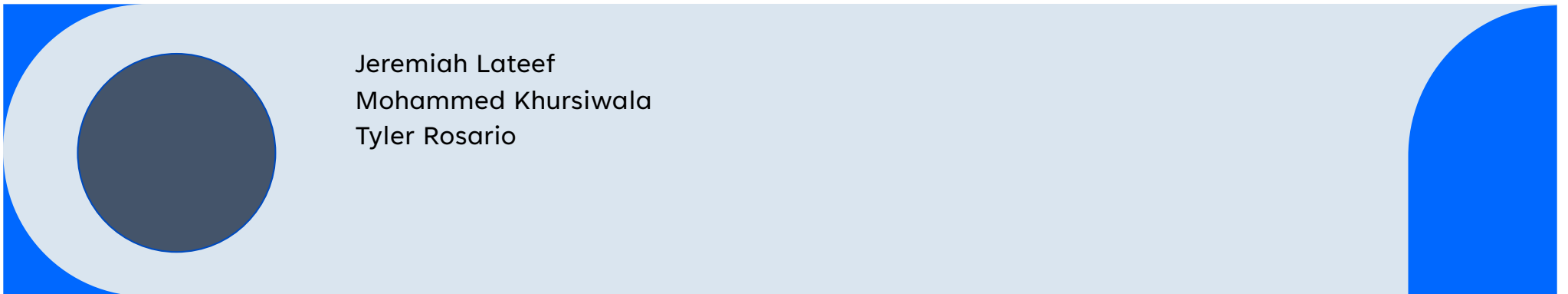


Capstone Project



Jeremiah Lateef
Mohammed Khursiwala
Tyler Rosario



Agenda

Initial Impressions

Design Approach

Challenges

Overview UML

Shopper & User Interface

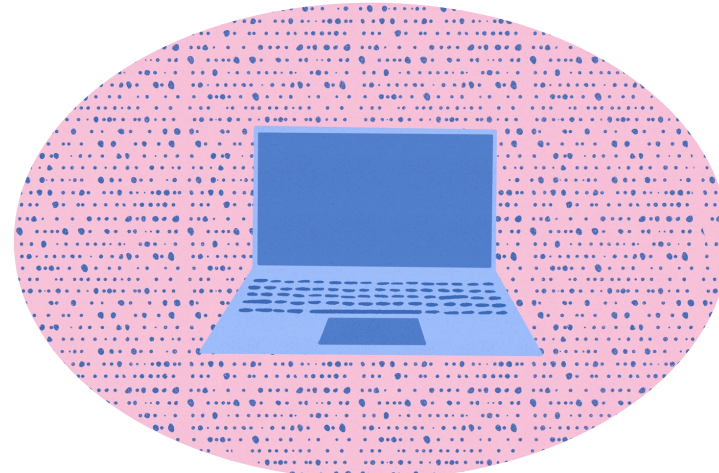
Children Extension

Software Engineering Metrics

Code Review

Takeaways

App Demo





Initial Impressions

- Duplicate code
- Different naming conventions
- Different designs
- Methods expected different parameters
- Lots of errors
- We each wrote code for ourselves without knowing other people would use it
- Nothing worked!

Jeremiah – Grocery Code

Mohammed – Person Code

Tyler – Course code



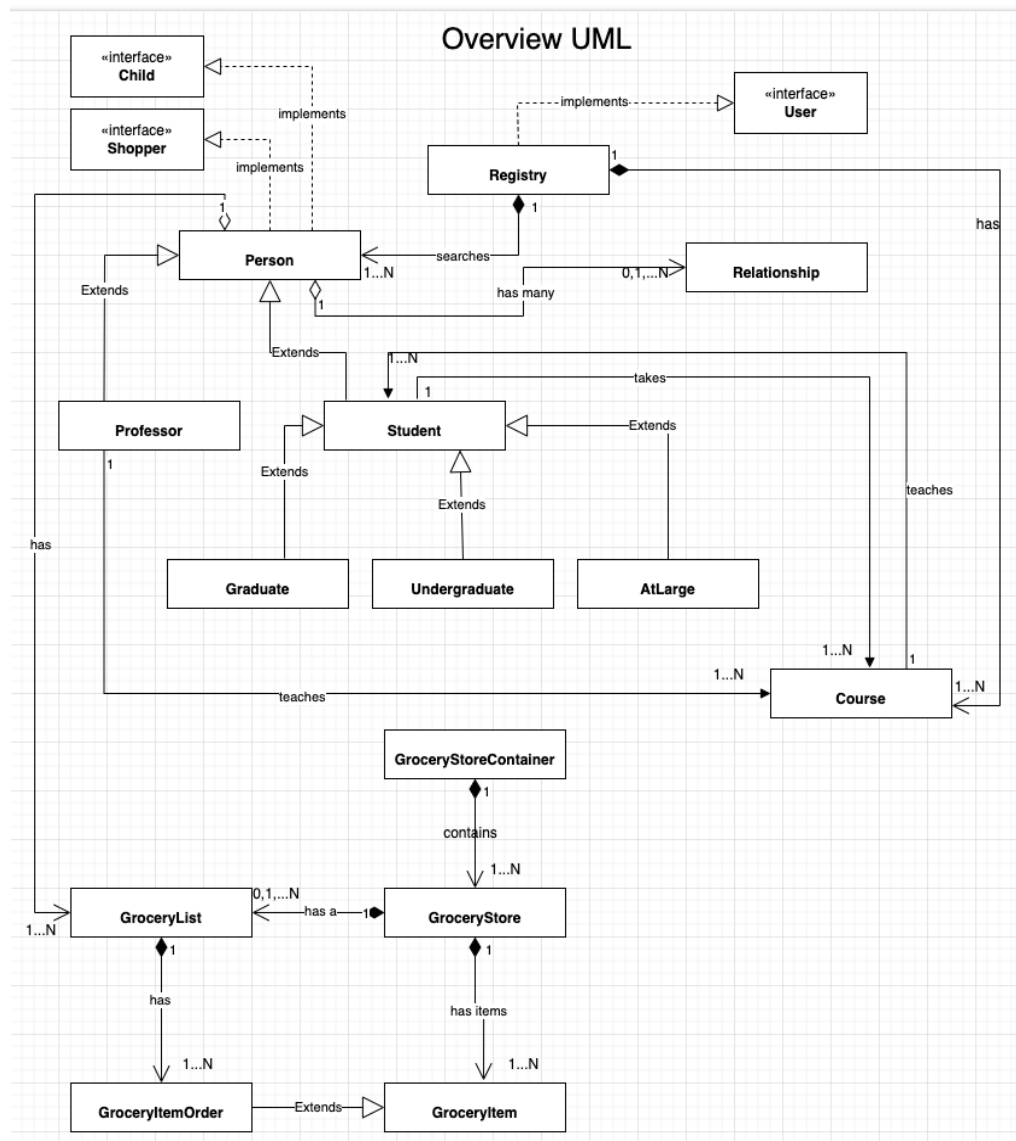
Design Approach

1. Meet regularly
2. Divide & Conquer
3. Use shared GitHub repo
4. Encapsulation, Inheritance, & Polymorphism
5. Code review
6. Documentation



Challenges

- Finding times to meet outside of class
- We had to modify our code to work with each other as opposed to working from the same design
- Our design decisions were made in isolation
- Lots of things had to be changed like return types, names of methods, duplicate code
- Coding as a team with a shared repo
- Code showed remnants of working individually like of comments or clear names for things



Shopper & User Interface

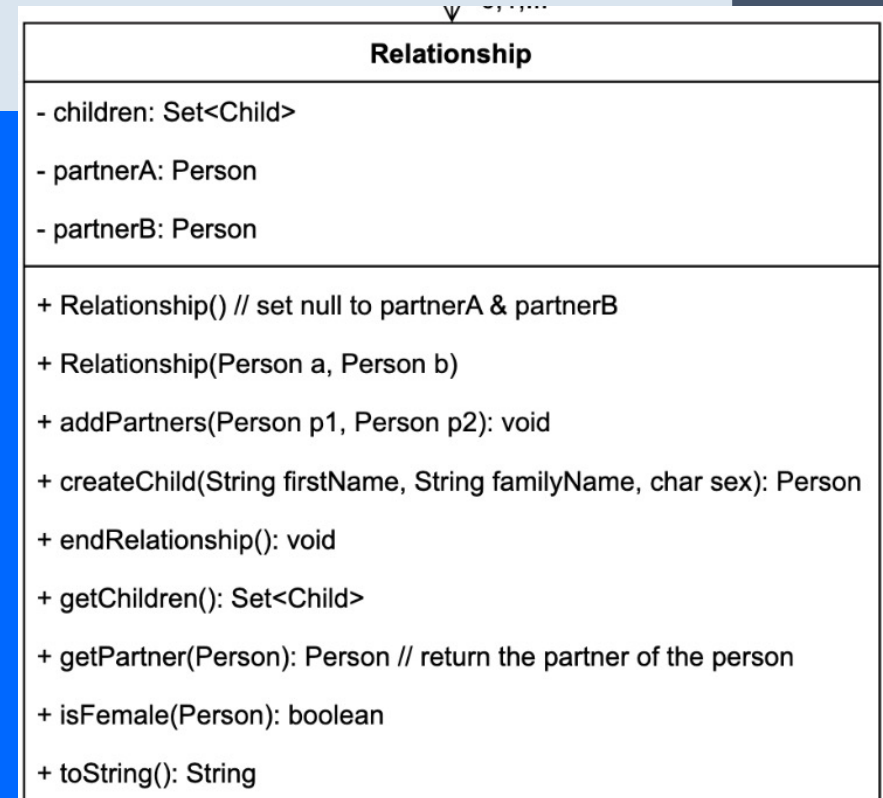
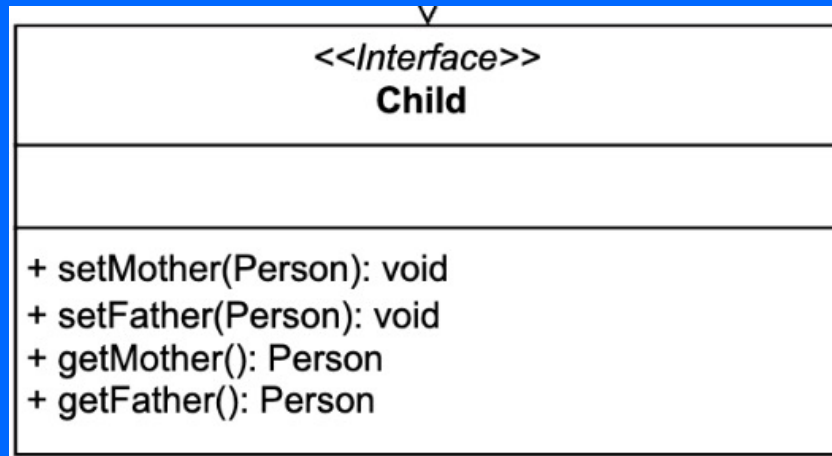
```
public interface Shopper {  
    no usages 1 implementation ① mkhursiwala2005  
    public double adjustGroceryBudget(double amount);  
    3 usages 3 implementations ① mkhursiwala2005  
    public double calculateDiscount(GroceryList list);  
    no usages 1 implementation ① trosario102  
    public void setGroceryBudget(double amount);  
}
```

```
public interface User {  
    1 usage 1 implementation ① mkhursiwala2005  
    public void addGroceryItems(String item, int quantity, GroceryStore store, GroceryList list);  
    1 usage 1 implementation ① trosario102  
    public double checkPersonDiscount(Person person, GroceryList list); // checks if person is pro  
    1 usage 1 implementation ① trosario102  
    public boolean selectGroceryStore(String storeName, GroceryStoreContainer storesLists);  
}
```

Key Part in Our Program


- Person
- Registry
- Course

Children Extension





Software Engineering Metrics

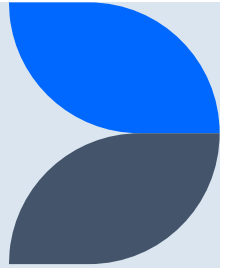
- Number of classes: 14
 - Number of interfaces: 3
 - Number of methods: 115
 - Lines of code: 1144
 - Number of unresolved bugs: 22
- 

Code Review Results (Jeremiah)

GroceryItem, GroceryStore, GroceryList

Name	Defects ¹		Preparation Data			Est.
	Major	Minor	Size	Time	Rate	Yield
Tyler Rosario		2	118	50	98.3	0%
Mohammed Khursiwala		3	118	70	137.6	0%
Totals:	0	5	236	120 mins = 2 hrs	118	

Code Review Results (Jeremiah)



- Some of the complex methods did not have any comment explaining what it does.
- There were redundant `java.util` package in some of the files.
- The code had some stylistic issues that had some minor impact on the beauty of the code, i.e. extra spaces between methods.
- A few of the comments were reductive, and did not explain the code well.

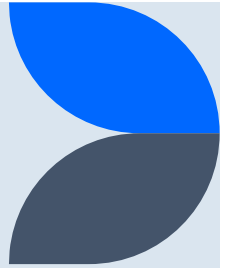
Code Review Results (Mohammed)

Person.java

Engineer Data

Name	Defects ¹		Preparation Data			Est.
	Major	Minor	Size	Time	Rate	Yield
Jeremiah Lateef		4	158	70		
Tyler Rosario		7	158	50		
Totals:		11	316	120 min = 2.00 hr		

Code Review Results (Mohammed)



- Comments are not written in full, i. e they're abbreviated
- Over importing with java.util.*
- Extra spaces before some of the methods
- Unused some of methods in Person class
- Not many comments to explain code

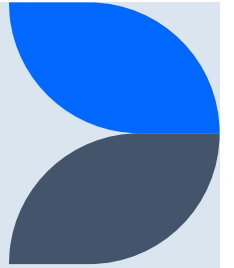
Code Review Results (Tyler)

Course, Relationship

Name	Defects		Preparation Data			Est.
	Major	Minor	Size (LOC)	Time (Min)	Rate (LOC/ <u>hr</u>)	Yield (major defects only)
Jeremiah Lateef (J)		4	202	120 min	101	0%
Mohammed Khursiwala (M)		7	202	90 min	134.6	0%
Totals:	0	11	404	210 min = <u>3.5 hrs</u>	117.8	


Code Review Results (Tyler)

- Some methods had no usages
- Some constructor parameters did not have descriptive names
- Not enough comments
- Unused imports





Code Review Takeaways

- See each other's coding style
 - Different ways to implement code
 - Learn exactly how different classes/methods work
 - Allows for bug detection leading to higher quality code
 - Standardize coding practices
 - Team collaboration and cohesion
- 

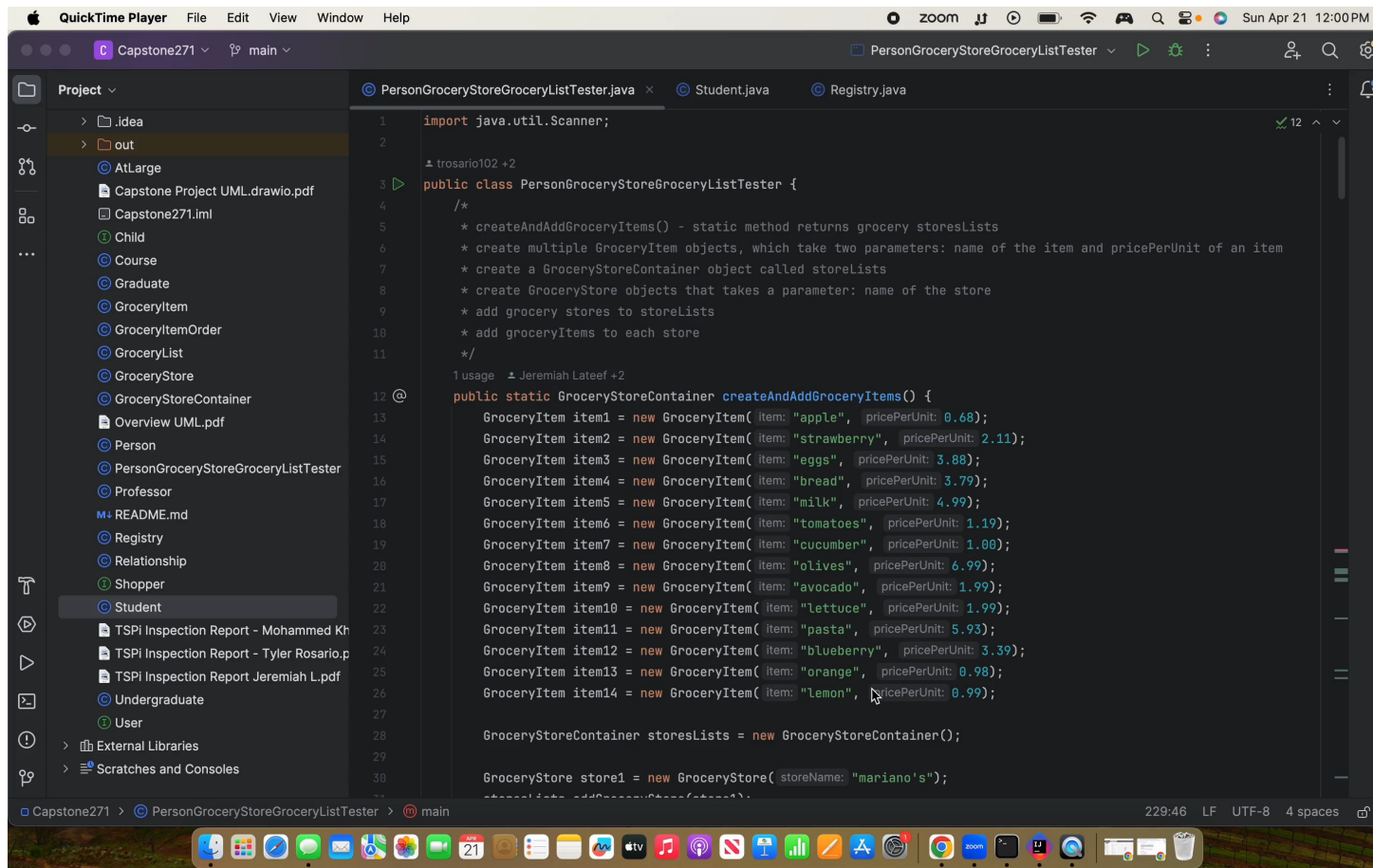
Takeaways

- Communicate early and often
- Establish naming conventions
- Design a unified UML to simplify coding process
- Use feature branches and review code before merging
- Test code individually and together early in the process

App Demo



Ancestry Demo 1

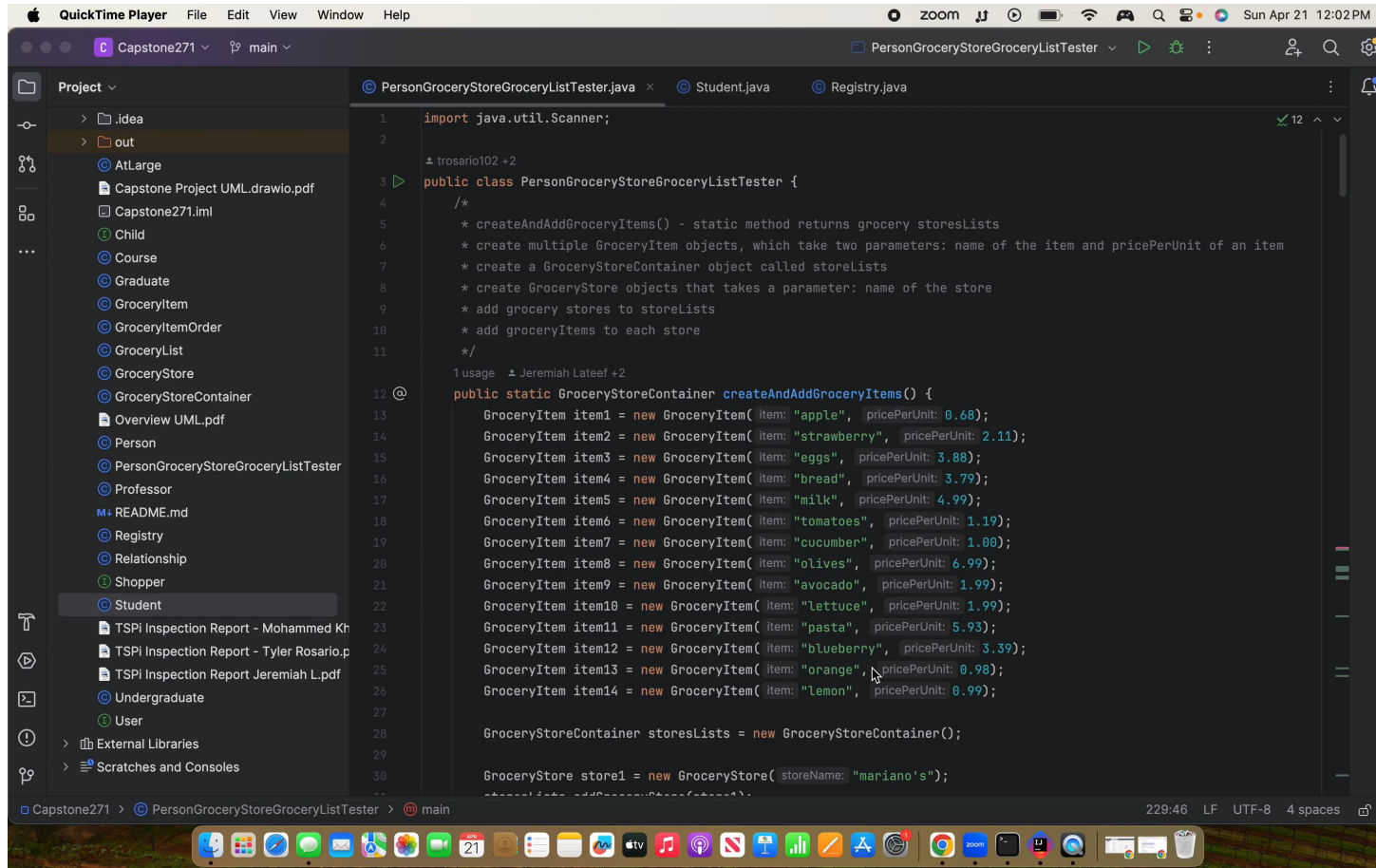


The screenshot shows an IDE window titled "QuickTime Player" with a menu bar (File, Edit, View, Window, Help) and a status bar at the bottom. The main editor displays the file "PersonGroceryStoreGroceryListTester.java". The code is as follows:

```
1 import java.util.Scanner;
2
3 // trossario102 +2
4 public class PersonGroceryStoreGroceryListTester {
5     /*
6      * createAndAddGroceryItems() - static method returns grocery storesLists
7      * create multiple GroceryItem objects, which take two parameters: name of the item and pricePerUnit of an item
8      * create a GroceryStoreContainer object called storeLists
9      * create GroceryStore objects that takes a parameter: name of the store
10     * add grocery stores to storeLists
11     * add groceryItems to each store
12     */
13     1 usage  trossario102 +2
14     public static GroceryStoreContainer createAndAddGroceryItems() {
15         GroceryItem item1 = new GroceryItem( item: "apple", pricePerUnit: 0.68);
16         GroceryItem item2 = new GroceryItem( item: "strawberry", pricePerUnit: 2.11);
17         GroceryItem item3 = new GroceryItem( item: "eggs", pricePerUnit: 3.88);
18         GroceryItem item4 = new GroceryItem( item: "bread", pricePerUnit: 3.79);
19         GroceryItem item5 = new GroceryItem( item: "milk", pricePerUnit: 4.99);
20         GroceryItem item6 = new GroceryItem( item: "tomatoes", pricePerUnit: 1.19);
21         GroceryItem item7 = new GroceryItem( item: "cucumber", pricePerUnit: 1.00);
22         GroceryItem item8 = new GroceryItem( item: "olives", pricePerUnit: 6.99);
23         GroceryItem item9 = new GroceryItem( item: "avocado", pricePerUnit: 1.99);
24         GroceryItem item10 = new GroceryItem( item: "lettuce", pricePerUnit: 1.99);
25         GroceryItem item11 = new GroceryItem( item: "pasta", pricePerUnit: 5.93);
26         GroceryItem item12 = new GroceryItem( item: "blueberry", pricePerUnit: 3.39);
27         GroceryItem item13 = new GroceryItem( item: "orange", pricePerUnit: 0.98);
28         GroceryItem item14 = new GroceryItem( item: "lemon", pricePerUnit: 0.99);
29
30         GroceryStoreContainer storeLists = new GroceryStoreContainer();
31
32         GroceryStore store1 = new GroceryStore( storeName: "mariano's");
33         storeLists.addGroceryStore(store1);
34     }
35 }
```

The IDE interface includes a "Project" sidebar on the left with a tree view of files and folders, and a "Scratches and Consoles" panel at the bottom. The status bar at the bottom right shows "229:46 LF UTF-8 4 spaces".

Ancestry Demo 2



The screenshot shows an IDE window titled "QuickTime Player" with a menu bar (File, Edit, View, Window, Help) and a status bar at the bottom. The main editor displays the file "PersonGroceryStoreGroceryListTester.java". The left sidebar shows a project structure with folders like ".idea" and "out", and a list of files including "Capstone Project UML.drawio.pdf", "Capstone271.iml", "Child", "Course", "Graduate", "GroceryItem", "GroceryItemOrder", "GroceryList", "GroceryStore", "GroceryStoreContainer", "Overview UML.pdf", "Person", "PersonGroceryStoreGroceryListTester", "Professor", "README.md", "Registry", "Relationship", "Shopper", "Student", "TSPI Inspection Report - Mohammed Kh", "TSPI Inspection Report - Tyler Rosario.p", "TSPI Inspection Report Jeremiah L.pdf", "Undergraduate", and "User". The main editor shows the following Java code:

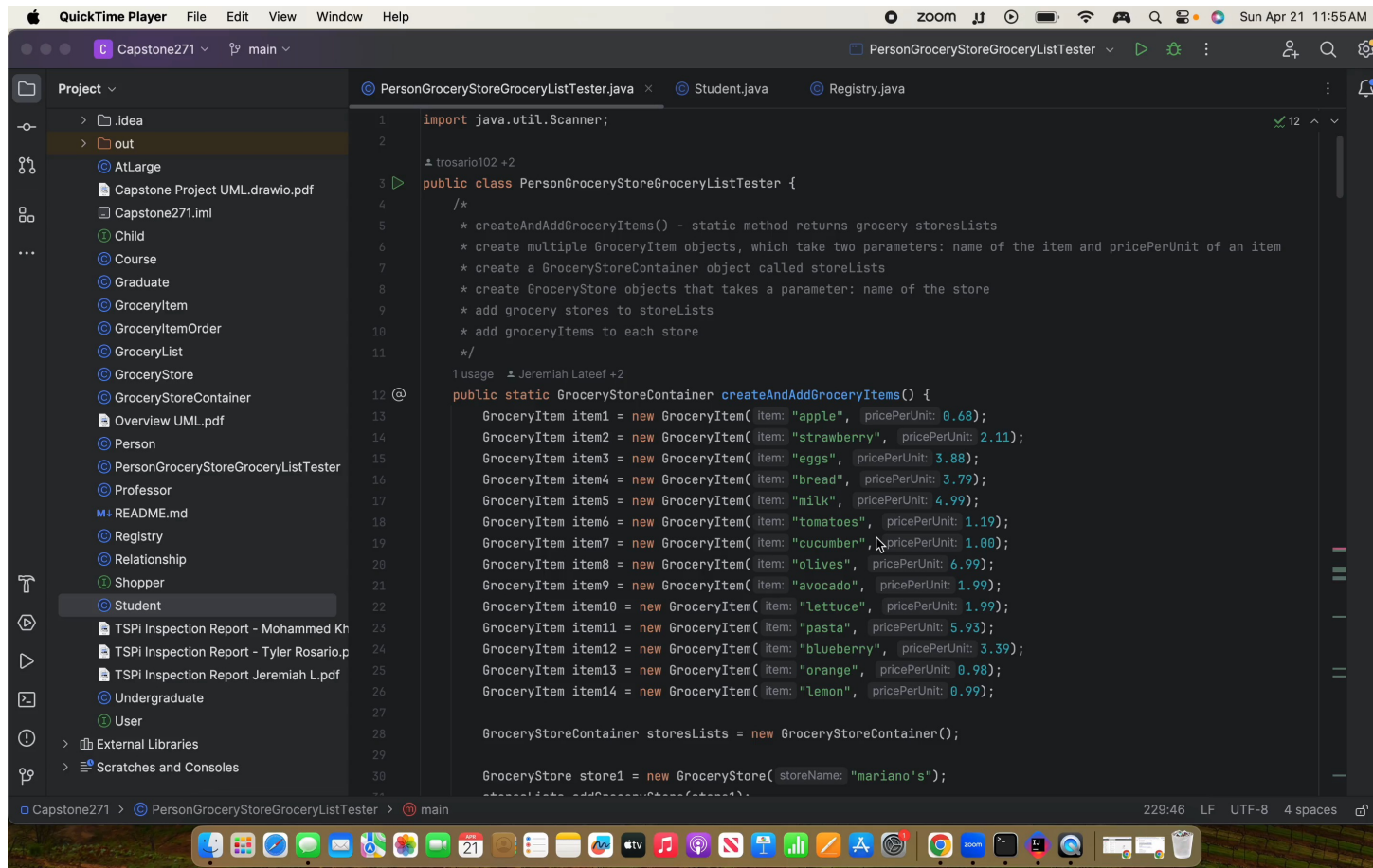
```
1 import java.util.Scanner;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

The code defines a class `PersonGroceryStoreGroceryListTester` with a static method `createAndAddGroceryItems()` that creates and adds grocery items to a store. The items are:

- apple (0.68)
- strawberry (2.11)
- eggs (3.88)
- bread (3.79)
- milk (4.99)
- tomatoes (1.19)
- cucumber (1.00)
- olives (6.99)
- avocado (1.99)
- lettuce (1.99)
- pasta (5.93)
- blueberry (3.39)
- orange (0.98)
- lemon (0.99)

The code also creates a `GroceryStoreContainer` object and a `GroceryStore` object named "mariano's".

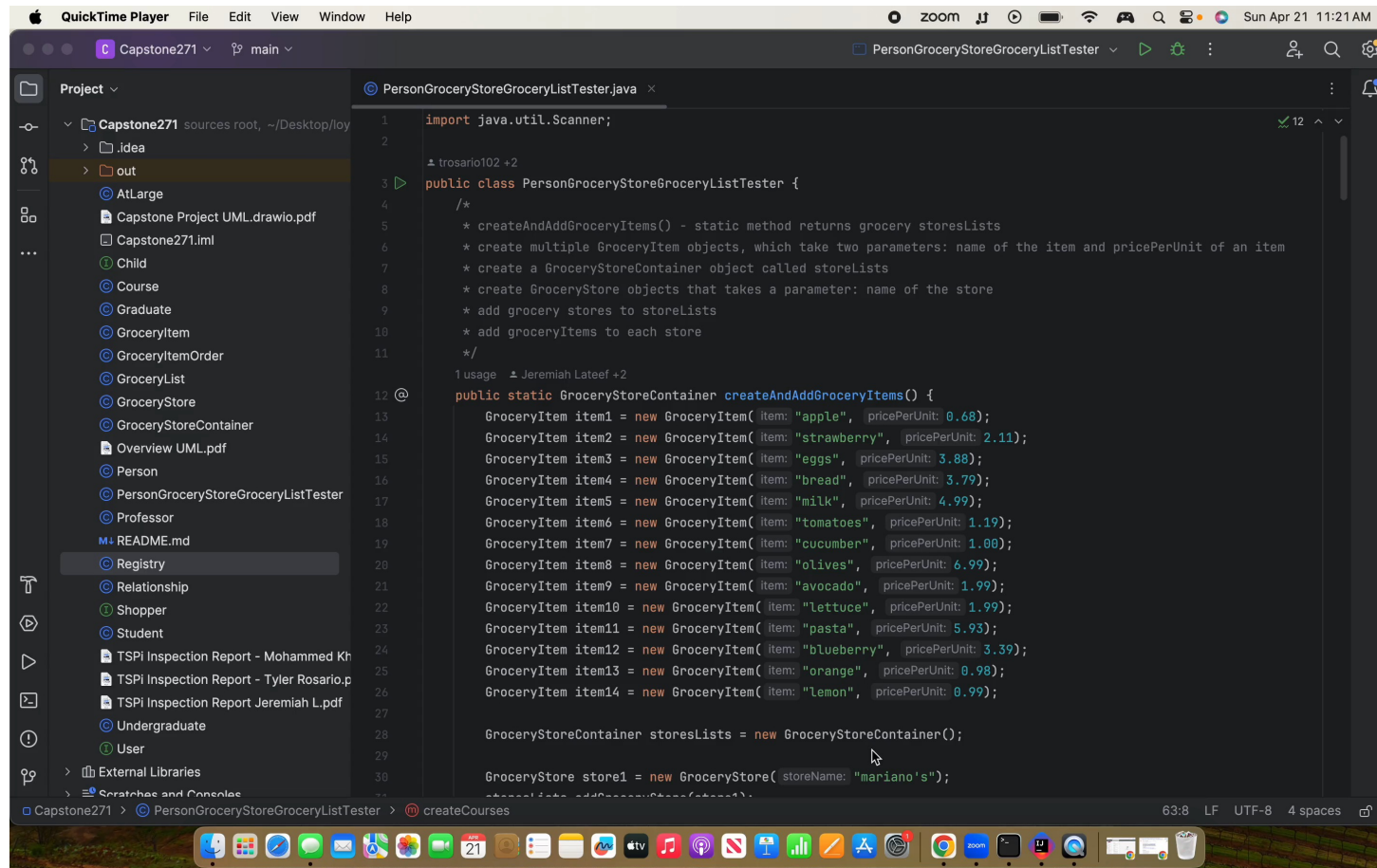
Add/Drop Course



The screenshot shows an IDE window titled 'QuickTime Player' with a menu bar (File, Edit, View, Window, Help). The main window displays a Java project named 'Capstone271' with a 'main' module. The project structure on the left includes a 'Project' view with a 'Course' entity and a 'PersonGroceryStoreGroceryListTester' class. The 'PersonGroceryStoreGroceryListTester.java' file is open, showing a Java class with a static method 'createAndAddGroceryItems()' that initializes a 'GroceryStoreContainer' and adds 14 'GroceryItem' objects. The code is as follows:

```
1 import java.util.Scanner;
2
3 import javax.swing.*;
4
5 public class PersonGroceryStoreGroceryListTester {
6     /*
7      * createAndAddGroceryItems() - static method returns grocery storesLists
8      * create multiple GroceryItem objects, which take two parameters: name of the item and pricePerUnit of an item
9      * create a GroceryStoreContainer object called storeLists
10     * create GroceryStore objects that takes a parameter: name of the store
11     * add grocery stores to storeLists
12     * add groceryItems to each store
13     */
14     @ static
15     public static GroceryStoreContainer createAndAddGroceryItems() {
16         GroceryItem item1 = new GroceryItem( item: "apple", pricePerUnit: 0.68);
17         GroceryItem item2 = new GroceryItem( item: "strawberry", pricePerUnit: 2.11);
18         GroceryItem item3 = new GroceryItem( item: "eggs", pricePerUnit: 3.88);
19         GroceryItem item4 = new GroceryItem( item: "bread", pricePerUnit: 3.79);
20         GroceryItem item5 = new GroceryItem( item: "milk", pricePerUnit: 4.99);
21         GroceryItem item6 = new GroceryItem( item: "tomatoes", pricePerUnit: 1.19);
22         GroceryItem item7 = new GroceryItem( item: "cucumber", pricePerUnit: 1.00);
23         GroceryItem item8 = new GroceryItem( item: "olives", pricePerUnit: 6.99);
24         GroceryItem item9 = new GroceryItem( item: "avocado", pricePerUnit: 1.99);
25         GroceryItem item10 = new GroceryItem( item: "lettuce", pricePerUnit: 1.99);
26         GroceryItem item11 = new GroceryItem( item: "pasta", pricePerUnit: 5.93);
27         GroceryItem item12 = new GroceryItem( item: "blueberry", pricePerUnit: 3.39);
28         GroceryItem item13 = new GroceryItem( item: "orange", pricePerUnit: 0.98);
29         GroceryItem item14 = new GroceryItem( item: "lemon", pricePerUnit: 0.99);
30
31         GroceryStoreContainer storeLists = new GroceryStoreContainer();
32
33         GroceryStore store1 = new GroceryStore( storeName: "mariano's");
34         store1.addGroceryItems(item1);
35         store1.addGroceryItems(item2);
36         store1.addGroceryItems(item3);
37         store1.addGroceryItems(item4);
38         store1.addGroceryItems(item5);
39         store1.addGroceryItems(item6);
40         store1.addGroceryItems(item7);
41         store1.addGroceryItems(item8);
42         store1.addGroceryItems(item9);
43         store1.addGroceryItems(item10);
44         store1.addGroceryItems(item11);
45         store1.addGroceryItems(item12);
46         store1.addGroceryItems(item13);
47         store1.addGroceryItems(item14);
48         storeLists.addStore(store1);
49     }
50 }
```

Grocery Demo





Thank you for listening!

Any questions?

