**PDA: Evidence for Implementation and Testing Unit**

**Verity Ashforth**
**E19**

**IT1 - Demonstrate one example of encapsulation you have written in a program**

**IT2 - Example the use of inheritance in a program**

**IT3 - Example of Searching**

my_money.sql | show.erb

```
DROP TABLE budget;
DROP TABLE transactions;
DROP TABLE tags;
DROP TABLE merchants;

CREATE TABLE merchants (
  id SERIAL4 PRIMARY KEY,
  merchant_name VARCHAR(255) NOT NULL
);

CREATE TABLE tags (
  id SERIAL4 PRIMARY KEY,
  tag_name VARCHAR(255) NOT NULL
);

CREATE TABLE transactions (
  id SERIAL4 PRIMARY KEY,
  amount NUMERIC,
  merchant_id INT4 REFERENCES merchants,
  tag_id INT4 REFERENCES tags,
  trans_date DATE
);

CREATE TABLE budget (
  id SERIAL4 PRIMARY KEY,
  weekly NUMERIC,
  monthly NUMERIC,
  yearly NUMERIC
);
```

To the left is a screenshot of a table created for the database ‘my-money’ using PostgreSQL (PSQL).

Directly below is a screenshot of the transactions table sitting in PSQL, viewed in the terminal.

```
 id  | amount | merchant_id | tag_id | trans_date
-----+--------+-------------+--------+------------
 815 |   34.5 |         324 |    396 | 2018-01-30
 816 |   7.95 |         325 |    393 | 2018-02-09
 817 |   60.2 |         323 |    395 | 2018-03-19
 818 |  12.79 |         322 |    393 | 2018-11-01
 819 |  15.45 |         325 |    397 | 2018-01-15
 820 |   34.5 |         324 |    398 | 2018-01-30
 821 |   7.95 |         325 |    393 | 2018-02-09
 822 |   60.2 |         323 |    395 | 2018-03-19
 823 |  12.79 |         322 |    398 | 2018-11-01
 824 |  15.45 |         325 |    395 | 2018-01-15
 825 |   34.5 |         324 |    394 | 2018-01-30
 826 |   7.95 |         325 |    393 | 2018-02-09
 827 |   60.2 |         323 |    399 | 2018-03-19
 828 |  12.79 |         322 |    399 | 2018-11-01
 829 |  15.45 |         325 |    395 | 2018-01-15
(15 rows)
```

Below is a screenshot of function written in Ruby which will search the transaction table and pull back all transactions of a specified tag (or transaction type) ID. Ruby is able to communicate with PSQL using Sinatra.

```
def Transaction.show_by_type(id)
  sql = "SELECT * FROM transactions
  WHERE tag_id = $1
  ORDER BY trans_date DESC;"
  transactions = SqlRunner.run(sql, [id])
  return transactions.map { |transaction| Transaction.new(transaction)  }
end
```

```
[2] pry(main)> Transaction.show_by_type(395)
=> [#<Transaction:0x007fadd9429e18 @amount=60.2, @id=817, @merchant_id=323, @tag_id=395, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd9429a58 @amount=60.2, @id=822, @merchant_id=323, @tag_id=395, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd9429918 @amount=15.45, @id=824, @merchant_id=325, @tag_id=395, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd94297b0 @amount=15.45, @id=829, @merchant_id=325, @tag_id=395, @trans_date="2018-01-15">]
[3] pry(main)>
```

Running the function Transaction.show_by_type() in the terminal using PRY we can see above that the function returns all transactions with tag_id = 395 as expected.

**IT4 - Example of Sorting**

```ruby
def Transaction.show_all()
  sql = "SELECT * FROM transactions
  ORDER BY trans_date DESC;"
  transactions = SqlRunner.run(sql)
  return transactions.map { |transaction| Transaction.new(transaction)  }
end
```

The function above, written in Ruby, extracts all transactions from the transaction table in PSQL and returns them as instances of the defined class Transaction in an array like fashion. As shown below we can assign this a variable name ‘transactions’.

```
[3] pry(main)> transactions = Transaction.show_all()
=> [#<Transaction:0x007fadd903f2d0 @amount=12.79, @id=823, @merchant_id=322, @tag_id=398, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903f168 @amount=12.79, @id=818, @merchant_id=322, @tag_id=393, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903f028 @amount=12.79, @id=828, @merchant_id=322, @tag_id=399, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903eee8 @amount=60.2, @id=822, @merchant_id=323, @tag_id=395, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd903ed58 @amount=60.2, @id=827, @merchant_id=323, @tag_id=399, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd903ebc8 @amount=60.2, @id=817, @merchant_id=323, @tag_id=395, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd903eab0 @amount=7.95, @id=816, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">,
 #<Transaction:0x007fadd903e998 @amount=7.95, @id=821, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">,
 #<Transaction:0x007fadd903e880 @amount=7.95, @id=826, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">,
 #<Transaction:0x007fadd903e768 @amount=34.5, @id=815, @merchant_id=324, @tag_id=396, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903e650 @amount=34.5, @id=820, @merchant_id=324, @tag_id=398, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903e538 @amount=34.5, @id=825, @merchant_id=324, @tag_id=394, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903e3d0 @amount=15.45, @id=824, @merchant_id=325, @tag_id=395, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd903e2b8 @amount=15.45, @id=819, @merchant_id=325, @tag_id=397, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd903e178 @amount=15.45, @id=829, @merchant_id=325, @tag_id=395, @trans_date="2018-01-15">]
```

In Ruby I have written the following function which will sort these transactions by amount either ascending or descending:

```ruby
def Transaction.order_by_amount(transactions, direction)
  transactions_asc = transactions.sort_by {|transaction| transaction.amount }
  transactions_desc = transactions_asc.reverse
  return transactions_asc if direction == "asc"
  return transactions_desc
end
```

Below is what is the result of this function, which sorts the array-like entity transactions as expected:

```
[4] pry(main)> Transaction.order_by_amount(transactions, 'asc')
=> [#<Transaction:0x007fadd903e880 @amount=7.95, @id=826, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">,
 #<Transaction:0x007fadd903e998 @amount=7.95, @id=821, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">,
 #<Transaction:0x007fadd903eab0 @amount=7.95, @id=816, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">,
 #<Transaction:0x007fadd903f2d0 @amount=12.79, @id=823, @merchant_id=322, @tag_id=398, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903f168 @amount=12.79, @id=818, @merchant_id=322, @tag_id=393, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903f028 @amount=12.79, @id=828, @merchant_id=322, @tag_id=399, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903e178 @amount=15.45, @id=829, @merchant_id=325, @tag_id=395, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd903e3d0 @amount=15.45, @id=824, @merchant_id=325, @tag_id=395, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd903e2b8 @amount=15.45, @id=819, @merchant_id=325, @tag_id=397, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd903e650 @amount=34.5, @id=820, @merchant_id=324, @tag_id=398, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903e768 @amount=34.5, @id=815, @merchant_id=324, @tag_id=396, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903e538 @amount=34.5, @id=825, @merchant_id=324, @tag_id=394, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903ebc8 @amount=60.2, @id=817, @merchant_id=323, @tag_id=395, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd903eee8 @amount=60.2, @id=822, @merchant_id=323, @tag_id=395, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd903ed58 @amount=60.2, @id=827, @merchant_id=323, @tag_id=399, @trans_date="2018-03-19">]
[5] pry(main)> Transaction.order_by_amount(transactions, 'desc')
=> [#<Transaction:0x007fadd903ed58 @amount=60.2, @id=827, @merchant_id=323, @tag_id=399, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd903eee8 @amount=60.2, @id=822, @merchant_id=323, @tag_id=395, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd903ebc8 @amount=60.2, @id=817, @merchant_id=323, @tag_id=395, @trans_date="2018-03-19">,
 #<Transaction:0x007fadd903e538 @amount=34.5, @id=825, @merchant_id=324, @tag_id=394, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903e768 @amount=34.5, @id=815, @merchant_id=324, @tag_id=396, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903e650 @amount=34.5, @id=820, @merchant_id=324, @tag_id=398, @trans_date="2018-01-30">,
 #<Transaction:0x007fadd903e2b8 @amount=15.45, @id=819, @merchant_id=325, @tag_id=397, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd903e3d0 @amount=15.45, @id=824, @merchant_id=325, @tag_id=395, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd903e178 @amount=15.45, @id=829, @merchant_id=325, @tag_id=395, @trans_date="2018-01-15">,
 #<Transaction:0x007fadd903f028 @amount=12.79, @id=828, @merchant_id=322, @tag_id=399, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903f168 @amount=12.79, @id=818, @merchant_id=322, @tag_id=393, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903f2d0 @amount=12.79, @id=823, @merchant_id=322, @tag_id=398, @trans_date="2018-11-01">,
 #<Transaction:0x007fadd903eab0 @amount=7.95, @id=816, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">,
 #<Transaction:0x007fadd903e998 @amount=7.95, @id=821, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">,
 #<Transaction:0x007fadd903e880 @amount=7.95, @id=826, @merchant_id=325, @tag_id=393, @trans_date="2018-02-09">]
```

**IT5 - An example of an array, a function that uses an array, and the result.**

Below is an example of an array which has been assigned the variable name ‘vegetables’.

.reverse is a function applied to arrays to reverse the order of the elements within the array.

```
vegetables = ["carrot", "celery", "pepper", "onion"]

p vegetables

p vegetables.reverse
```

Below is the result outputted to the terminal both of printing the array vegetables and underneath the result of reversing the order of the items within the array.

```
➜  PDA Work ruby IT\ examples
["carrot", "celery", "pepper", "onion"]
["onion", "pepper", "celery", "carrot"]
➜  PDA Work
```

**IT6 - An example of a hash, a function that uses a hash, and the result.**

Below is an example of a hash of staff names along with their staff numbers.

```
staff_nos = {
  David: 12345,
  Hayley: 12346,
  Simone: 12346,
  Colette: 12347,
  Alistair: 12348
}

p staff_nos
p staff_nos.length
```

Length is a function that can be applied to a hash, which returns the number of items in the hash. In this case, and as can be seen below, there are 5 items in the staff_nos hash and hence the length is returned as 5.

```
[➜  PDA Work ruby IT_examples.rb
{:David=>12345, :Hayley=>12346, :Simone=>12346, :Colette=>12347, :Alistair=>12348}
5
```

**IT7 - Example of Polymorphism in a program.**

Apple.java | Grocery.java | Toothpaste.java | Customer.java | Cereal.java

```
1   package Groceries;
2
3   public class Toothpaste extends Grocery{
4       private String name;
5       private double price;
6       private String barcode;
7       private String whiteningLevel;
8
9
10      public Toothpaste(String name, double price, String barcode, String whiteningLevel) {
11          super(name, price, barcode);
12          this.whiteningLevel = whiteningLevel;
13      }
14
15
16  }
```

Apple.java | Grocery.java | Toothpaste.java | Customer.java | Cereal.java

```java
package Groceries;

public class Cereal extends Grocery{

    private String name;
    private double price;
    private String barcode;
    private String boxSize;


    public Cereal(String name, double price, String barcode, String boxSize) {
        super(name, price, barcode);
        this.boxSize = boxSize;
    }


}
```

pple.java | Grocery.java | Toothpaste.java | Customer.java | Cereal.java

```java
package Groceries;

public class Apple extends Grocery {

    private String name;
    private double price;
    private String barcode;
    private double weight;


    public Apple(String name, double price, String barcode, double weight) {
        super(name, price, barcode);
        this.weight = weight;
    }

}
```

ple.java | Grocery.java | Toothpaste.java | Customer.java | Ce

```java
package Groceries;

public abstract class Grocery {

    private String name;
    private double price;
    private String barcode;

    public Grocery(String name, double price, String barcode) {
        this.name = name;
        this.price = price;
        this.barcode = barcode;
    }

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }

    public String getBarcode() {
        return barcode;
    }


}
```

In this example we have an abstract superclass Grocery and three types of grocery, apple, cereal and toothpaste, all of which have their own class extending from Grocery. As such they share the common properties of a Grocery: name, price and barcode.

The customer has a method to purchase a Grocery, which will reduce their wallet value by the price of the grocery. Because Apple, Toothpaste and Cereal are all extensions of Grocery we can plug in any object instance of either class. We do not need to write separate methods to purchase apples, toothpaste and cereal. This is an example of polymorphism.