

Evidence for Implementation and Testing.

Name: Benjamin R Conway

Cohort: E15

Date: 05/10/2017

Benjamin R Conway
I.T. 1 Encapsulation in a program.

Abstract Class Flora
has private methods
which cannot be
directly set by users,
however can be
modified through
specified methods.

```
public abstract class Flora {  
    private int height;  
    private int width;  
    private String bark;  
    private Enum<Habit> growthHabit;  
  
    public Flora( int height, int width, String bark, Enum<Habit> growthHabit){  
        this.height = height;  
        this.width = width;  
        this.bark = bark;  
        this.growthHabit = growthHabit;  
    }  
  
    public int getHeight() { return height; }  
    public int getWidth() { return width; }  
    public String feelTheBark() {  
        return "the bark of this plant feels " + bark + " on your skin.";  
    }  
    public Enum<Habit> getGrowthHabit() {  
        return growthHabit;  
    }  
  
    public void pruneTop(int amountPruned){  
        this.height = this.height - amountPruned;  
    }  
    public void pruneSides(int amountPruned){  
        this.width = this.width - amountPruned;  
    }  
}
```

Benjamin R Conway
I.T. 2 Inheritance in a Program.

```
public abstract class Flora {
    private int height;
    private int width;
    private String bark;
    private Enum<Habit> growthHabit;

    public Flora( int height, int width, String bark, Enum<Habit> growthHabit){
        this.height = height;
        this.width = width;
        this.bark = bark;
        this.growthHabit = growthHabit;
    }

    public int getHeight() { return height; }
    public int getWidth() { return width; }
    public String feelTheBark() {
        return "the bark of this plant feels " + bark + " on your skin.";
    }
    public Enum<Habit> getGrowthHabit() { return growthHabit; }
    public void pruneTop(int amountPruned) { this.height = this.height - amountPruned; }
    public void pruneSides(int amountPruned) { this.width = this.width - amountPruned; }
}
```

Class Angiosperm inherits from
Abstract Class Flora

```
public class Angiosperm extends Flora implements Flowering {
    private String name;
    Enum<FlowerType> flowers;

    public Angiosperm(int height, int width, String bark, Enum<Habit> growthHabit,
        String name, Enum<FlowerType> flowers) {
        super(height, width, bark, growthHabit);
        this.name = name;
        this.flowers = flowers;
    }

    public String smellFlowers(){
        return "You nose is filled with the sweet scent of " + name + " flowers.";
    }

    public String pickFruit(){
        return "Your mouth waters as you reach for fresh fruit off the branch";
    }
}
```

This Runner creates an
Object of Class Angiosperm

```
public class Runner {
    public static void main(String[] args) {
        Angiosperm plant1 = new Angiosperm( 3, 4, "smooth", Habit.SHRUB,
            "Gardenia", FlowerType.POLYPETALOUS);
        System.out.println(plant1.feelTheBark());
    }
}
```

Result of calling a method
inherited from the Parent Class

PDA — user@users-MacBook-Pro — ..clan_work/I
[→ PDA java Runner
the bark of this plant feels smooth on your skin.
→ PDA █

I.T. 3: Searching data in a program.

```
sponsorship.rb
72
73
74 #Function that searches data.
75 def Sponsorship.find(id)
76   sql = "SELECT * FROM sponsorships WHERE id = $1;"
77   result = SqlRunner.run(sql, [id])
78   return Sponsorship.new(result.first)
79 end
80
81
82 end
83

sqlRunner.rb
1 require("pg")
2
3 class SqlRunner
4
5   def SqlRunner.run( sql, values = [] )
6     begin
7       db = PG.connect({ dbname:"conservation", host:"localhost" })
8       db.prepare( "query", sql )
9       result = db.exec_prepared( "query", values)
10      ensure
11        db.close
12      end
13      return result
14    end
15  end
16 end
17

[1] pry(main)> Sponsorship.find(3)
=> #<Sponsorship:0x007fd86fc94a28 @animal_id=3, @date_sponsored="2015-07-18", @id=3, @member_id=1, @value=1250>
[2] pry(main)>
```

```
conservation=# SELECT * FROM sponsorships;
id | animal_id | member_id | value | date_sponsored
---|---|---|---|---
1 | 1 | 3 | 500 | 2017-04-18
2 | 2 | 2 | 5000 | 2016-01-12
3 | 3 | 1 | 1250 | 2015-07-18
4 | 2 | 4 | 15 | 2017-08-28
(4 rows)

conservation=#
```

I.T. 4: Sorting data in a program.

```
sponsorship.rb
76 end
77
78
79
80 #Function that sorts data.
81
82 def Sponsorship.by_date()
83   sql = "SELECT * FROM sponsorships ORDER BY date_sponsored;"
84   result = SqlRunner.run(sql)
85   sponsorships_array = result.map() { |data| Sponsorship.new(data)}
86   return sponsorships_array
87 end
88
89 end
90

sqlRunner.rb
1 require("pg")
2
3 class SqlRunner
4
5   def SqlRunner.run( sql, values = [] )
6     begin
7       db = PG.connect({ dbname:"conservation", host:"localhost" })
8       db.prepare( "query", sql )
9       result = db.exec_prepared( "query", values)
10      ensure
11        db.close
12      end
13      return result
14    end
15  end
16 end
17

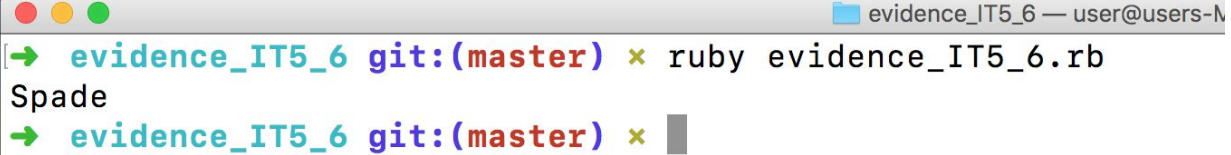
[1] pry(main)> Sponsorship.by_date
=> [#<Sponsorship:0x007fdc7a404990 @animal_id=3, @date_sponsored="2015-07-18", @id=3, @member_id=1, @value=1250>,
#<Sponsorship:0x007fdc7a4046e8 @animal_id=2, @date_sponsored="2016-01-12", @id=2, @member_id=2, @value=5000>,
#<Sponsorship:0x007fdc7a404588 @animal_id=1, @date_sponsored="2017-04-18", @id=1, @member_id=3, @value=500>,
#<Sponsorship:0x007fdc7a404328 @animal_id=2, @date_sponsored="2017-08-28", @id=4, @member_id=4, @value=15>]
[2] pry(main)>
```

```
conservation=# SELECT * FROM sponsorships;
id | animal_id | member_id | value | date_sponsored
---|---|---|---|---
1 | 1 | 3 | 500 | 2017-04-18
2 | 2 | 2 | 5000 | 2016-01-12
3 | 3 | 1 | 1250 | 2015-07-18
4 | 2 | 4 | 15 | 2017-08-28
(4 rows)

conservation=#
```

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

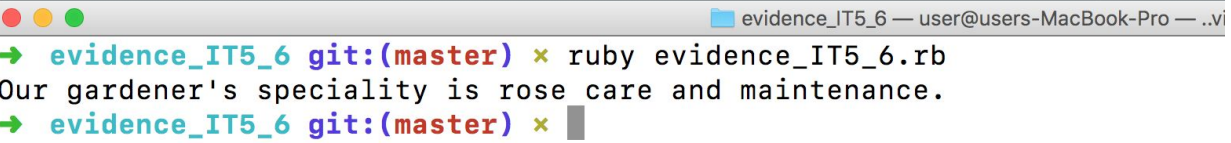
```
1 tool_library = ["Spade", "Ladder", "Dutch Hoe", "Rubber Mallet"]
2
3 def first_tool(tool_array)
4   return tool_array[0]
5 end
6
7 puts first_tool(tool_library)
```



```
→ evidence_IT5_6 git:(master) × ruby evidence_IT5_6.rb
Spade
→ evidence_IT5_6 git:(master) ×
```

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

```
1 gardener1 = {
2   name: "Bob",
3   speciality: "Rose care and maintenance",
4   favourite_tool: "Secateurs"
5 }
6
7 def speciality(gardener_hash)
8   return gardener_hash[:speciality]
9 end
10
11 puts "Our gardener's speciality is #{speciality(gardener1).downcase}."
```



```
→ evidence_IT5_6 git:(master) × ruby evidence_IT5_6.rb
Our gardener's speciality is rose care and maintenance.
→ evidence_IT5_6 git:(master) ×
```

Benjamin R Conway
I.T. 7 Polymorphism in a Program

The Garden Class has an ArrayList that takes in objects implementing the Plantable interface.

```
public class Garden {  
    private ArrayList<Plantable> gardenBed;  
  
    public Garden() { gardenBed = new ArrayList<>(); }  
  
    public ArrayList<Plantable> getGardenBed() { return gardenBed; }  
  
    public void addPlantToBed(Plantable plantable) { gardenBed.add(plantable); }  
}
```

```
public interface Plantable {  
}
```

The two classes of Angiosperm and Gymnosperm implement the Plantable interface.

```
public class Angiosperm implements Plantable {  
}
```

In this Runner file, three objects are instantiated, one of each class, Garden, Angiosperm and Gymnosperm.

```
public class Gymnosperm implements Plantable {  
}
```

The initial size is declared in the first output. The two objects of different classes that implement the Plantable interface are then added to the ArrayList.

```
public class Runner {  
    public static void main(String[] args){  
        Garden garden = new Garden();  
        Angiosperm flower = new Angiosperm();  
        Gymnosperm pine = new Gymnosperm();  
  
        //Show the array is empty of objects  
        System.out.println("My garden has " + garden.getGardenBed().size() + " plants.");  
  
        //Add the Angiosperm class object to the Garden class ArrayList<Plant>.  
        garden.addPlantToBed(flower);  
  
        //Add the Gymnosperm class object to the Garden class ArrayList<Plant>  
        garden.addPlantToBed(pine);  
  
        //Proof of Polymorphism  
        System.out.println("My garden now has " + garden.getGardenBed().size() + " plants.");  
    }  
}
```

The second output proves Polymorphism by showing that the size of the ArrayList has increased to match the number of objects presently added.

```
Runner  
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...  
My garden has 0 plants.  
My garden now has 2 plants.  
  
Process finished with exit code 0
```