# PDA in Software Programming

## Evidence for Implementation and Testing Unit

Simon Atkins - Cohort e20

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

In this programme, private variables such as Airport Name and ArrayList of Planes are encapsulated within the Airport class and can only be accessed with getters such as .getName().

Airport.java

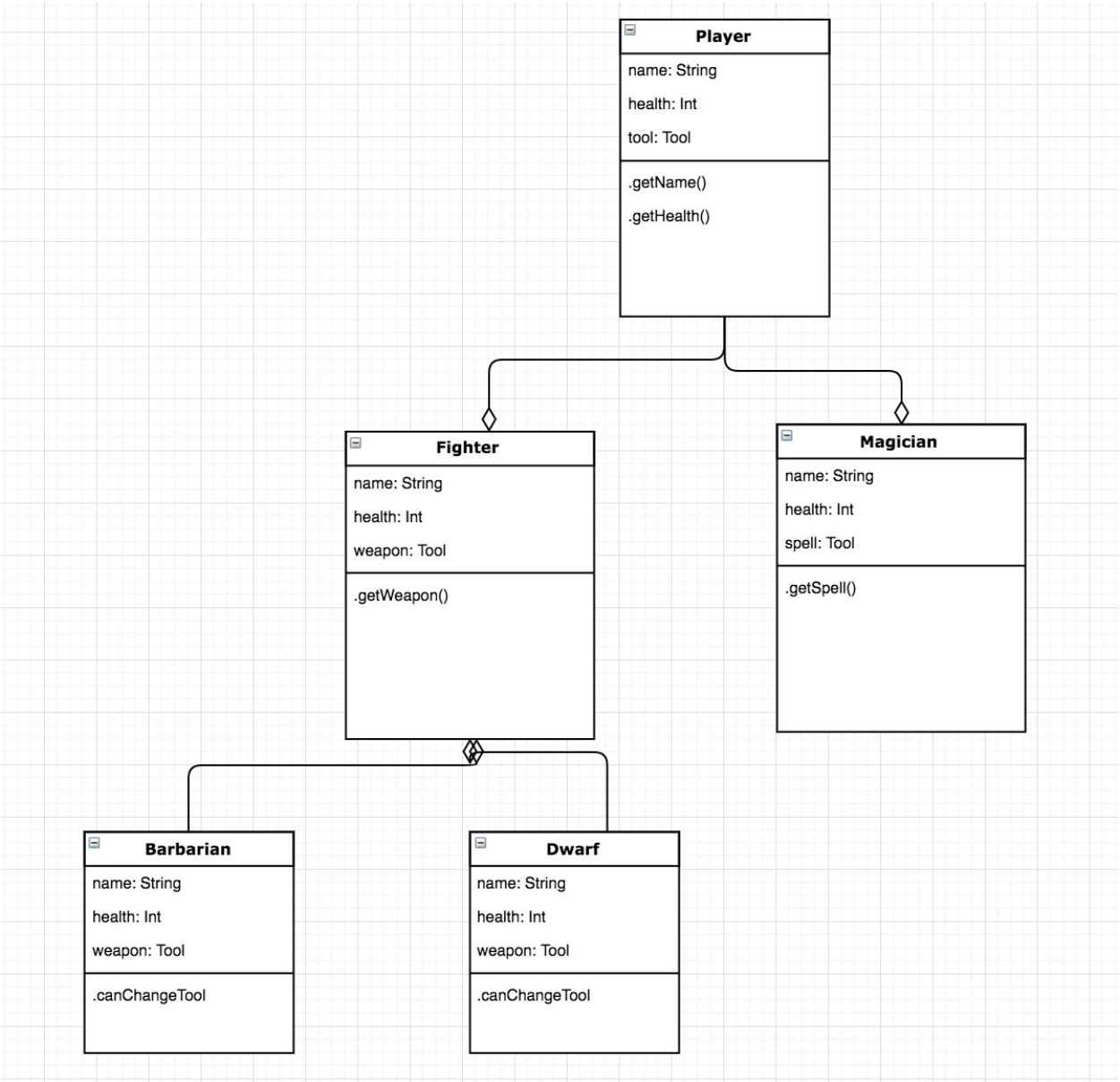
```java
package Airport;

import Airport.Person.Passenger;

import java.util.ArrayList;

public class Airport {

    private String name;
    private ArrayList<Plane> fleet;
    private int maxPlanes;
    private ArrayList<Passenger> tourists;

    public Airport(String name, ArrayList<Plane> fleet, int maxPlanes){
    this.name =name;
    this.fleet = fleet;
    this.maxPlanes = maxPlanes;
    this.tourists = new ArrayList<>();
    }

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

    public int getNumberOfPlanesInFleet() { return fleet.size(); }

    public void planeLeaveAirport(Plane plane) { fleet.remove(plane); }

    public void planeArrivesAtAirport(Plane plane){
        if (fleet.size() < maxPlanes) {
        fleet.add(plane);} else {
            System.out.println("Your airport is full!");
        }
    }

}
```

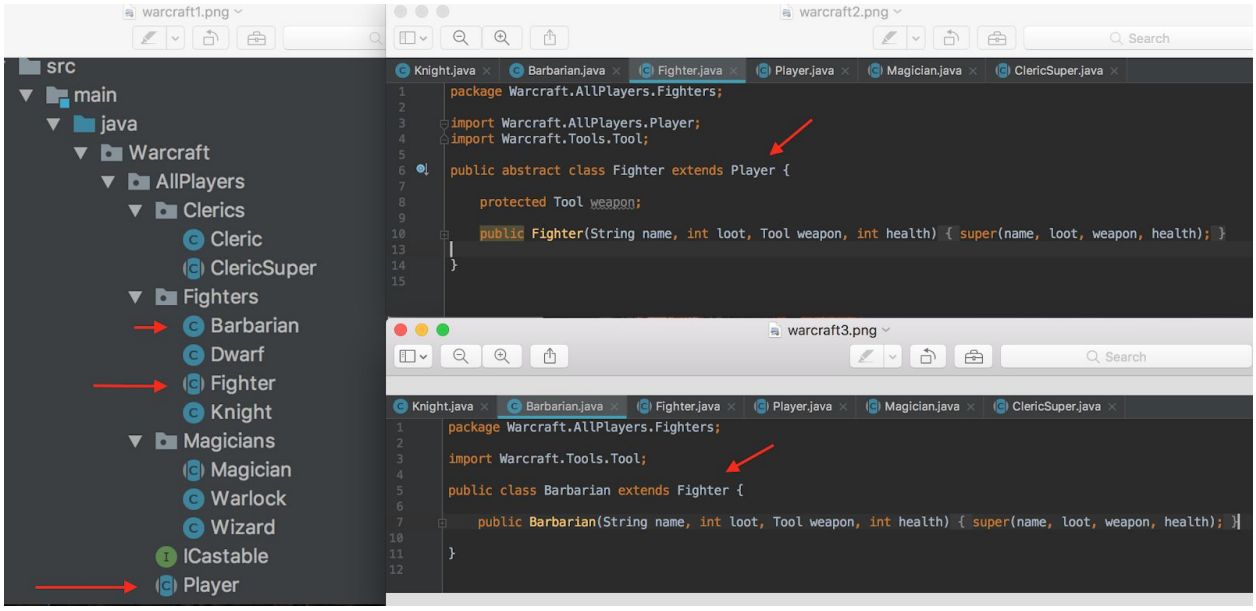
In this second example of encapsulation, the data is passed in to the class using a generic parameter called ‘options’ which cannot be interfered with from the outside world.

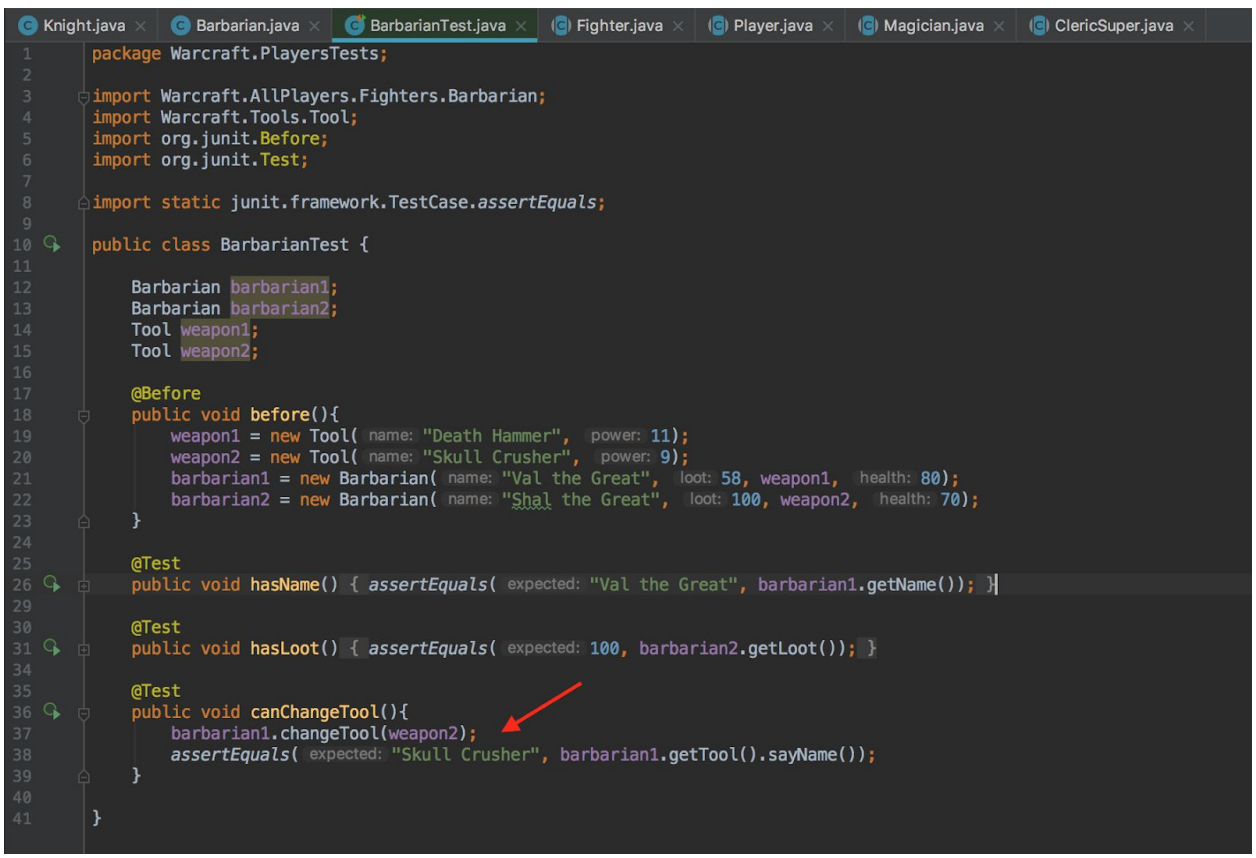
```ruby
require_relative('../db/sql_runner')

class Troll

  attr_reader :id
  attr_accessor :name, :breed, :adoptable, :admission_date, :owner_id

  def initialize( options )
    @id = options['id'].to_i if options['id']
    @name = options['name']
    @breed = options['breed']
    @adoptable = options['adoptable']
    @admission_date = options['admission_date']
    @owner_id = options['owner_id'] if options['owner_id']
  end

  def save

    sql = "INSERT INTO trolls
    (
    name, breed, adoptable, admission_date, owner_id
    )
    VALUES
    (
    $1, $2, $3, $4, $5
    )
    RETURNING id"

    values = [@name, @breed, @adoptable, @admission_date, @owner_id]
    result = SqlRunner.run( sql, values )
    @id = result.first['id']

  end
```

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

In this Warcraft program, the Barbarian class inherits from both the Fighter and the Player classes. The Barbarian is able to access the ‘weapon’ method because of the Fighter class directly above, and the ‘name’ method because of the Player abstract class at the top.


Player
name: String
health: Int
tool: Tool
.getName()
.getHealth()
Fighter
name: String
health: Int
weapon: Tool
.getWeapon()
Magician
name: String
health: Int
spell: Tool
.getSpell()
Barbarian
name: String
health: Int
weapon: Tool
.canChangeTool
Dwarf
name: String
health: Int
weapon: Tool
.canChangeTool




# I.T 3 - Example of searching

(if you do not have a search and sort algorithm, write one up, take a screenshot. Remember to include the results as well.)

Evidence for unit: A Ruby function & result

```ruby
#An example array
rabbit_names = ["Harry", "Barry", "Larry", "Carrie"]

#A search function
def find_name(all_names, name_to_find)
  for name in all_names
    if name == name_to_find
      puts "#{name_to_find} is here!"
    end
  end
end

#A method calling the function
find_name(rabbit_names, "Harry")
```

```
ruby 2.5.0p0 (2017
[x86_64-linux]
>
Harry is here!
=> ["Harry", "Barry
>
```

Evidence for unit: A ruby and SQL search function

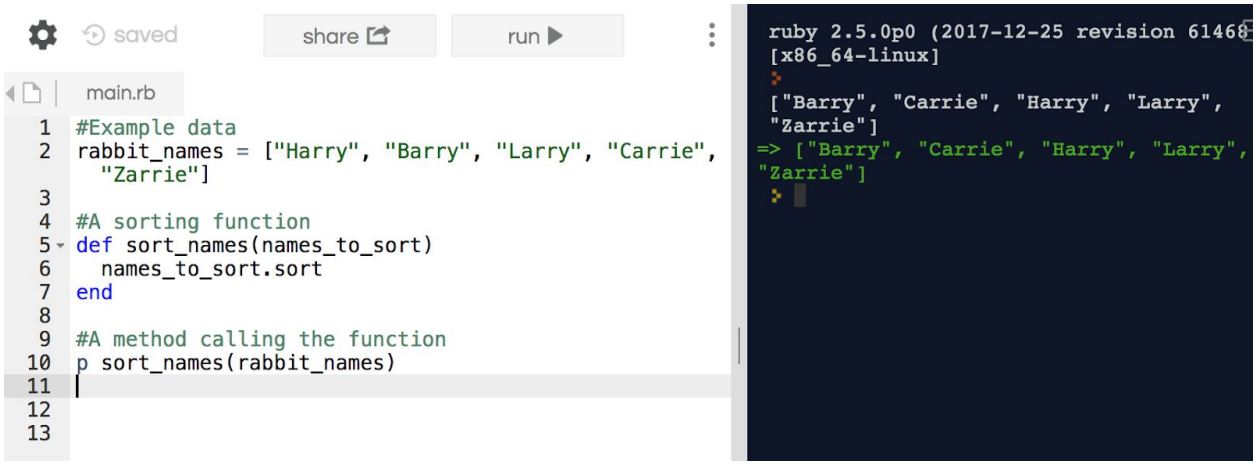
```ruby
def find_films
  sql = "SELECT films.*
    FROM films
    INNER JOIN tickets
    ON tickets.film_id = films.id
    WHERE customer_id = $1"
  values = [@id]
  films_array = SqlRunner.run(sql, values)
  film_objects = films_array.map {|film|
  Film.new(film)}
  return film_objects
end
```

Evidence for unit: Terminal showing result of calling the Ruby/SQL function above

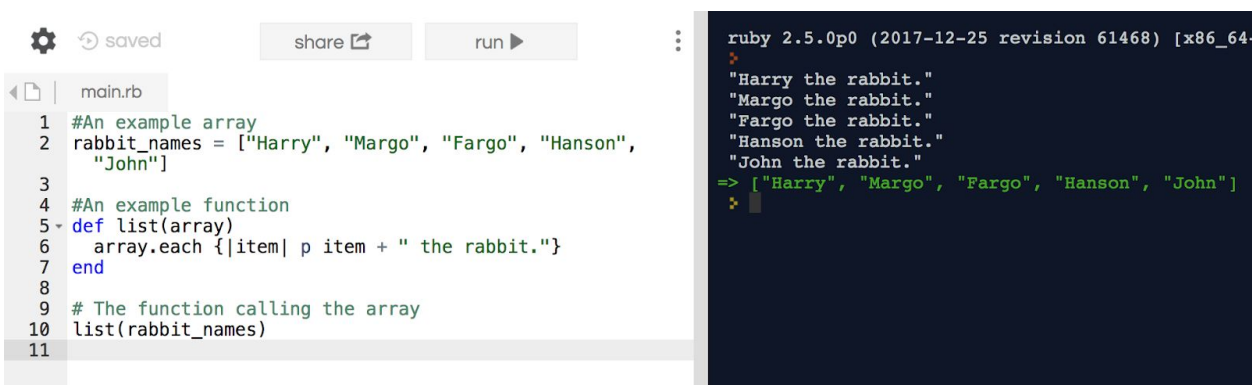
```
➜  day_5_homework git:(master) ✗ ruby console.rb
[#<Film:0x007fb15e9dc378 @title="Texas Chainsaw Massacre", @price="10
", @id=111>, #<Film:0x007fb15e9dc260 @title="Texas Chainsaw Massacre"
, @price="10", @id=111>, #<Film:0x007fb15e9dc148 @title="Babe", @pric
e="10", @id=112>]
```

# I.T 4 – Example of sorting

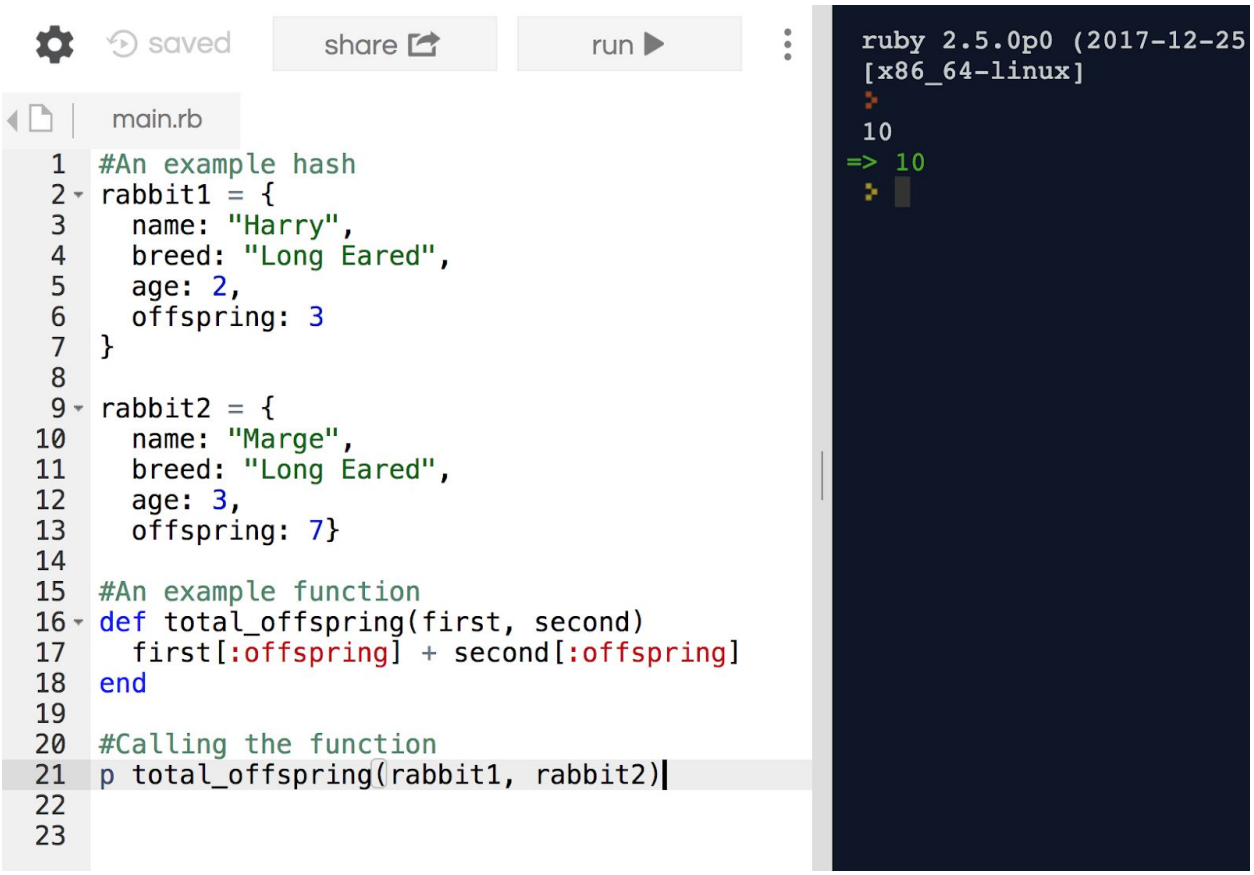
Evidence for unit




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

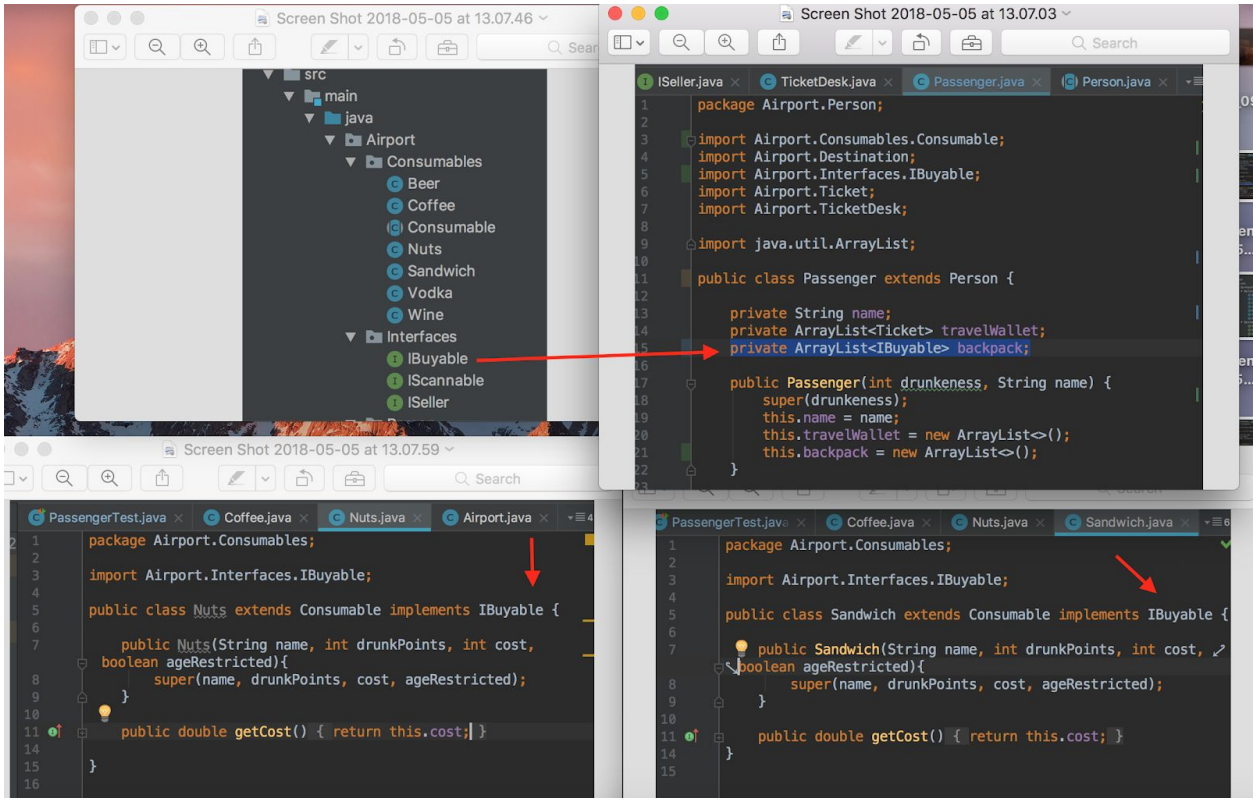


## I.T 6 - Example of a hash, a function that uses a hash and the result




## I.T 7 - Example of polymorphism in a program

In this Airport programme, Consumables classes such as Nuts and Sandwiches implement the IBuyable interface, allowing them to be collected within an ArrayList of IBuyables. Here the ArrayList is a ‘backpack’ and the Testing (second screenshot) shows that the interface is working.


Screen Shot 2018-05-05 at 13.07.46
src
main
java
Airport
Consumables
Beer
Coffee
Consumable
Nuts
Sandwich
Vodka
Wine
Interfaces
IBuyable
IScannable
ISeller
Screen Shot 2018-05-05 at 13.07.03
ISeller.java
TicketDesk.java
Passenger.java
Person.java
package Airport.Person;
import Airport.Consumables.Consumable;
import Airport.Destination;
import Airport.Interfaces.IBuyable;
import Airport.Ticket;
import Airport.TicketDesk;
import java.util.ArrayList;
public class Passenger extends Person {
private String name;
private ArrayList<Ticket> travelWallet;
private ArrayList<IBuyable> backpack;
public Passenger(int drunkeness, String name) {
super(drunkeness);
this.name = name;
this.travelWallet = new ArrayList<>();
this.backpack = new ArrayList<>();
}
Screen Shot 2018-05-05 at 13.07.59
PassengerTest.java
Coffee.java
Nuts.java
Airport.java
package Airport.Consumables;
import Airport.Interfaces.IBuyable;
public class Nuts extends Consumable implements IBuyable {
public Nuts(String name, int drunkPoints, int cost,
boolean ageRestricted){
super(name, drunkPoints, cost, ageRestricted);
}
public double getCost() { return this.cost; }
}
PassengerTest.java
Coffee.java
Nuts.java
Sandwich.java
package Airport.Consumables;
import Airport.Interfaces.IBuyable;
public class Sandwich extends Consumable implements IBuyable {
public Sandwich(String name, int drunkPoints, int cost,
boolean ageRestricted){
super(name, drunkPoints, cost, ageRestricted);
}
public double getCost() { return this.cost; }
}

PassengerTest.java | Coffee.java | Nuts.java

```java
( expected: 0, passenger2.howManyTicketsInWallet()); }

    @Test
    public void canAddTicketToWallet(){
        passenger1.addATicket(ticket1);
        assertEquals( expected: 1,
passenger1.howManyTicketsInWallet());
    }

    @Test
    public void canBuyTicket(){
        passenger1.buyTicket(Destination.MAGALUF, ticketDesk);
        assertEquals( expected: 1,
passenger1.howManyTicketsInWallet());
        assertEquals( expected: 1, ticketDesk
.howManyTicketsAreLeft());
    }

    @Test
    public void getPassengersTicketDestination(){
        passenger1.buyTicket(Destination.IBIZA, ticketDesk);
        assertEquals( expected: "Ibiza",
passenger1.getPassengersTicketDestination().getDestination()
);
    }

    @Test
    public void backpackStartsEmpty() { assertEquals
( expected: 0, passenger1.lookInBackpack()); }

    @Test public void backPackCanTakeIBuyables(){
        passenger1.putThingsInBackpack(coffee1);
        passenger1.putThingsInBackpack(sandwich1);
        assertEquals( expected: 2, passenger1.lookInBackpack());
    }
```

ISeller.java | TicketDesk.java | Passenger.java | Person.java

```java
package Airport.Person;

import Airport.Consumables.Consumable;
import Airport.Destination;
import Airport.Interfaces.IBuyable;
import Airport.Ticket;
import Airport.TicketDesk;

import java.util.ArrayList;

public class Passenger extends Person {

    private String name;
    private ArrayList<Ticket> travelWallet;
    private ArrayList<IBuyable> backpack;

    public Passenger(int drunkeness, String name) {
        super(drunkeness);
        this.name = name;
        this.travelWallet = new ArrayList<>();
        this.backpack = new ArrayList<>();
    }

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

    public int howManyTicketsInWallet() { return travelWallet
.size(); }

    public void addATicket(Ticket ticket) { this.travelWallet
.add(ticket); }

    public void buyTicket(Destination destination, TicketDesk
ticketDesk){
        this.travelWallet.add(ticketDesk.sellATicket
(destination));
    }
```

# I.T - Coding Exercise 1: Static & Dynamic Testing

Screen shot of static testing

testing_task_2_spec.rb testing_task_1.md

```ruby
require_relative('card.rb')
class CardGame

  def checkforAce(card)
    #should be check_for_ace
    if card.value = 1
      return true
    else
      return false
    end
  end

  dif highest_card(card1 card2)
  #def highest_card(card1, card2)
  if card1.value > card2.value
    return card.name
    #remove .name
  else
    card2
    #return card2?
  end
end
#end should be at the end of file
end

def self.cards_total(cards)
  total
  #total = 0
  for card in cards
    total += card.value
    return "You have a total of" + total
    #need space after of
  end
end
```

Screen shot of tests failing

card.rb | testing_task_2.rb

```ruby
# Carry out dynamic testing on the code below.
# Correct the errors below that you spotted in task 1.

require_relative('card.rb')

class CardGame

  def check_for_ace(card)
    if card.value == 1
      return true
    else
      return false
    end
  end

  def highest_card(card1, card2)
    if card1.value > card2.value
      return card.name
    else
      return card2
    end
  end

  def self.cards_total(cards)
    total = 0
    for card in cards
      total += card.value
      return "You have a total of #{total}."
    end
  end
```

PDA_Static_and_Dynamic_Task_A/testing_task_2.rb 31:8 LF UTF-8 Ruby 0 files

testing_task_2_spec.rb

```ruby
def setup

  @card1 = Card.new('diamonds', 5)
  @card2 = Card.new('spades', 1)
  card3 = Card.new('spades', 2)
  card4 = Card.new('spades', 7)

  @cards = [card3, card4]

  @cardgame1 = CardGame.new()

end

def test_check_for_ace
  assert_equal(true, @cardgame1.check_for_ace(@card2))

  assert_equal(false, @cardgame1.check_for_ace(@card1))
end

def test_highest_card
  assert_equal(@card1, @cardgame1.highest_card(@card2, @card1))
end

def test_cards_total
  assert_equal(9, CardGame.cards_total(@cards))
end

end
```

```
    /Users/user/codeclan/e20/workfiles/week_05/weekend_homework/PDA_Static_and_Dynamic_Task_A/testing_task_2.rb:28:in `each'
    /Users/user/codeclan/e20/workfiles/week_05/weekend_homework/PDA_Static_and_Dynamic_Task_A/testing_task_2.rb:28:in `cards_total'
    specs/testing_task_2_spec.rb:31:in `test_cards_total'

3 runs, 3 assertions, 0 failures, 1 errors, 0 skips
➜ PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
Run options: --seed 48537

# Running:

F..

Finished in 0.001088s, 2757.3529 runs/s, 3676.4706 assertions/s.

  1) Failure:
CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
Expected: 9
  Actual: "You have a total of."

3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
➜ PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
Run options: --seed 16308

# Running:

.F.

Finished in 0.001156s, 2595.1557 runs/s, 3460.2076 assertions/s.

  1) Failure:
CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
Expected: 9
  Actual: "You have a total of 2."

3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
➜ PDA_Static_and_Dynamic_Task_A
```

Screen shot of tests passing

card.rb | testing_task_2.rb

```ruby
# Carry out dynamic testing on the code below.
# Correct the errors below that you spotted in task 1.

require_relative('card.rb')

class CardGame

  def check_for_ace(card)
    if card.value == 1
      return true
    else
      return false
    end
  end

  def highest_card(card1, card2)
    if card1.value > card2.value
      return card.name
    else
      return card2
    end
  end

  def self.cards_total(cards)
    total = 0
    for card in cards
      total += card.value
    end
    return "You have a total of #{total}."
  end
```

PDA_Static_and_Dynamic_Task_A/specs/testing_task_2_spec.rb 31:41 LF UTF-8 Ruby 0 files

testing_task_2_spec.rb

```ruby
def setup

  @card1 = Card.new('diamonds', 5)
  @card2 = Card.new('spades', 1)
  card3 = Card.new('spades', 2)
  card4 = Card.new('spades', 7)

  @cards = [card3, card4]

  @cardgame1 = CardGame.new()

end

def test_check_for_ace
  assert_equal(true, @cardgame1.check_for_ace(@card2))

  assert_equal(false, @cardgame1.check_for_ace(@card1))
end

def test_highest_card
  assert_equal(@card1, @cardgame1.highest_card(@card2, @card1))
end

def test_cards_total
  assert_equal("You have a total of 9.", CardGame.cards_total(@cards))
end

end
```

```
# Running:

.F.

Finished in 0.001156s, 2595.1557 runs/s, 3460.2076 assertions/s.

  1) Failure:
CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
Expected: 9
  Actual: "You have a total of 2."

3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
➜ PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
Run options: --seed 47520

# Running:

F..

Finished in 0.001024s, 2929.6875 runs/s, 3906.2500 assertions/s.

  1) Failure:
CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
Expected: 9
  Actual: "You have a total of 9."

3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
➜ PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
Run options: --seed 31940

# Running:

...

Finished in 0.001103s, 2719.8549 runs/s, 3626.4732 assertions/s.

3 runs, 4 assertions, 0 failures, 0 errors, 0 skips
➜ PDA_Static_and_Dynamic_Task_A
```