

Evidence for Implementation and Testing Unit

Ross Hill
Cohort E18

I.T. 1 – Encapsulation in a program

```
class Rooms

  attr_reader :room_name, :room_capacity, :room_fee, :songs_in_room

  def initialize (room_name, room_capacity, room_fee, songs_in_room)
    @room_name = room_name
    @guests_in_room = []
    @songs_in_room = []
    @room_capacity = room_capacity
    @room_fee = room_fee
  end

  def guest_count
    @guests_in_room.length()
  end

  def add_guest_to_room(guest_name)
    @guests_in_room << guest_name
  end

  def remove_guest_from_room(guest_name)
    @guests_in_room.delete(guest_name)
  end

  def room_capacity_check
    if guest_count > @room_capacity
      return "Room full!"
    else
      return guest_count
    end
  end
end
```

I.T. 2 – Inheritance in a program

```
package Player.Fighters;
import Actions.IAttackable;
import NPC.Enemy;

public abstract class Fighters extends Player.Player implements IAttackable {

    private Weapon weapon;

    public Fighters(String name, int healthPoints) {
        super(name, healthPoints);
        this.weapon = null;
    }

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

    public void attack(Enemy enemy) {
        int enemyHealth = enemy.getHealth();
        int weaponValue = weapon.getValue();
        enemy.setHealth(enemyHealth - weaponValue);
    }
}
```

```
package Player.Fighters;

public class Knight extends Fighters {

    public Knight(String name, int healthPoints){
        super(name, healthPoints);
    }

}
```

```
import NPC.Orc;
import Player.Fighters.Barbarian;
import Player.Fighters.Dwarf;
import Player.Fighters.Knight;
import Player.Fighters.Weapon;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.assertEquals;
```

Run Test | Debug Test

```
public class PlayerTest {
```

```
    Knight knight;
    Barbarian barbarian;
    Dwarf dwarf;
```

@Before

```
public void before(){
    knight = new Knight("Arthur", 400);
    barbarian = new Barbarian("Conan", 500);
    dwarf = new Dwarf("Gimmlly", 350);
}
```

@Test

Run Test | Debug Test

```
public void hasName() {
    assertEquals("Arthur", knight.getName());
    assertEquals("Conan", barbarian.getName());
    assertEquals("Gimmlly", dwarf.getName());
}
```

@Test

Run Test | Debug Test

```
public void knightCanAttackOrc() {
    Orc orc = new Orc("Mr. Orc", 400);
    knight.setWeapon(Weapon.SWORD);
    knight.attack(orc);
    assertEquals(350, orc.getHealth());
}
```

I.T. 3 – Searching in a program

```
def Artist.find(id)
  sql = "SELECT * FROM artists WHERE id = $1"
  result = SqlRunner.run(sql, [id])
  return result.map {|artist| Artist.new(artist)}
end
```

```
[7] pry(main)> Artist.find(2)
=> [#<Artist:0x007f9ad7aa3768 @id=2, @name="Hot Snakes">]
[8] pry(main)>
```

I.T. 4 – Sorting data in a program

```
examples.rb
1 birds = ["sparrow", "robin", "pigeon", "magpie", "penguin"]
2
3 def sort_A_to_Z(array)
4   p array.sort
5 end
6
7 sort_A_to_Z(birds)
8
```

```
→ pda git:(master) x ruby examples.rb
["magpie", "penguin", "pigeon", "robin", "sparrow"]
→ pda git:(master) x
```

I.T. 5 – Use of an array

```
fruits = ['banana', 'apple', 'orange']
```

```

1  def add_fruit(new_fruit)
2    fruits = ['banana', 'apple', 'orange']
3    fruits.push(new_fruit)
4    p fruits
5  end
6
7  add_fruit('avocado')
8

```

```

[→ pda git:(master) x ruby examples.rb
["banana", "apple", "orange", "avocado"]
→ pda git:(master) x █

```

I.T. 6 – Use of a hash

```

9  @customer =
10   {
11     name: "Ross",
12     pets: [],
13     cash: 1000
14   }
15
16  def add_cash(add_amount)
17    @customer[:cash] += add_amount
18    p @customer
19  end
20
21  add_cash(800)
22

```

```

[→ pda git:(master) x ruby examples.rb
{:name=>"Ross", :pets=>[], :cash=>1800}
→ pda git:(master) x █

```

I.T. 7 – Use of Polymorphism

```
1  import Accessories.ISell;
2
3  import java.util.ArrayList;
4
5  public class Shop implements Accessories.ISell{
6
7      ArrayList<ISell> stock;
8
9      public Shop(){
10         this.stock = new ArrayList<>();
11     }
12
13     public int stockCount() {
14         return stock.size();
15     }
16
17     public void addItem(ISell stockItem) {
18         this.stock.add(stockItem);
19     }
20 }
```