.weapon = weapon;
    }

    public Defence getDefence() {
        return defence;
    }
```

I.T 3 Demonstrate searching data in a program.

Screenshot of a function that searches data:

```
62    def is_song_in_playlist?(song_to_check)
63      song_titles = @playlist.map { |song| song.title }
64      song_titles.include?(song_to_check.title)
65    end
```

Screenshot of the result of the function running:

```
[→  homework git:(master) ✗ ruby runner.rb
true
```

I.T 4 Demonstrate sorting data in a program.

Screenshot of a function that sorts data:

```
def most_popular_screening()
  sql = "
    SELECT COUNT(screenings.time), screenings.time
      FROM tickets
      INNER JOIN films ON tickets.film_id = films.id
      INNER JOIN screenings ON tickets.screening_id =
      screenings.id
      WHERE films.id = $1
      GROUP BY screenings.time
      ORDER BY COUNT(screenings.time) DESC
      LIMIT 1;
    "
  values = [@id]
  array = SqlRunner.run(sql, values)
  return array[0]
end
```

Screenshot of the result of the function running:

```
→  homework git:(master) ✗ ruby runner.rb
{"count"=>"4", "time"=>"9am"}
```

I.T 5 Demonstrate the use of an array in a program.

Screenshot of an array in a program and function that uses the array:

```ruby
claire = Guest.new("Claire")
ewa = Guest.new("Ewa")
mike = Guest.new("Mike")
aileen = Guest.new("Aileen")

@guest = Guest.new("Lewis", 20)

@occupants = [claire, ewa, mike, aileen]

@room = Room.new("Karaoke Room",
@occupants, @playlist)
```

```ruby
def add_guest(guest_to_add)
    if @occupants.count < @capacity &&
    guest_to_add.wallet >= @fee
      @occupants << guest_to_add
      @till += @fee
    end
end
```

Screenshot of the result of the function running:

```ruby
p @room.occupants
@room.add_guest(@guest)
p @room.occupants
```

```
→  homework git:(master) ✗ ruby runner.rb
[#<Guest:0x007fcacda98cc0 @name="Claire", @wallet=0, @fave_song=nil>, #<Guest:0x
007fcacda98c70 @name="Ewa", @wallet=0, @fave_song=nil>, #<Guest:0x007fcacda98c20
 @name="Mike", @wallet=0, @fave_song=nil>, #<Guest:0x007fcacda98bd0 @name="Ailee
n", @wallet=0, @fave_song=nil>]
[#<Guest:0x007fcacda98cc0 @name="Claire", @wallet=0, @fave_song=nil>, #<Guest:0x
007fcacda98c70 @name="Ewa", @wallet=0, @fave_song=nil>, #<Guest:0x007fcacda98c20
 @name="Mike", @wallet=0, @fave_song=nil>, #<Guest:0x007fcacda98bd0 @name="Ailee
n", @wallet=0, @fave_song=nil>, #<Guest:0x007fcacda98b80 @name="Lewis", @wallet=
20, @fave_song=nil>]
→  homework git:(master) ✗
```

I.T 6 - Demonstrate the use of a hash in a program.

Screenshot of a hash in a program and a function that uses the hash:

```ruby
@customers = [
  { name: "Craig", pets: [], cash: 1000 },
  { name: "Zsolt", pets: [], cash: 50 }
]

@new_pet = { name: "Bors the Younger", pet_type: :cat, breed: "Cornish Rex", price: 100 }

@pet_shop = { pets: [
      { name: "Sir Percy", pet_type: :cat, breed: "British Shorthair", price: 500 },
      { name: "King Bagdemagus", pet_type: :cat, breed: "British Shorthair", price: 500 },
      { name: "Sir Lancelot", pet_type: :dog, breed: "Pomsky", price: 1000 },
      { name: "Arthur", pet_type: :dog, breed: "Husky", price: 900 },
      { name: "Tristan", pet_type: :dog, breed: "Basset Hound", price: 800 },
      { name: "Merlin", pet_type: :cat, breed: "Egyptian Mau", price: 1500 }
    ],
    admin: { total_cash: 1000, pets_sold: 0 }, name: "Camelot of Pets" }
```

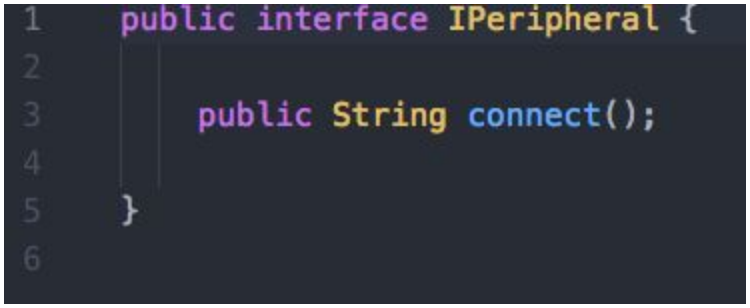
```ruby
def find_pet_by_name(shop, expected_name)
  result = nil
  for pet in shop[:pets]
    if pet[:name] == expected_name
      result = pet
    end
  end
  return result
end
```

Screenshot of the result of the function running:

```
[→  start_point git:(master) ✗ ruby runner.rb
{:name=>"Arthur", :pet_type=>:dog, :breed=>"Husky", :price=>900}
→  start_point git:(master) ✗
```

I.T 7 - Week 6. Demonstrate the use of Polymorphism in a program:

IPeripherals interface provides iPeripheral objects with connectivity.



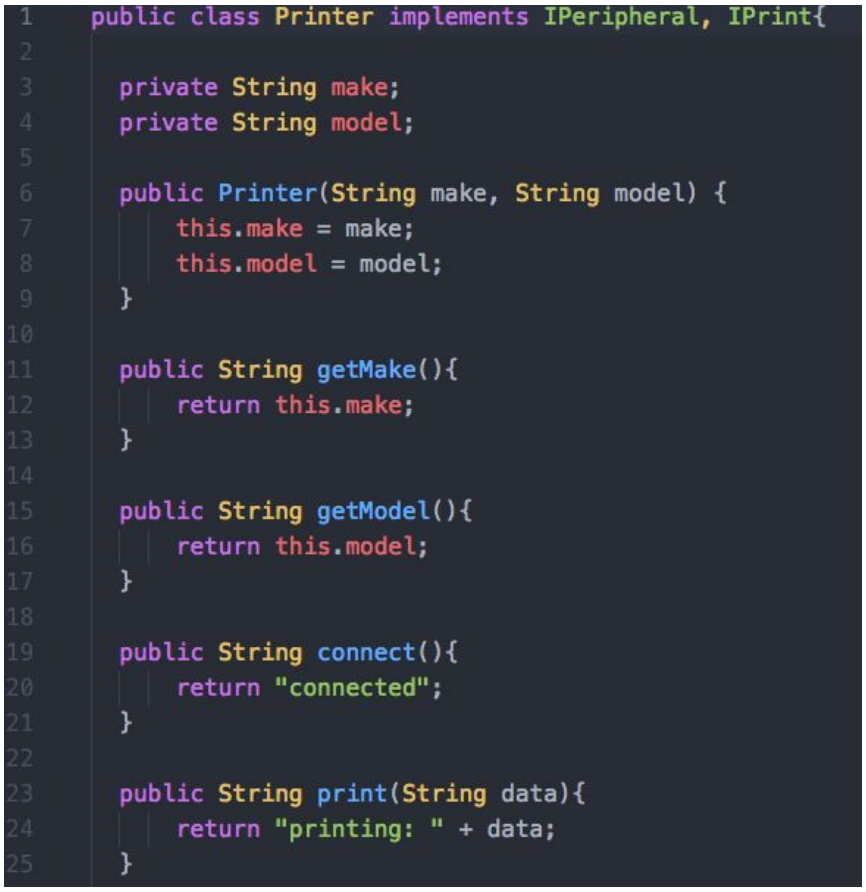
```
public interface IPeripheral {

    public String connect();

}

```

Printer class implements iPeripheral



```
public class Printer implements IPeripheral, IPrint{

  private String make;
  private String model;

  public Printer(String make, String model) {
      this.make = make;
      this.model = model;
  }

  public String getMake(){
      return this.make;
  }

  public String getModel(){
      return this.model;
  }

  public String connect(){
      return "connected";
  }

  public String print(String data){
      return "printing: " + data;
  }
```

Mouse class implements iPeripheral.

```
public class Mouse implements IPeripheral{

    private String make;
    private String model;

    public Mouse(String make, String model) {
        this.make = make;
        this.model = model;
        }

     public String getMake(){
        return this.make;
     }

     public String getModel(){
        return this.model;
     }

     public String connect(){
        return "mouse is connected";
     }


}
```

Office class implements iPeripheral and has an arraylist of iperipheral objects

```
public class Office implements IPeripheral{

    private ArrayList<IPeripheral> office_items;

    public Office() {
        this.office_items = new ArrayList<>;
        }

     public String getOfficeItems(){
        return this.office_items;
     }

```

Method to add iPeripheral object to iPeripheral arraylist in Office class.

```
public void addItem(IPeripheral mouse) { office.add(mouse) }
```