PDA: Software Development Level 8

Student: David Sanchez Rodriguez

EVIDENCE: UNIT I & T

Ref: I.T. 5

Demonstrate the use of an array in a program.

- An array in a program

```
david_sanchez_pda.rb
# IT5 - Demonstrate the use of an array in a program. Take screenshots of:

# - An array in a program
cars = ["Ferrari", "Porsche", "Bugatti", "Lamborghini", "Camaro", "Mustang"]
p cars
```

```
Desktop — rocco@Davids-MBP — ~/Desktop — -zsh — 104×24
➜ Desktop ruby david_sanchez_pda.rb
["Ferrari", "Porsche", "Bugatti", "Lamborghini", "Camaro", "Mustang"]
➜ Desktop
```

- A function that uses the array

```
david_sanchez_pda.rb
# - A function that uses the array
#This function adds one car to the array
def add_cars(cars,car)
  p cars.push(car)
end

#This function counts the cars in the array
def count_cars(brands)
  puts brands.count
end
```

- The result of the function running

```
# - The result of the function running
show_cars(cars, "Fiat")
count_cars(cars)
```

```
Desktop — rocco@Davids-MBP — -zsh — 104×24
~/Desktop                                  ..eek_01/
➜ Desktop ruby david_sanchez_pda.rb
["Ferrari", "Porsche", "Bugatti", "Lamborghini", "Camaro", "Mustang", "Fiat"]
7
```

Ref: I.T. 6

Demonstrate the use of a hash in a program.

- A hash in a program

```ruby
# IT6 - Demonstrate the ue of a hash in a program. Take screenshots of:

# - A hash in a program
cars = {"Ferrari" => {
  :speed => 320,
  :horsepower => 500,
  :weight => 350,
  :code => [21, 15, 24, 56]},
  "Porsche" => {
    :speed => 280,
    :horsepower => 340,
    :weight => 220,
    :code => [13, 18, 23, 32]},
    "Bugatti" => {
      :speed => 260,
      :horsepower => 210,
      :weight => 180,
      :code => [23, 11, 20, 38]
      }}

p cars
```

- A function that uses the hash

```ruby
#- A function that uses the hash
#This function shows the content of the hash
def show_cars(cars)
  puts cars
end
#This function shows the minimum code of the input car
def min_code_car(cars, car)
  p cars[car][:code].min
end
```

- The result of the function running

```ruby
# - The result of the function running
show_cars(cars)
min_code_car(cars, "Porsche")
```

```
Desktop — rocco@Davids-MBP — -zsh — 97×24
~/Desktop                    ..eek_01/day_02
➜ Desktop ruby david_sanchez_pda.rb
{"Ferrari"=>{:speed=>320, :horsepower=>500, :weight=>350, :code=>[21, 15, 24, 56]}, "Porsche"=>{:
speed=>280, :horsepower=>340, :weight=>220, :code=>[13, 18, 23, 32]}, "Bugatti"=>{:speed=>260, :h
orsepower=>210, :weight=>180, :code=>[23, 11, 20, 38]}}
13
```

Ref: I.T. 3

Demonstrate searching data in a program.

- A function that searches data

david_sanchez_pda.rb

```ruby
# IT3 – Demonstrate searching data in a program. Take screenshots of:

cars = ["Ferrari" , "Porsche", "Bugatti", "Lambo", "Fiat", "VW", "Audi", "Vauxhall" ]

# – Function that searches data
#This function tella s if the car model is in the list
def find_car(cars, model)
    if cars.include?(model)
      print "The #{model} is in the list"
    else
      print "Car not in the list"
  end
end
```

- The result of the function running

david_sanchez_pda.rb

```ruby
# IT3 – Demonstrate searching data in a program. Take screenshots of:

cars = ["Ferrari" , "Porsche", "Bugatti", "Lambo", "Fiat", "VW", "Audi", "Vauxhall" ]

# – Function that searches data
#This function tella s if the car model is in the list
def find_car(cars, model)
    if cars.include?(model)
      print "The #{model} is in the list"
    else
      print "Car not in the list"
  end
end

# – The result of the function running
find_car(cars, "Audi")
```

Desktop — rocco@Davids-MBP — ~/Desktop — -zsh — 79×24

```
➜ Desktop ruby david_sanchez_pda.rb
The Audi is in the list%
```

Ref: I.T. 4

Demonstrate sorting data in a program.

- A function that sorts data

david_sanchez_pda.rb

```ruby
#IT4 - Demonstrate sorting data in a program. Take screenshots of:

# - Function that sorts data
numbers = [100 , 98, 130, 23, 12, 1000]

def sort( numbers)
  a = numbers.sort
  b = a.reverse
  p b
end
```

- The result of the function running

david_sanchez_pda.rb

```ruby
#IT4 - Demonstrate sorting data in a program. Take screenshots of:

# - Function that sorts data
numbers = [100 , 98, 130, 23, 12, 1000]

def sort( numbers)
  a = numbers.sort
  b = a.reverse
  p b
end

# - The result of the function running
sort(numbers)
```

Desktop — rocco@Davids-MBP — ~/Desktop — -zsh — 79×24

```
➜ Desktop ruby david_sanchez_pda.rb
[1000, 130, 100, 98, 23, 12]
```

Ref: I.T. 7

Demonstrate the use of Polymorphism in a program

```java
import javax.sound.midi.Instrument;
import java.util.ArrayList;

public class Shop {

    private String name;
    private ArrayList<ISell> items;




    public Shop(String name){
        this.name = name;
        items = new ArrayList<ISell>();
    }

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

    public void addItemToStock(ISell instrument){
        items.add(instrument);
    }

    public void removeItemFromStock(ISell instrument){
        items.remove(instrument);
    }

    public int countItems(){
        return this.items.size();
    }

    public double totalProfitOnStock() {
        double profit = 0;
        for(ISell sellable : items) {
            profit += sellable.calculateMarkup();
        }
        return profit;
    }
}
```

```java
public abstract class Instruments implements ISell{

    String made;
    String colour;
    String type;
    double sellingprice;
    double buyingprice;
    private InstType instType;

    public Instruments(String made, String colour, String type, double sellingprice, double buyingprice, InstType instType) {
    this.made = made;
    this.colour = colour;
    this.type = type;
    this.sellingprice = sellingprice;
    this.buyingprice = buyingprice;
    this.instType = instType;

    }

    public String getType(){
        return this.type;
    }

    public String getColour(){
        return this.colour;
    }

    public String getMade(){
        return this.made;
    }

    public double getSellingprice(){
        return this.sellingprice;
    }

    public double getBuyingprice(){
        return this.buyingprice;
    }

    public String getInstType(){
        return this.instType.getType();
    }

    public double calculateMarkup(){
        double markup = (sellingprice - buyingprice);
        return markup;
    }
}
```

```java
public class Piano extends Instruments implements IPlay,ISell {

    int pedal;


    public Piano(String made, String colour, String type, int pedal, double sellingprice, double buyingprice, InstType instType){
        super(made, colour,type, sellingprice,buyingprice, instType);

        this.pedal = pedal;

    };

    public String sound() {return "Plong, Plong";}

    public int getPedal(){return this.pedal;}

}
```

```java
public interface ISell {

    double calculateMarkup();

}
```

Ref: I.T. 1

Take a screenshot of an example of encapsulation in a program

```java
public abstract class  Paddock {

    public ArrayList<Dinosaur> dinosaurs;
    private String name;
    private int capacity;
    private Food food;
    private DinosaurType dinoType;



    public Paddock(String name, int capacity,DinosaurType dinoType ) {
        this.name = name;
        this.dinoType = dinoType;
        this.capacity = capacity;
        this.dinosaurs = new ArrayList<Dinosaur>();

    }

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

    public void setPaddockName(String name) {
        this.name = name;
    }

    public int getPaddockCapacity() {
        return this.capacity;
    }

    public String getDinoType(){
        return this.dinoType.getDinoType();
    }
```

Ref: I.T. 2

Take a screenshot of the use of inheritance in a program. Take screenshots of:

- A Class

```java
package Paddocks;

import Dinosaurs.Dinosaur;
import Dinosaurs.DinosaurType;
import Dinosaurs.Herbivore;

import java.util.ArrayList;

public abstract class  Paddock {

    public ArrayList<Dinosaur> dinosaurs;
    private String name;
    private int capacity;
    private Food food;
    private DinosaurType dinoType;



    public Paddock(String name, int capacity,DinosaurType dinoType ) {
        this.name = name;
        this.dinoType = dinoType;
        this.capacity = capacity;
        this.dinosaurs = new ArrayList<Dinosaur>();

    }

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

    public void setPaddockName(String name) {
        this.name = name;
    }

    public int getPaddockCapacity() {
        return this.capacity;
    }

    public String getDinoType(){
        return this.dinoType.getDinoType();
    }
```

- A Class that inherits from the previous class

```java
package Paddocks;

import Dinosaurs.DinosaurType;

public class CarnPaddock extends Paddock {

    public CarnPaddock(String name, int capacity, DinosaurType dinoType){
        super(name,capacity, dinoType);
    }


}
```

- An Object in the inherited class

```java
public class CarnPaddockTest {

    CarnPaddock carnPaddock;
    CarnPaddock carnPaddock1;
    HerbPaddock herbPaddock;
    Dinosaur dinosaur;
    Carnivore carnivore;
    Herbivore herbivore;


    @Before
    public void before(){
        carnPaddock = new CarnPaddock( name: "The Meat Train",  capacity: 5,DinosaurType.CARNIVORE);
        carnPaddock1 = new CarnPaddock( name: "Poor little prey",  capacity: 5,DinosaurType.CARNIVORE);
```

- A Method that uses the information inherited from another class

```java
public class CarnPaddockTest {

    CarnPaddock carnPaddock;
    CarnPaddock carnPaddock1;
    HerbPaddock herbPaddock;
    Dinosaur dinosaur;
    Carnivore carnivore;
    Herbivore herbivore;


    @Before
    public void before(){
        carnPaddock = new CarnPaddock( name: "The Meat Train",  capacity: 5,DinosaurType.CARNIVORE);
        carnPaddock1 = new CarnPaddock( name: "Poor little prey",  capacity: 5,DinosaurType.CARNIVORE);
        herbPaddock = new HerbPaddock( name: "Herbivore",  capacity: 5, DinosaurType.HERBIVORE);
        carnivore = new Carnivore( name: "Antony", DinosaurType.CARNIVORE,  healthPoints: 10, CarnSubType.SMALL);
        herbivore = new Herbivore( name: "Raul", DinosaurType.HERBIVORE,  healthPoints: 10);
    }

    @Test
    public void canGetPaddockName(){
        assertEquals( expected: "The Meat Train", carnPaddock.getPaddockName());
    }

    @Test
    public void canSetPaddockName(){
        carnPaddock.setPaddockName("The last Monsters");
        assertEquals( expected: "The last Monsters", carnPaddock.getPaddockName());
    }

    @Test
    public void canGetDinoType(){
        assertEquals( expected: "Carnivore", carnPaddock.getDinoType());

    }
```