

# 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()`.

```

1  package Airport;
2
3  import Airport.Person.Passenger;
4
5  import java.util.ArrayList;
6
7  public class Airport {
8
9      private String name;
10     private ArrayList<Plane> fleet;
11     private int maxPlanes;
12     private ArrayList<Passenger> tourists;
13
14     public Airport(String name, ArrayList<Plane> fleet, int maxPlanes){
15         this.name =name;
16         this.fleet = fleet;
17         this.maxPlanes = maxPlanes;
18         this.tourists = new ArrayList<>();
19     }
20
21     public String getName() { return this.name; }
22
23
24     public int getNumberOfPlanesInFleet() { return fleet.size(); }
25
26
27
28     public void planeLeaveAirport(Plane plane) { fleet.remove(plane); }
29
30
31
32
33     public void planeArrivesAtAirport(Plane plane){
34         if (fleet.size() < maxPlanes) {
35             fleet.add(plane);} else {
36                 System.out.println("Your airport is full!");
37             }
38     }
39
40 }

```

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.

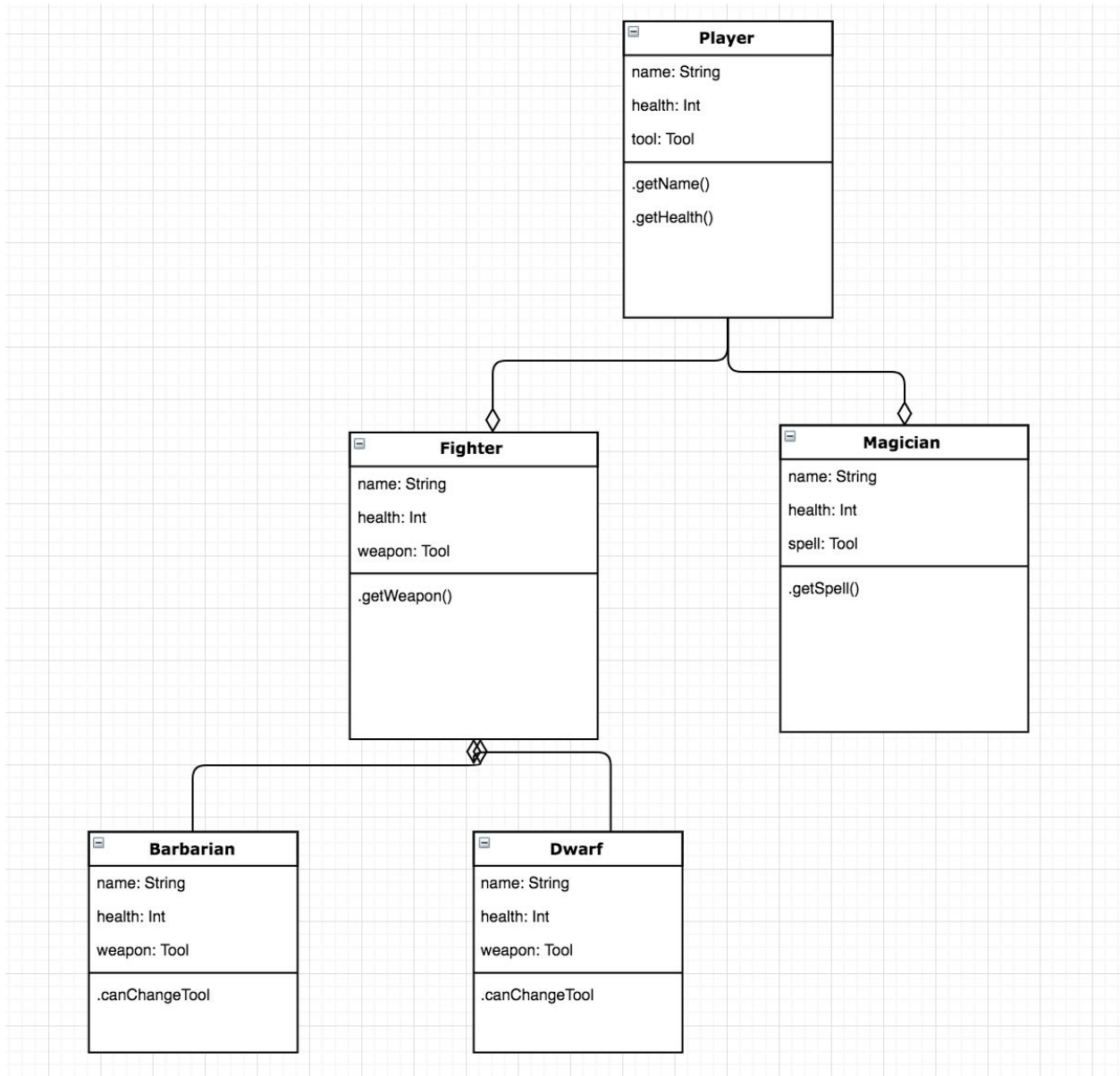
```

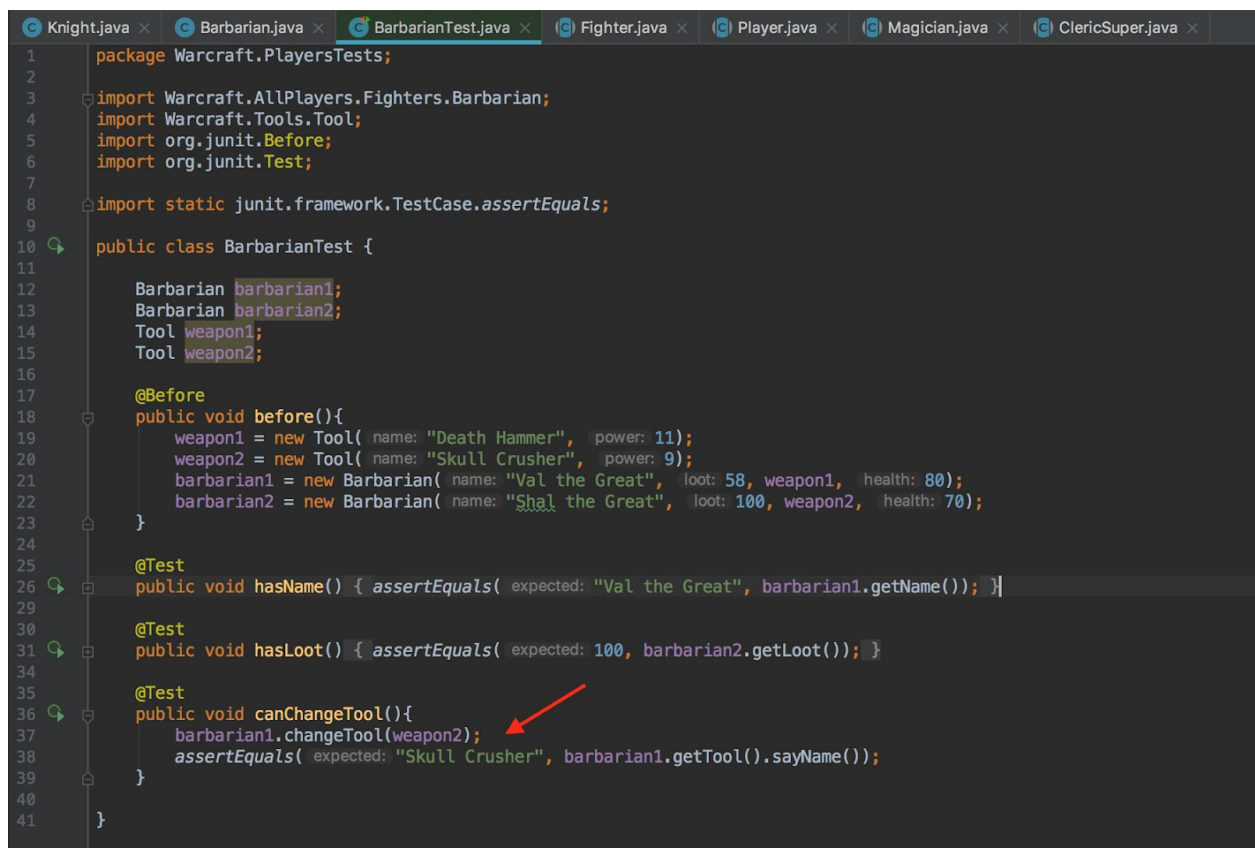
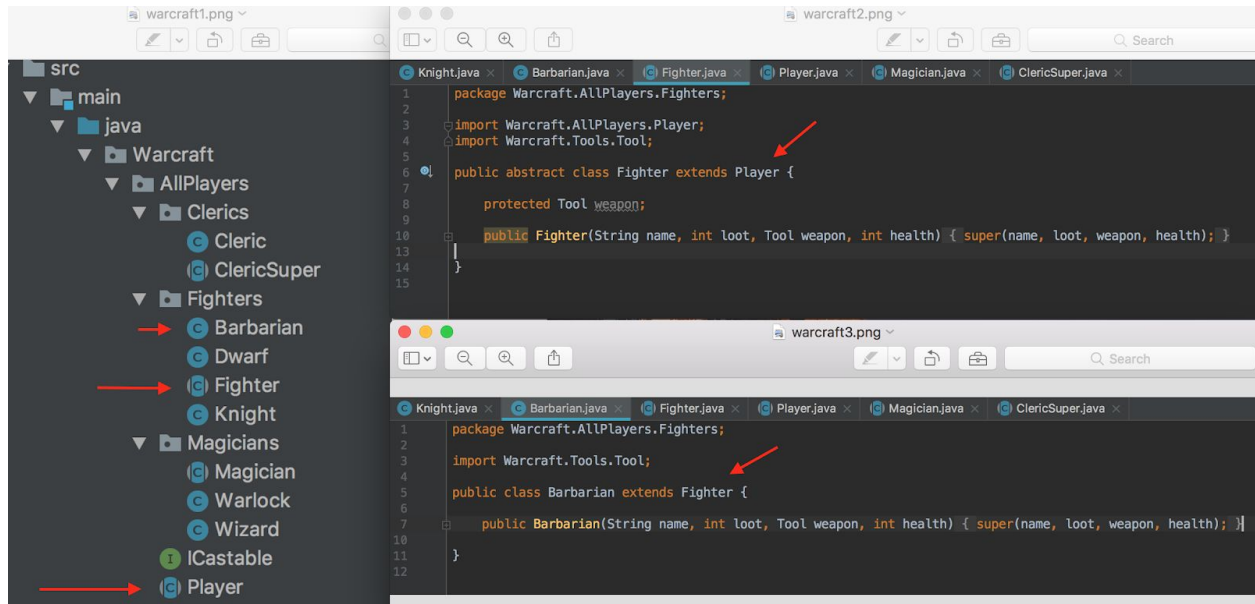
1  require_relative('../db/sql_runner')
2
3  class Troll
4
5      attr_reader :id
6      attr_accessor :name, :breed, :adoptable, :admission_date, :owner_id
7
8      def initialize( options )
9          @id = options['id'].to_i if options['id']
10         @name = options['name']
11         @breed = options['breed']
12         @adoptable = options['adoptable']
13         @admission_date = options['admission_date']
14         @owner_id = options['owner_id'] if options['owner_id']
15     end
16
17     def save
18
19         sql = "INSERT INTO trolls
20         (
21         name, breed, adoptable, admission_date, owner_id
22         )
23         VALUES
24         (
25         $1, $2, $3, $4, $5
26         )
27         RETURNING id"
28
29         values = [@name, @breed, @adoptable, @admission_date, @owner_id]
30         result = SqlRunner.run( sql, values )
31         @id = result.first['id']
32
33     end
34
35 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.

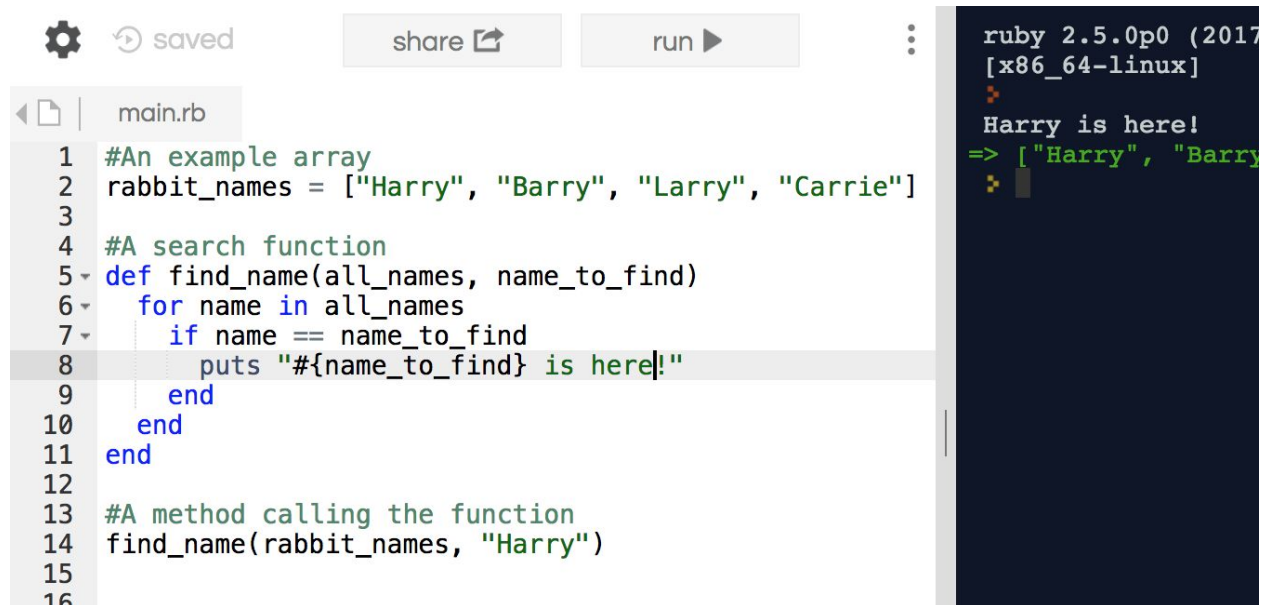




## I.T 3 - Example of searching

(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

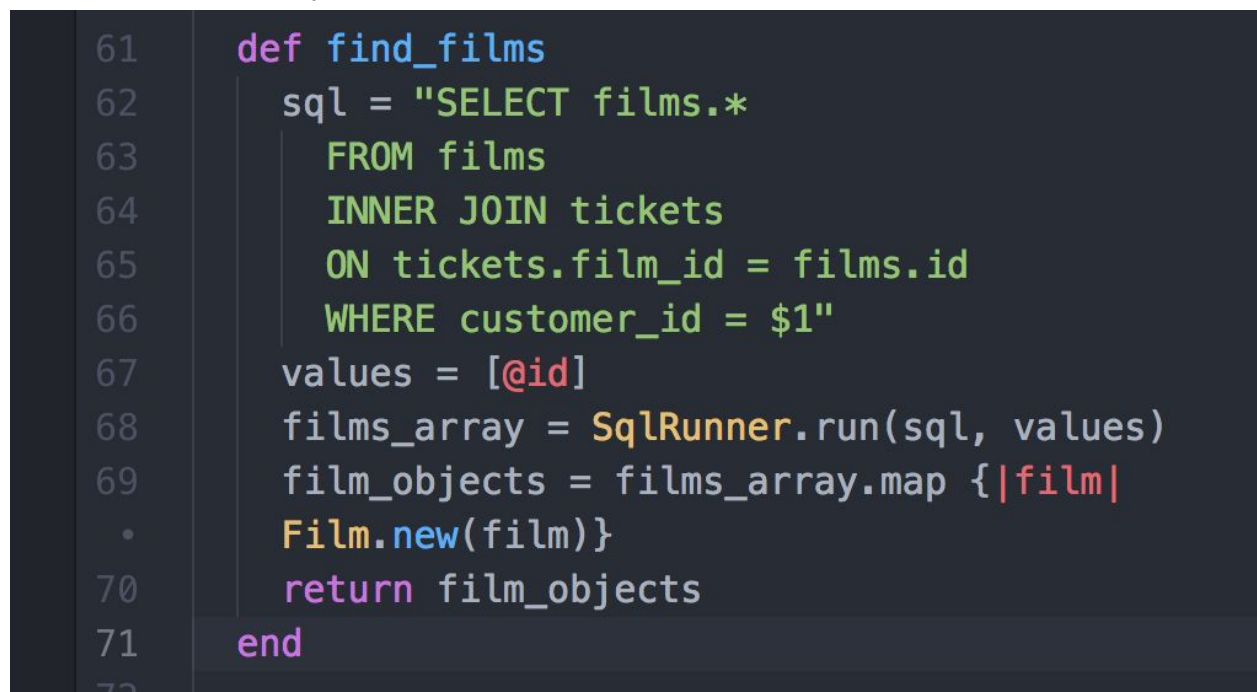


The screenshot shows a Ruby code editor with a file named `main.rb`. The code defines an array `rabbit_names` with the values `["Harry", "Barry", "Larry", "Carrie"]`. A function `find_name` is defined that takes `all_names` and `name_to_find` as arguments. It uses a `for` loop to iterate over `all_names` and an `if` statement to check if the current name matches `name_to_find`. If there is a match, it prints `"#{name_to_find} is here!"`. The function is then called with `find_name(rabbit_names, "Harry")`. To the right of the code editor, a terminal window shows the output of the program: `ruby 2.5.0p0 (2017...)`, `[x86_64-linux]`, and `Harry is here!`.

```
1 #An example array
2 rabbit_names = ["Harry", "Barry", "Larry", "Carrie"]
3
4 #A search function
5 def find_name(all_names, name_to_find)
6   for name in all_names
7     if name == name_to_find
8       puts "#{name_to_find} is here!"
9     end
10  end
11 end
12
13 #A method calling the function
14 find_name(rabbit_names, "Harry")
15
16
```

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

Evidence for unit: A ruby and SQL search function



The screenshot shows a Ruby code editor with a function `find_films`. The function takes `@id` as an argument and constructs an SQL query to select all columns from the `films` table, joined with the `tickets` table on the condition `tickets.film_id = films.id`, where `customer_id` is equal to `$1`. The query is executed using `SqlRunner.run` with the values `[@id]`. The results are mapped to `Film` objects and returned as an array.

```
61 def find_films
62   sql = "SELECT films.*
63         FROM films
64         INNER JOIN tickets
65         ON tickets.film_id = films.id
66         WHERE customer_id = $1"
67   values = [@id]
68   films_array = SqlRunner.run(sql, values)
69   film_objects = films_array.map {|film|
70     • Film.new(film)}
71   return film_objects
72 end
```

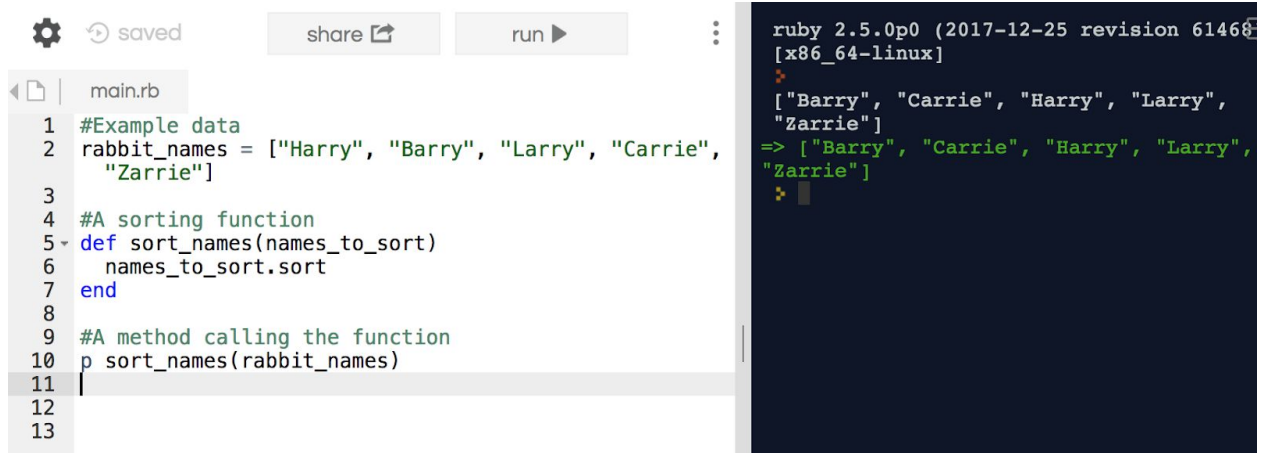
Evidence for unit: Terminal showing result of calling the Ruby/SQL function above



```
→ 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", @price="10", @id=112>]
```

## I.T 4 – Example of sorting

Evidence for unit



```

  1 #Example data
  2 rabbit_names = ["Harry", "Barry", "Larry", "Carrie", "Zarrie"]
  3
  4 #A sorting function
  5 def sort_names(names_to_sort)
  6   names_to_sort.sort
  7 end
  8
  9 #A method calling the function
 10 p sort_names(rabbit_names)
 11
 12
 13

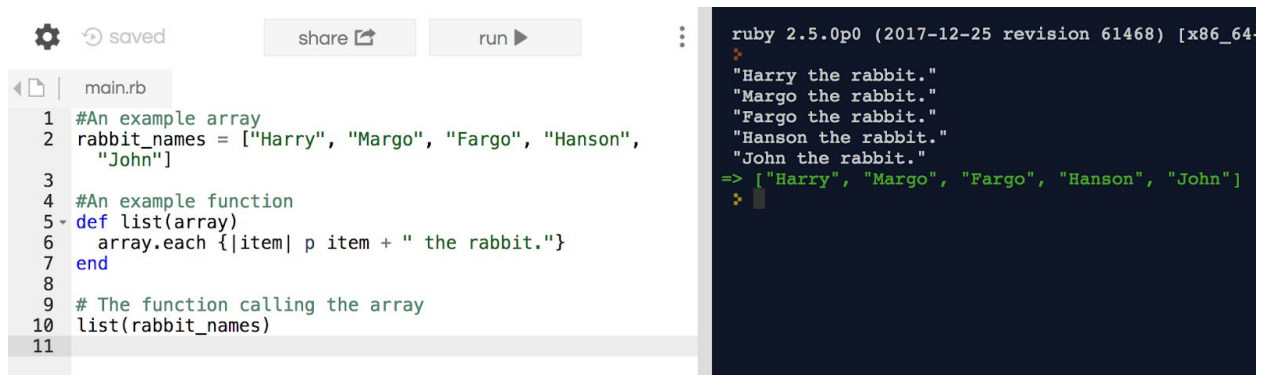
```

```

ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-linux]
>
["Barry", "Carrie", "Harry", "Larry", "Zarrie"]
=> ["Barry", "Carrie", "Harry", "Larry", "Zarrie"]
>

```

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



```

  1 #An example array
  2 rabbit_names = ["Harry", "Margo", "Fargo", "Hanson", "John"]
  3
  4 #An example function
  5 def list(array)
  6   array.each {|item| p item + " the rabbit."}
  7 end
  8
  9 # The function calling the array
 10 list(rabbit_names)
 11

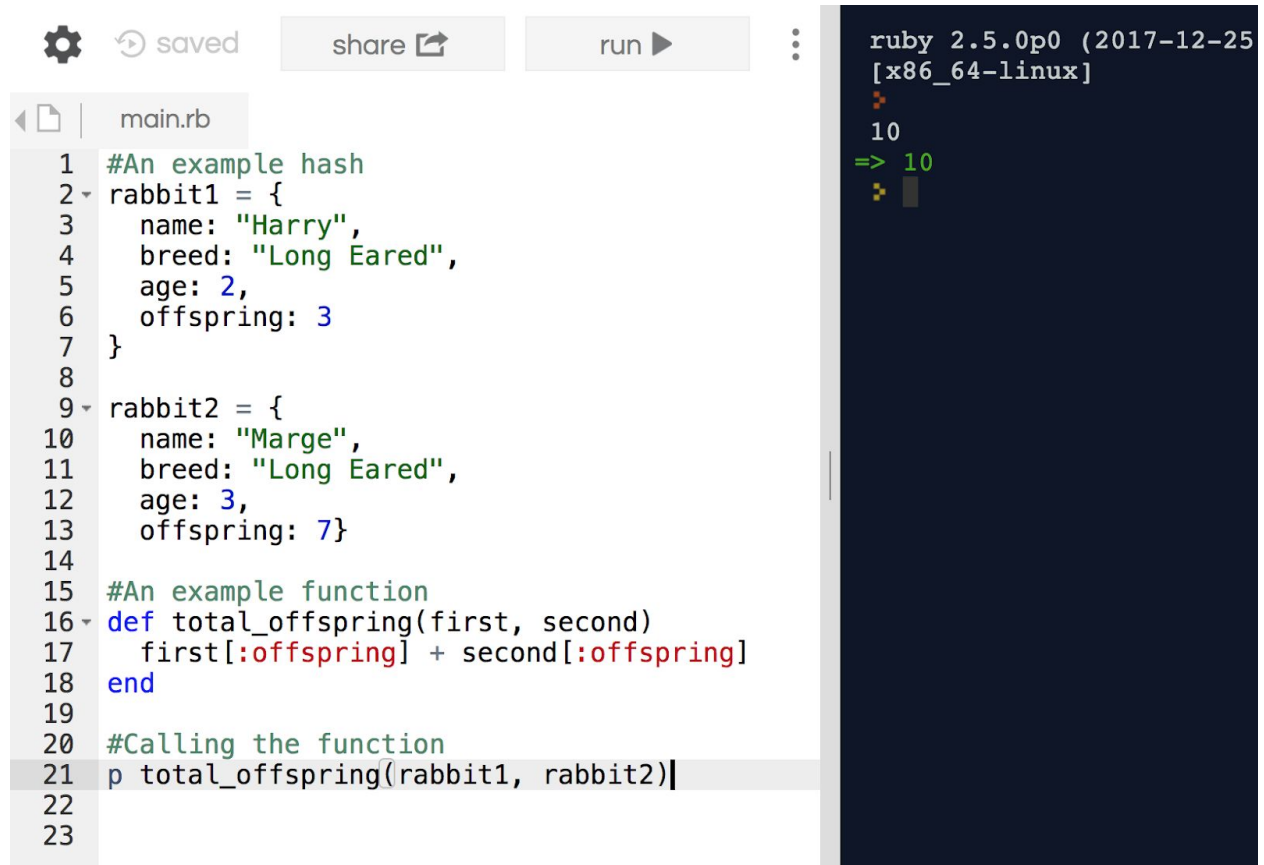
```

```

ruby 2.5.0p0 (2017-12-25 revision 61468) [x86_64-linux]
>
"Harry the rabbit."
"Margo the rabbit."
"Fargo the rabbit."
"Hanson the rabbit."
"John the rabbit."
=> ["Harry", "Margo", "Fargo", "Hanson", "John"]
>

```

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



The screenshot shows a Ruby IDE interface. At the top, there are icons for settings, a 'saved' status, a 'share' button, and a 'run' button. Below these is a file explorer showing 'main.rb'. The code in 'main.rb' is as follows:

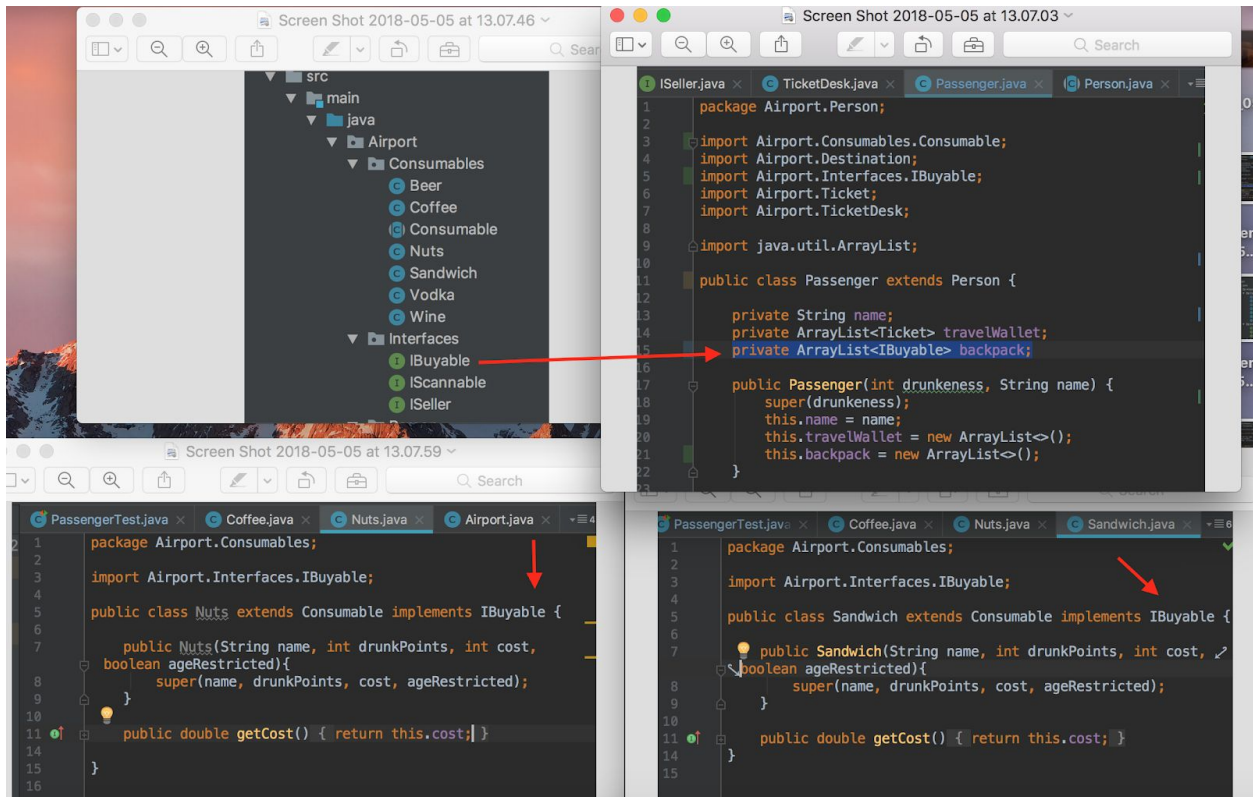
```
1 #An example hash
2 rabbit1 = {
3   name: "Harry",
4   breed: "Long Eared",
5   age: 2,
6   offspring: 3
7 }
8
9 rabbit2 = {
10  name: "Marge",
11  breed: "Long Eared",
12  age: 3,
13  offspring: 7}
14
15 #An example function
16 def total_offspring(first, second)
17   first[:offspring] + second[:offspring]
18 end
19
20 #Calling the function
21 p total_offspring(rabbit1, rabbit2)
22
23
```

On the right side of the IDE, the Ruby version and architecture are shown: 'ruby 2.5.0p0 (2017-12-25 [x86\_64-linux])'. Below this, the output of the program is displayed: '10' and '=> 10'.

## 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.





```
PassengerTest.java | Coffee.java | Nuts.java | ISeller.java | TicketDesk.java | Passenger.java | Person.java
50 | ( expected: 0, passenger2.howManyTicketsInWallet()); }
51 |
52 | @Test
53 | public void canAddTicketToWallet(){
54 |     passenger1.addATicket(ticket1);
55 |     assertEquals( expected: 1,
56 |     passenger1.howManyTicketsInWallet());
57 | }
58 |
59 | @Test
60 | public void canBuyTicket(){
61 |     passenger1.buyTicket(Destination.MAGALUF, ticketDesk);
62 |     assertEquals( expected: 1,
63 |     passenger1.howManyTicketsInWallet());
64 |     assertEquals( expected: 1, ticketDesk
65 |     .howManyTicketsAreLeft());
66 | }
67 |
68 | @Test
69 | public void getPassengersTicketDestination(){
70 |     passenger1.buyTicket(Destination.IBIZA, ticketDesk);
71 |     assertEquals( expected: "Ibiza",
72 |     passenger1.getPassengersTicketDestination().getDestination()
73 |     );
74 | }
75 |
76 | @Test
77 | public void backpackStartsEmpty() { assertEquals
78 | ( expected: 0, passenger1.lookInBackpack()); }
79 |
80 |
81 | @Test public void backPackCanTakeIBuyables(){
82 |     passenger1.putThingsInBackpack(coffee1);
83 |     passenger1.putThingsInBackpack(sandwich1);
84 |     assertEquals( expected: 2, passenger1.lookInBackpack());
85 | }

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 drunkenness, String name) {
        super(drunkenness);
        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
11 require_relative('card.rb')
12 class CardGame
13
14   def checkforAce(card)
15     #should be check_for_ace
16     if card.value = 1
17       return true
18     else
19       return false
20     end
21   end
22
23   def highest_card(card1 card2)
24     #def highest_card(card1, card2)
25     if card1.value > card2.value
26       return card.name
27       #remove .name
28     else
29       card2
30       #return card2?
31     end
32   end
33   #end should be at the end of file
34 end
35
36 def self.cards_total(cards)
37   total
38   #total = 0
39   for card in cards
40     total += card.value
41     return "You have a total of" + total
42     #need space after of
43   end
44 end
```

Screen shot of tests failing

```
card.rb      testing_task_2.rb      testing_task_2_spec.rb
3 # Carry out dynamic testing on
4 # the code below.
5 # Correct the errors below that
6 # you spotted in task 1.
7
8 require_relative('card.rb')
9
10 class CardGame
11   def check_for_ace(card)
12     if card.value == 1
13       return true
14     else
15       return false
16     end
17   end
18
19   def highest_card(card1, card2)
20     if card1.value > card2.value
21       return card.name
22     else
23       return card2
24     end
25   end
26
27   def self.cards_total(cards)
28     total = 0
29     for card in cards
30       total += card.value
31     end
32     return "You have a total of #{total}."
33   end
34 end
35
36 testing_task_2_spec.rb
37
38 def setup
39   @card1 = Card.new('diamonds', 5)
40   @card2 = Card.new('spades', 1)
41   @card3 = Card.new('spades', 2)
42   @card4 = Card.new('spades', 7)
43
44   @cards = [card3, card4]
45
46   @cardgame1 = CardGame.new()
47 end
48
49 def test_check_for_ace
50   assert_equal(true, @cardgame1.check_for_ace(@card2))
51 end
52
53 def test_highest_card
54   assert_equal(false, @cardgame1.check_for_ace(@card1))
55 end
56
57 def test_cards_total
58   assert_equal(9, CardGame.cards_total(@cards))
59 end
60
61 # Running:
62
63 .F..
64
65 Finished in 0.001088s, 2757.3529 runs/s, 3676.4706 assertions /s.
66
67 1) Failure:
68 CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
69 Expected: 9
70 Actual: "You have a total of 2."
71
72 3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
73
74 PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
75
76 Run options: --seed 48537
77
78 # Running:
79
80 .F..
81
82 Finished in 0.001156s, 2595.1557 runs/s, 3460.2076 assertions /s.
83
84 1) Failure:
85 CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
86 Expected: 9
87 Actual: "You have a total of 2."
88
89 3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
90
91 PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
92
93 Run options: --seed 16308
94
95 # Running:
96
97 .F..
98
99 Finished in 0.001156s, 2595.1557 runs/s, 3460.2076 assertions /s.
100
101 1) Failure:
102 CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
103 Expected: 9
104 Actual: "You have a total of 2."
105
106 3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
107
108 PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
109
110 Run options: --seed 48537
```

Screen shot of tests passing

```
card.rb      testing_task_2.rb      testing_task_2_spec.rb
3 # Carry out dynamic testing on
4 # the code below.
5 # Correct the errors below that
6 # you spotted in task 1.
7
8 require_relative('card.rb')
9
10 class CardGame
11   def check_for_ace(card)
12     if card.value == 1
13       return true
14     else
15       return false
16     end
17   end
18
19   def highest_card(card1, card2)
20     if card1.value > card2.value
21       return card.name
22     else
23       return card2
24     end
25   end
26
27   def self.cards_total(cards)
28     total = 0
29     for card in cards
30       total += card.value
31     end
32     return "You have a total of
33     #{total}."
34   end
35 end
36
37 testing_task_2_spec.rb
38
39 def setup
40   @card1 = Card.new('diamonds', 5)
41   @card2 = Card.new('spades', 1)
42   @card3 = Card.new('spades', 2)
43   @card4 = Card.new('spades', 7)
44
45   @cards = [card3, card4]
46
47   @cardgame1 = CardGame.new()
48 end
49
50 def test_check_for_ace
51   assert_equal(true, @cardgame1.check_for_ace(@card2))
52 end
53
54 def test_highest_card
55   assert_equal(false, @cardgame1.check_for_ace(@card1))
56 end
57
58 def test_cards_total
59   assert_equal("You have a total of
60   9.", CardGame.cards_total(@cards))
61 end
62
63 # Running:
64
65 .F..
66
67 Finished in 0.001156s, 2595.1557 runs/s, 3460.2076 assertions /s.
68
69 1) Failure:
70 CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
71 Expected: 9
72 Actual: "You have a total of 2."
73
74 3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
75
76 PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
77
78 Run options: --seed 47520
79
80 # Running:
81
82 .F..
83
84 Finished in 0.001024s, 2929.6875 runs/s, 3906.2500 assertions /s.
85
86 1) Failure:
87 CardGameTest#test_cards_total [specs/testing_task_2_spec.rb:31]:
88 Expected: 9
89 Actual: "You have a total of 9."
90
91 3 runs, 4 assertions, 1 failures, 0 errors, 0 skips
92
93 PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
94
95 Run options: --seed 31940
96
97 # Running:
98
99 ...
100
101 Finished in 0.001103s, 2719.8549 runs/s, 3626.4732 assertions /s.
102
103 3 runs, 4 assertions, 0 failures, 0 errors, 0 skips
104
105 PDA_Static_and_Dynamic_Task_A ruby specs/testing_task_2_spec.rb
106
107 Run options: --seed 31940
```