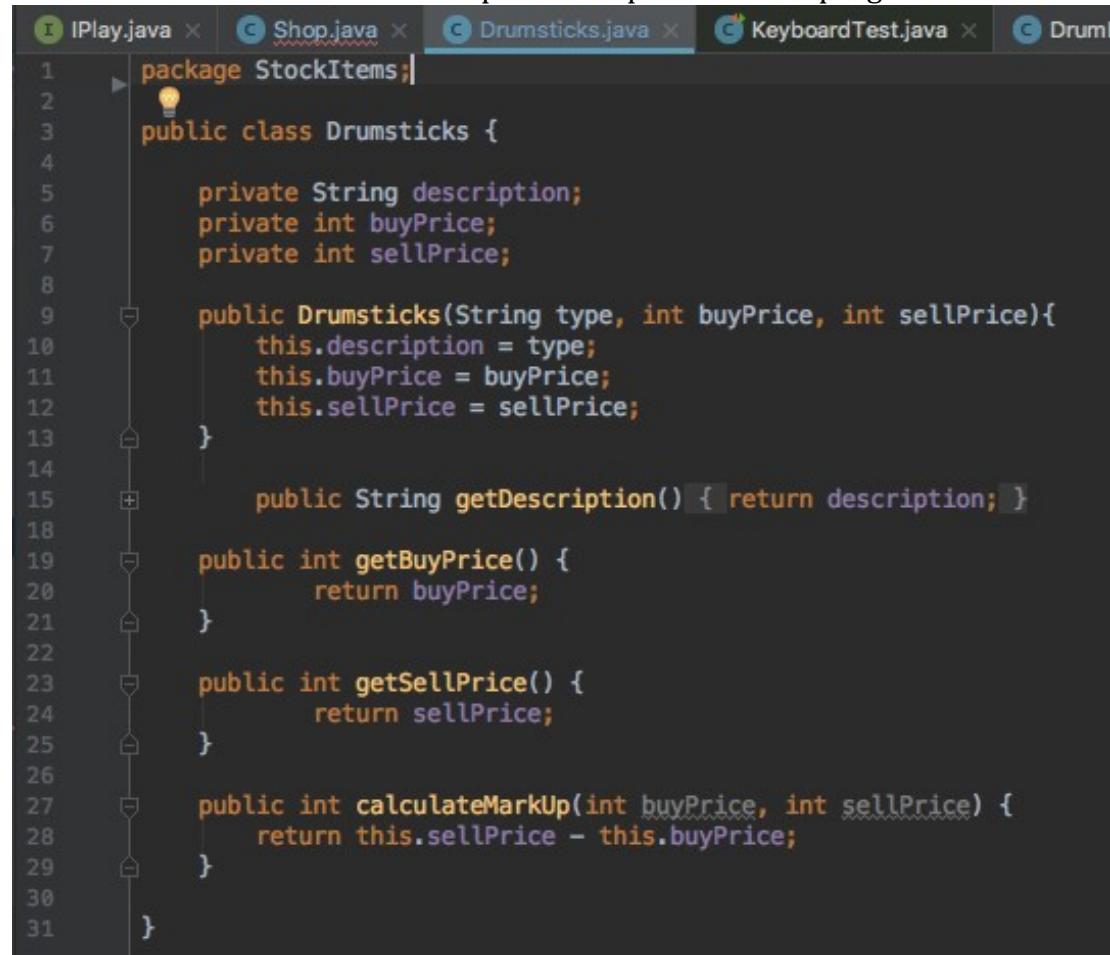


Evidence for Implementation and Testing Unit

Oliver Berry
Edinburgh 19

I.T 1 Take a screenshot of an example of encapsulation in a program.



```
1 package StockItems;
2
3 public class Drumsticks {
4
5     private String description;
6     private int buyPrice;
7     private int sellPrice;
8
9     public Drumsticks(String type, int buyPrice, int sellPrice){
10         this.description = type;
11         this.buyPrice = buyPrice;
12         this.sellPrice = sellPrice;
13     }
14
15     public String getDescription() { return description; }
16
17     public int getBuyPrice() {
18         return buyPrice;
19     }
20
21     public int getSellPrice() {
22         return sellPrice;
23     }
24
25     public int calculateMarkUp(int buyPrice, int sellPrice) {
26         return this.sellPrice - this.buyPrice;
27     }
28 }
29
30
31 }
```

I.T 2 Take a screenshot of the use of Inheritance in a program.

Take screenshots of:

- A Class

A screenshot of an IDE with three tabs: Instrument.java, DrumKit.java, and DrumKitTest.java. The Instrument.java tab is active, showing the following code:

```
1 public abstract class Instrument {  
2  
3     private String type;  
4     private String make;  
5     private String model;  
6     private int price;  
7  
8     public Instrument(String type, String make, String model, int price){  
9         this.type = type;  
10        this.make = make;  
11        this.model = model;  
12        this.price = price;  
13    }  
14    public String getType(){  
15        return type;  
16    }  
17  
18    public String setType(String type){  
19        return type;  
20    }  
21 }
```

- A Class that inherits from the previous class

A screenshot of an IDE with two tabs: Instrument.java and DrumKit.java. The DrumKit.java tab is active, showing the following code:

```
1 import Interfaces.IPlay;  
2  
3 public class DrumKit extends Instrument implements IPlay {  
4  
5     private String make;  
6     private String model;  
7     private int price;  
8  
9     public DrumKit(String type, String make, String model, int price){  
10        super(type, make, model, price);  
11    }  
12  
13  
14    public String play(String sound){  
15        return "This instrument goes " + sound;  
16    }  
17  
18 }
```

- An Object in the inherited class

```

Instrument.java x DrumKit.java x DrumKitTest.java x
1  import org.junit.Before;
2  import org.junit.Test;
3
4  import static org.junit.Assert.assertEquals;
5
6  public class DrumKitTest {
7
8      DrumKit drumkit;
9
10     @Before
11     public void before(){
12         drumkit = new DrumKit( type: "Percussion", make: "Ludwig", model: "Vistalite 5pc", price: 3400);
13     }
14
15     @Test
16     public void getType(){
17         assertEquals( expected: "Percussion", drumkit.getType());
18     }
19
20     @Test
21     public void setType(){
22         drumkit.setType("String");
23         assertEquals( expected: "String", drumkit.getType());
24     }
25 }

```

- A Method that uses the information inherited from another class.

```

Instrument.java x DrumKit.java x DrumKitTest.java x
1  import org.junit.Before;
2  import org.junit.Test;
3
4  import static org.junit.Assert.assertEquals;
5
6  public class DrumKitTest {
7
8      DrumKit drumkit;
9
10     @Before
11     public void before(){
12         drumkit = new DrumKit( type: "Percussion", make: "Ludwig", model: "Vistalite 5pc", price: 3400);
13     }
14
15     @Test
16     public void getType(){
17         assertEquals( expected: "Percussion", drumkit.getType());
18     }
19
20     @Test
21     public void setType(){
22         drumkit.setType("String");
23         assertEquals( expected: "String", drumkit.getType());
24     }
25 }

```

I.T 3 Demonstrate searching data in a program.

Take screenshots of:

- Function that searches data

```
83     def Album.find( id )
84         sql = "SELECT * FROM albums WHERE id = $1"
85         values = [id]
86         album = SqlRunner.run( sql, values )
87         result = Album.new( album.first )
88         return result
89     end
```

- The result of the function running

```
psql (10.1)
Type "help" for help.

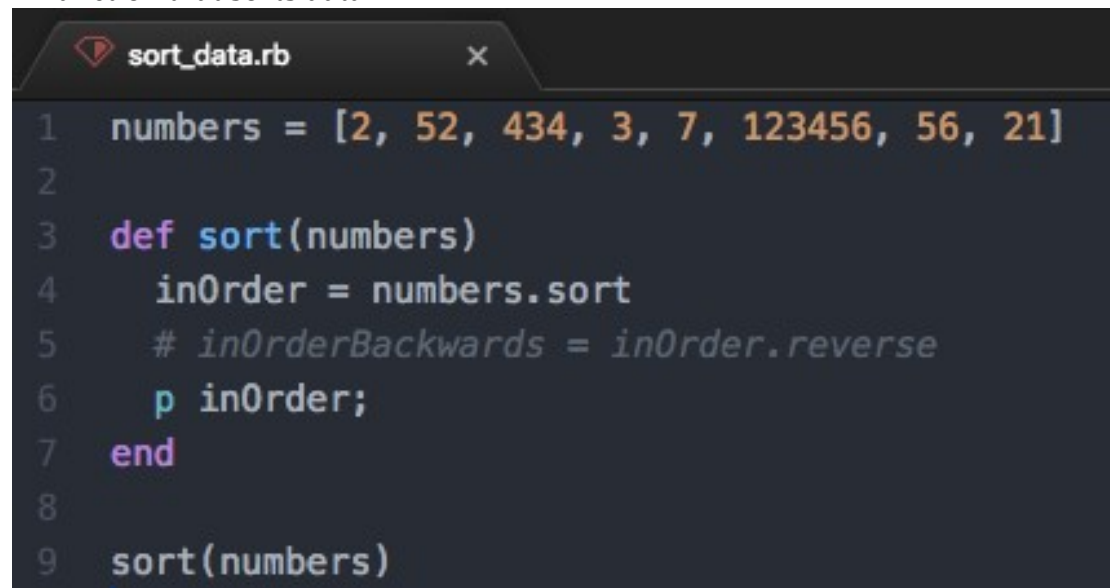
[records=# SELECT * FROM albums WHERE id = 4
[records=# ;
id | artist_id |      name      | buyprice | sellprice | quantity
-----+-----+-----+-----+-----+-----
 4 |          4 | Immortalized |         3 |          5 |         17
(1 row)

records=# █
```

I.T 4 Demonstrate sorting data in a program.

Take screenshots of:

- Function that sorts data



```
1 numbers = [2, 52, 434, 3, 7, 123456, 56, 21]
2
3 def sort(numbers)
4   inOrder = numbers.sort
5   # inOrderBackwards = inOrder.reverse
6   p inOrder;
7 end
8
9 sort(numbers)
```

- The result of the function running

```
[→ PDA_Work git:(master) ✕ ruby sort_data.rb
[2, 3, 7, 21, 52, 56, 434, 123456]
```

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

Take screenshots of:

- An array in a program

```
1  class WeddingPlan
2
3  attr_reader :table_name
4
5  def initialize (table_name)
6    @table_name = table_name
7    @seating_list = []
8  end
9
10 def add_to_table(person)
11   @seating_list.push(person)
12 end
13
14 end
15
```

- A function that uses the array

```
wedding_plan.rb  x  wedding_plan_spec.rb  x  person.rb  x
1  class WeddingPlan
2
3  attr_reader :table_name
4
5  def initialize (table_name)
6    @table_name = table_name
7    @seating_list = []
8  end
9
10 def add_to_table(person)
11   @seating_list.push(person)
12 end
13
14 end
15

1  require ("minitest/autorun")
2  require_relative("../wedding_plan")
3  require_relative("../person")
4
5  class WeddingPlanTest < MiniTest::Test
6    def setup
7      @weddingplan = WeddingPlan.new("CravenCottage")
8    end
9
10   def test_add_to_queue
11     tom = Person.new("Tom")
12     new_seating_list = @weddingplan.add_to_table(tom)
13     assert_equal([tom], new_seating_list)
14   end
15
16 end
17
```

- The result of the function running

```
[→ BusStopExample git:(master) ✖ ruby specs/wedding_plan_spec.rb
Run options: --seed 38085
```

```
# Running:
```

```
.
```

```
Finished in 0.000976s, 1024.5902 runs/s, 1024.5902 assertions/s.
```

```
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

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

Take screenshots of:

- A hash in a program

```
52  xmen = {  
53  
54    wolverine: {  
55      name: "Logan",  
56      moves: {  
57        slash: 50,  
58        skewer: 250  
59      }  
60    },  
61  
62    gambit: {  
63      name: "Remy LeBeau",  
64      moves: {  
65        royalflush: 350,  
66        cajunexplosion: 200  
67      }  
68    }  
69  }  
70  
71  p xmen [:wolverine][:gambit]
```

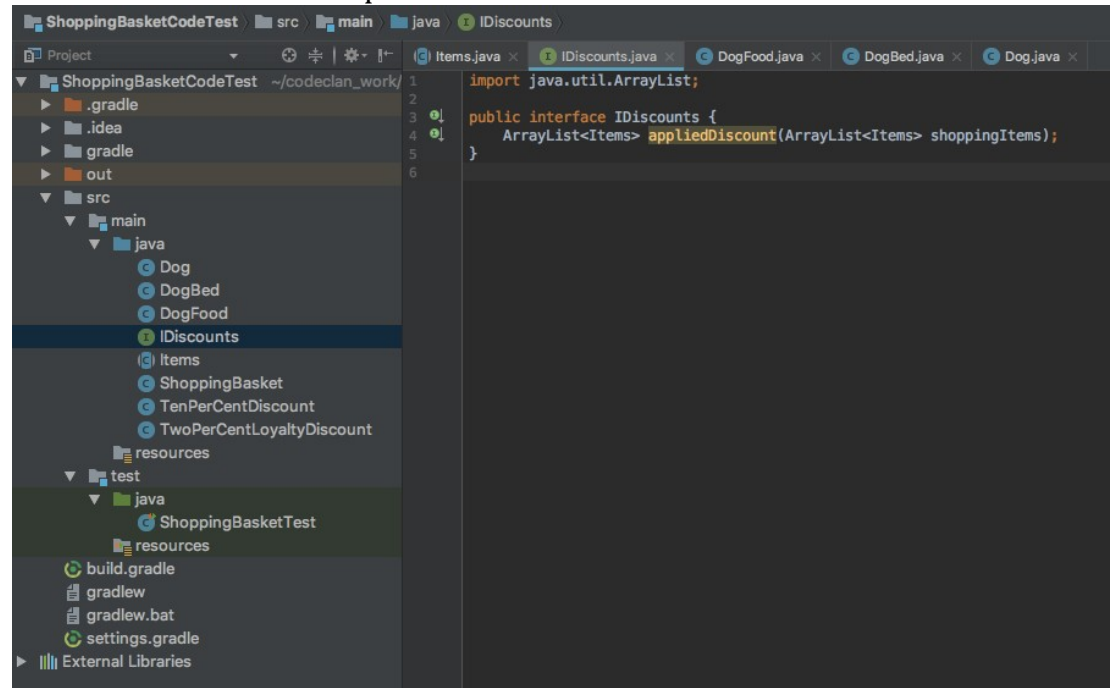
- A function that uses the hash and the function running

```
54  
55 xmen = {  
56  
57   wolverine: {  
58     name: "Logan",  
59     moves: {  
60       slash: 50,  
61       skewer: 250  
62     }  
63   },  
64  
65   gambit: {  
66     name: "Remy LeBeau",  
67     moves: {  
68       royalflush: 350,  
69       cajunexplosion: 200  
70     }  
71   }  
72 }  
73  
74 p xmen[:wolverine]  
75
```

[→ hashes git:(master) ✖ ruby hashes.rb
{:name=>"Logan", :moves=>{:slash=>50, :skewer=>250}}
→ hashes git:(master) ✖

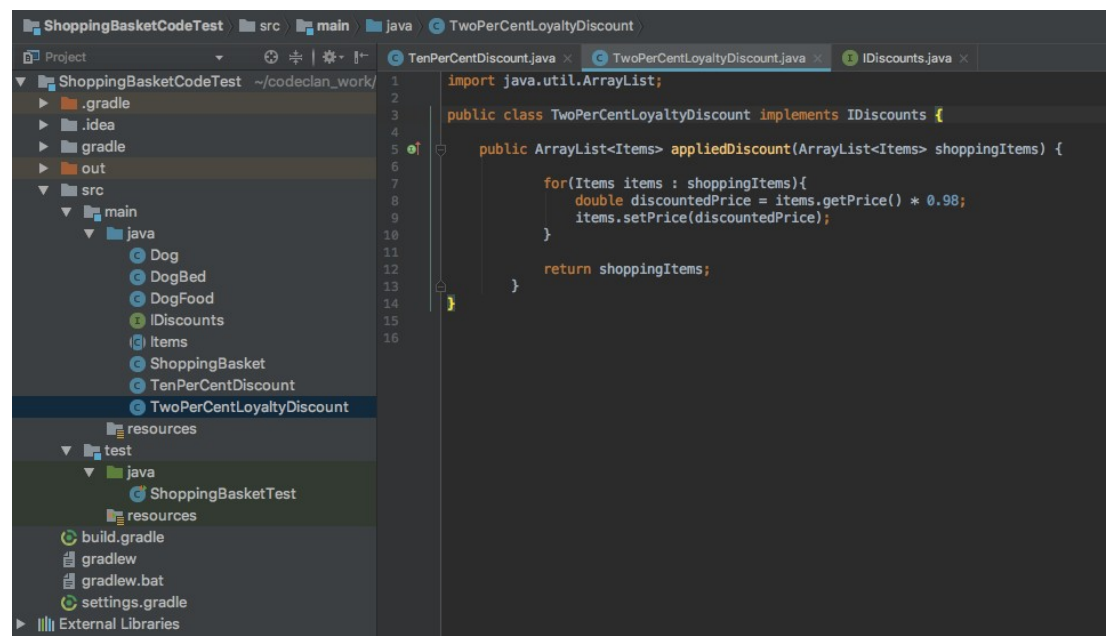
I.T 7 Demonstrate the use of Polymorphism in a program.

An interface used in multiple instances



This screenshot shows an IDE with the 'ShoppingBasketCodeTest' project. The left sidebar displays the project structure, with the 'src/main/java' directory expanded. The 'IDiscounts' interface is selected in the package list. The main editor window shows the code for 'IDiscounts.java'.

```
1 import java.util.ArrayList;
2
3 public interface IDiscounts {
4     ArrayList<Items> appliedDiscount(ArrayList<Items> shoppingItems);
5 }
6
```



This screenshot shows the same IDE with the 'TwoPerCentLoyaltyDiscount' class selected in the package list. The main editor window shows the code for 'TwoPerCentLoyaltyDiscount.java', which implements the 'IDiscounts' interface.

```
1 import java.util.ArrayList;
2
3 public class TwoPerCentLoyaltyDiscount implements IDiscounts {
4
5     public ArrayList<Items> appliedDiscount(ArrayList<Items> shoppingItems) {
6
7         for(Items items : shoppingItems){
8             double discountedPrice = items.getPrice() * 0.98;
9             items.setPrice(discountedPrice);
10        }
11
12        return shoppingItems;
13    }
14 }
15
16
```

The screenshot shows an IDE with two open Java files. The top editor displays `TenPerCentDiscount.java`, which implements the `IDiscounts` interface. The bottom editor displays `ShoppingBasket.java`, which is a simple class for managing a shopping basket.

```
import java.util.ArrayList;

public class TenPerCentDiscount implements IDiscounts{

    public ArrayList<Items> appliedDiscount(ArrayList<Items> shoppingItems) {

        double receiptTotal = 0;
        for(Items items : shoppingItems){
            receiptTotal += items.getPrice();
        }

        if (receiptTotal > 20){
            for(Items items : shoppingItems){
                double discountedPrice = items.getPrice() * 0.9;
                items.setPrice(discountedPrice);
            }
        }
        return shoppingItems;
    }
}
```

```
import java.util.ArrayList;

public class ShoppingBasket{

    private ArrayList<Items> shoppingItems;

    public ShoppingBasket(){
        this.shoppingItems = new ArrayList<>();
    }

    public ArrayList<Items> getShoppingItems(){
        return shoppingItems;
    }

    public void addItem(Items item){
        this.shoppingItems.add(item);
    }

    public void removeItem(Items item){
        this.shoppingItems.remove(item);
    }
}
```

ShoppingBasketTest.java ×

```
1  import org.junit.Before;
2  import org.junit.Test;
3
4  import static org.junit.Assert.assertEquals;
5
6  public class ShoppingBasketTest {
7
8      ShoppingBasket shoppingBasket;
9      Dog dog;
10     DogBed dogBed;
11     DogFood dogFood;
12     TenPerCentDiscount tenPerCentDiscount;
13     TwoPerCentLoyaltyDiscount twoPerCentLoyaltyDiscount;
14
15     @Before
16     public void before(){
17         shoppingBasket = new ShoppingBasket();
18         dog = new Dog("Boxer", 400.00);
19         dogBed = new DogBed("Comfort Plus", 45.00);
20         dogFood = new DogFood("Royal Canin Boxer", 50.00);
21         tenPerCentDiscount = new TenPerCentDiscount();
22         twoPerCentLoyaltyDiscount = new TwoPerCentLoyaltyDiscount();
23     }
24
25
26     @Test
27     public void canGetItemName(){
28         assertEquals("Boxer", dog.getName());
29     }
30
31     @Test
32     public void canSetName(){
33         dog.setName("Jack Russell");
34         assertEquals("Jack Russell", dog.getName());
35     }
36
```

```
41
42     @Test
43     public void canSetPrice(){
44         dogBed.setPrice(70.00);
45         assertEquals(70.00, dogBed.getPrice(), .05);
46     }
47
48     @Test
49     public void canAddItem(){
50         shoppingBasket.addItem(dogBed);
51         assertEquals(1, shoppingBasket.itemsInBasket());
52     }
53
54     @Test
55     public void canRemoveItem(){
56         shoppingBasket.addItem(dogBed);
57         shoppingBasket.addItem(dogFood);
58         assertEquals(2, shoppingBasket.itemsInBasket());
59         shoppingBasket.removeItem(dogFood);
60         assertEquals(1, shoppingBasket.itemsInBasket());
61     }
62
63     @Test
64     public void canEmptyBasket(){
65         shoppingBasket.addItem(dog);
66         shoppingBasket.addItem(dogFood);
67         shoppingBasket.emptyBasket();
68         assertEquals(0, shoppingBasket.itemsInBasket());
69     }
70
```