Homework 5 CS111

กำหนดส่ง 5 พฤษภาคม 2566

วัตถุประสงค์

- เพื่อฝึกใช้ ArrayList, interface และการสืบทอด classes
- เพื่อฝึกใช้เมท็อด sort จาก API
- เพื่อฝึกการ implementation (เช่นการกรอง items) ได้อย่างยืดหยุ่น
- เพื่อให้สามารถอ่านไฟล์และจัดการกับความผิดปกติได้

คำสั่งเพิ่มเติม

เนื่องจากแต่ละข้อมีการปรับแก้คลาสเดิมในบางส่วน จึงให้ทำแต่ละข้อแยก package กัน (ต้องสำเนาโค้ดที่ได้ไปใช้ในข้อต่อๆ ไป แล้วปรับแก้จากโค้ดที่สำเนามานั้น)

เกริ่นนำ

ในปัจจุบัน ตู้ขายสินค้าแบบฉลาด (smart vending machine) เช่น ตู้เต่าบิน (Tao-Bin)ได้รับความนิยมอย่างมาก บริษัทสตาร์ทอัพได้ขอให้พวกเรา implement prototype ของตู้ขายสินค้า

Courtesy: https://www.tao-bin.com/en/menu

1. หลังจากพิจารณา ทีมได้ออกแบบร่างแรก โดยโฟกัสที่ให้ตู้สามารถเพิ่มและแสดงสินค้าที่ขายในตู้ สินค้าที่จะนำมาทดลองจำหน่ายจะเป็นกลุ่มเครื่องดื่ม (Drink) ทีมได้ให้โค้ดของคลาส Drink และไดอะแกรมเพื่อให้พวกเราสร้างคลาสและอินเทอร์เฟสที่เหลือ โดยมีรายละเอียดเพิ่มเติมดังนี้

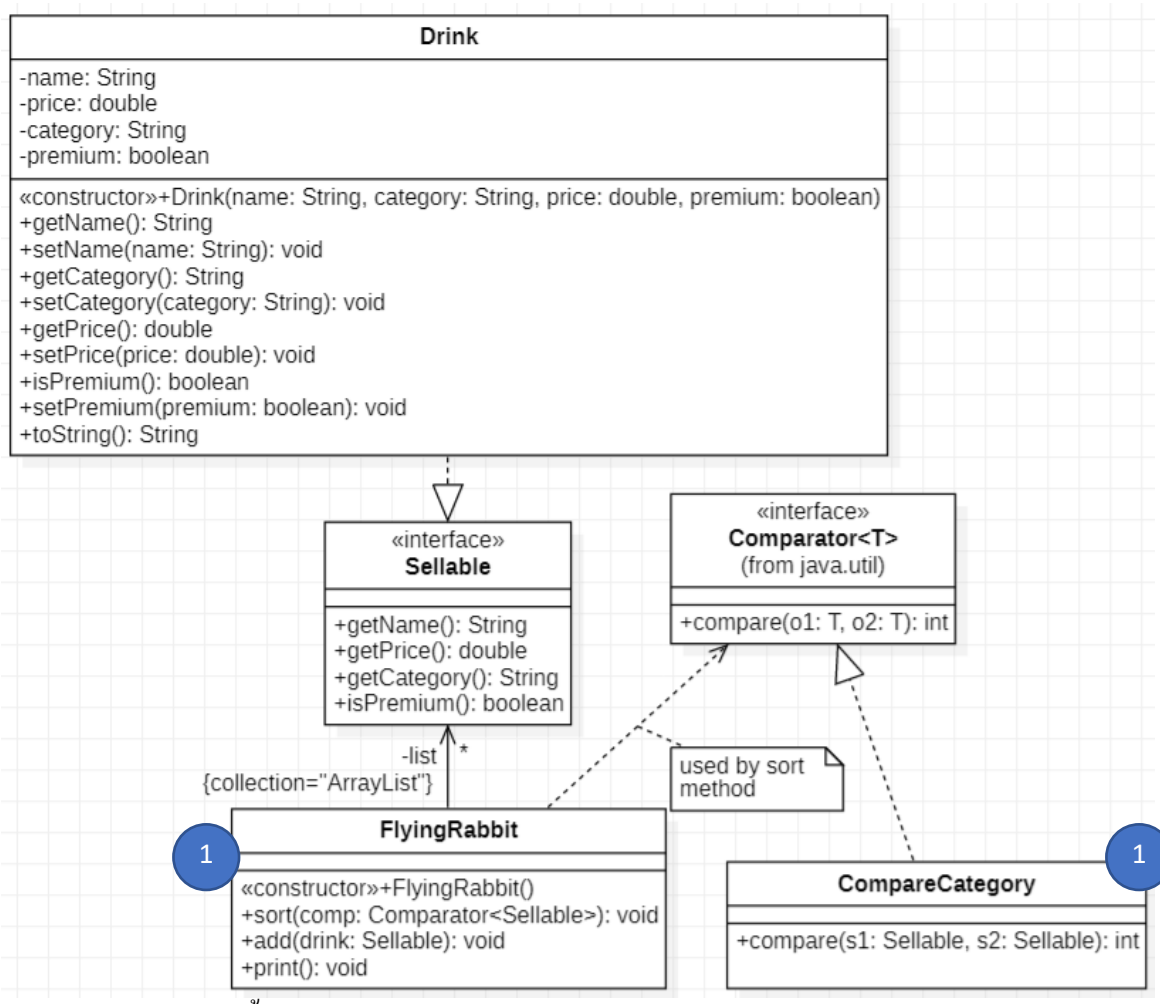
- อินเทอร์เฟส **`Sellable`** วางข้อกำหนดความสามารถที่จำเป็นของสินค้าที่จะนำมาขายในตู้ เพื่อให้ตู้รองรับการเพิ่มสินค้าที่จะนำมาขายใด ๆ ได้โดยอิสระ (เปลี่ยนตัวสินค้าที่ขายจริงได้ง่าย) อินเทอร์เฟสมีเมท็อดที่กำหนดไว้ได้แก่ `getName, getPrice, getCategory, isPremium` ซึ่งเมท็อดเหล่านี้คืนค่าชนิดต่าง ๆ ดังแสดงในไดอะแกรม (ให้โค้ด นศ. ไม่ต้อง implement เอง)
- คลาส **`FlyingRabbit`** แทนตู้ขายสินค้า มีเพียงตัวแปรวัตถุ list โดยใช้เป็นตัวแปรเก็บกลุ่มสินค้าที่จะนำมาขาย (Sellable) (เช่น Drink หรือสินค้าอื่น ๆ) โดยคลาสนี้มีเมท็อดดังนี้
    a) **Constructor**: ให้ค่าเริ่มต้นกับ list (สร้าง array list และให้ค่ากับมัน)
    b) **`add`** method เพิ่มสินค้าที่จะขายเข้าตู้
    c) **`sort`** method เรียงลำดับสินค้าในตู้ตามแต่วัตถุ comparator ที่ให้ผ่านมาทางพารามิเตอร์ของเมท็อด
    Hint: การเรียงลำดับสามารถใช้เมท็อดสถิต `java.util.Collections.sort` ที่เหมาะสม

d) **`print`** method พิมพ์สินค้าทั้งหมดในตู้ ตามตัวอย่างที่ให้

- คลาส **`CompareCategory`** เป็น subtype ของอินเทอร์เฟส **`Comparator<T>`** ที่อยู่ในแพกเกจ `java.util` โดยคลาสนี้จะเป็นตัวช่วยการทำการเรียงลำดับตาม `category` ซึ่งเราจะต้องเขียนทับเมท็อด **`compare`** เพื่อให้ทำหน้าที่เปรียบเทียบค่า category สำหรับ **`Sellable`** โดยวิธีการเปรียบเทียบของเมท็อด **`compare`** 2 sellable, s1 และ s2 ตาม category (เรียงลำดับตาม Alphabet) จะคืนค่าดังนี้
  - คืนค่าน้อยกว่า 0 ถ้า category ของวัตถุ s1 น้อยกว่า category ของวัตถุ s2
  - คืนค่า 0 ถ้าเท่ากัน
  - คืนค่ามากกว่า 0 ถ้า category ของวัตถุ s1 มากกว่า category ของวัตถุ s2

Hint: - การทำ realization กับอินเทอร์เฟส `Comparator<T>` ซึ่งเป็น generic ให้ระบุ `<Sellable>` แทน `<T>`

- เนื่องจาก category เป็นชนิด String เราอาจใช้เมท็อดการเปรียบเทียบของ String ได้



กำหนดคลาสทดสอบให้ดังนี้

```
package q1;

import java.util.Comparator;
public class Test1FlyingRabit {
```

```
        public static void main(String[] args) {
                FlyingRabbit machine = new FlyingRabbit();
                machine.add(new Drink("Double Espresso", "Hot Coffee", 45, false));
                machine.add(new Drink("Chocolate Protein Shake", "Protein Shakes", 60, true));
                machine.add(new Drink("Hot Cappucino", "Hot Coffee", 40, false));
                machine.add(new Drink("Hot Caramel Latte", "Hot Coffee", 45, true));
                machine.add(new Drink("Dirty", "Iced Coffee", 35, false));
                machine.add(new Drink("Iced Cappucino", "Iced Coffee", 45, false));
                machine.add(new Drink("Matcha Protein Shake", "Protein Shakes", 55, true));
                machine.add(new Drink("Hot Americano", "Hot Coffee", 35, false));
                machine.add(new Drink("Hot Cafe' Latte", "Hot Coffee", 40, false));
                machine.add(new Drink("Strawberry Protein Shake", "Protein Shakes", 60, true));
                machine.add(new Drink("Iced Mocha", "Iced Coffee", 40, false));
                machine.add(new Drink("Iced Matcha Cafe Latte", "Iced Coffee", 45, true));

                System.out.println("Before sorting: ");
                machine.print();

                // make sorting and then print
                Comparator cc = new CompareCategory();
                machine.sort(cc);

                System.out.println("After sorting by Category: ");
                machine.print();
        }

}
```

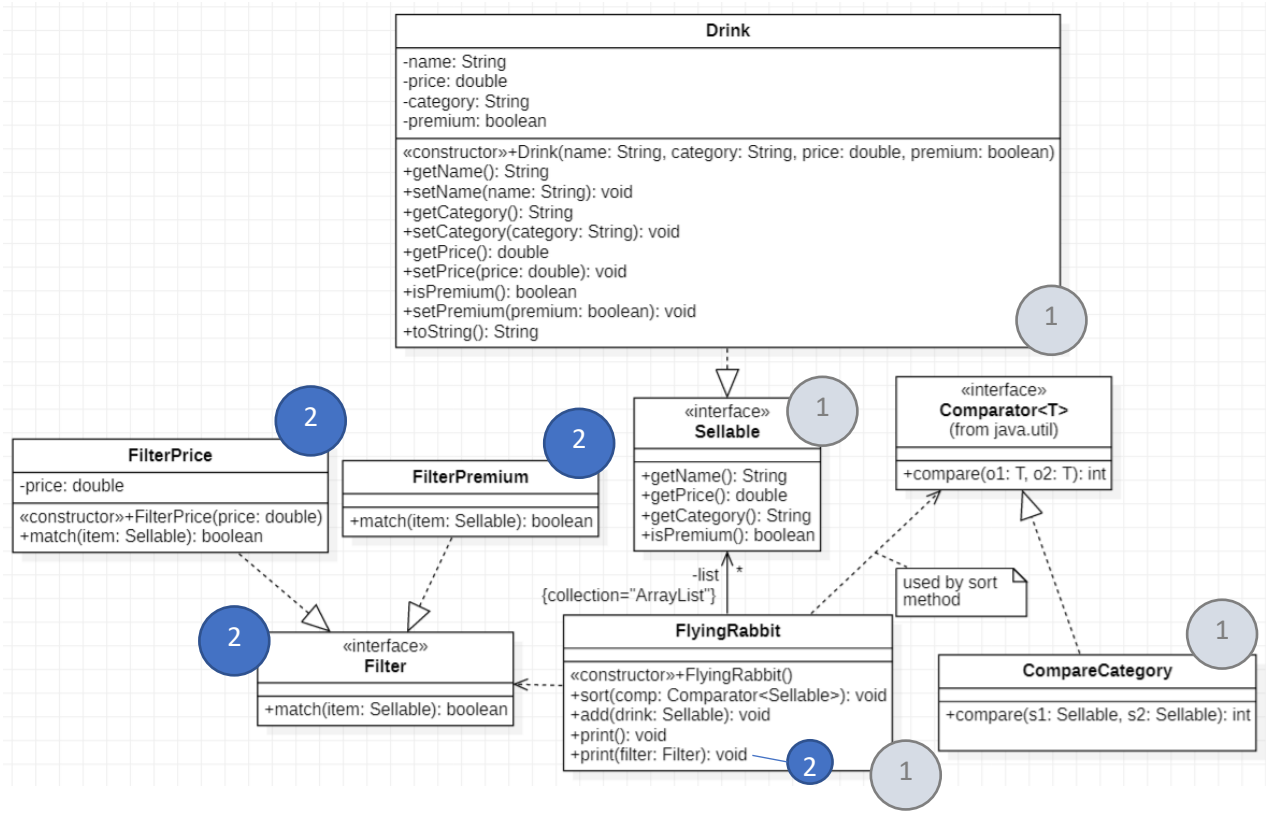
Sample Result:

```
Before sorting:
Item list:
        Hot Coffee     [Double Espresso (Regular 45.0)]
        Protein Shakes [Chocolate Protein Shake (Premium 60.0)]
        Hot Coffee     [Hot Cappucino (Regular 40.0)]
        Hot Coffee     [Hot Caramel Latte (Premium 45.0)]
        Iced Coffee    [Dirty (Regular 35.0)]
        Iced Coffee    [Iced Cappucino (Regular 45.0)]
        Protein Shakes [Matcha Protein Shake (Premium 55.0)]
        Hot Coffee     [Hot Americano (Regular 35.0)]
        Hot Coffee     [Hot Cafe' Latte (Regular 40.0)]
        Protein Shakes [Strawberry Protein Shake (Premium 60.0)]
        Iced Coffee    [Iced Mocha (Regular 40.0)]
        Iced Coffee    [Iced Matcha Cafe Latte (Premium 45.0)]
After sorting by Category:
Item list:
        Hot Coffee     [Double Espresso (Regular 45.0)]
        Hot Coffee     [Hot Cappucino (Regular 40.0)]
        Hot Coffee     [Hot Caramel Latte (Premium 45.0)]
        Hot Coffee     [Hot Americano (Regular 35.0)]
        Hot Coffee     [Hot Cafe' Latte (Regular 40.0)]
        Iced Coffee    [Dirty (Regular 35.0)]
        Iced Coffee    [Iced Cappucino (Regular 45.0)]
        Iced Coffee    [Iced Mocha (Regular 40.0)]
        Iced Coffee    [Iced Matcha Cafe Latte (Premium 45.0)]
        Protein Shakes [Chocolate Protein Shake (Premium 60.0)]
        Protein Shakes [Matcha Protein Shake (Premium 55.0)]
        Protein Shakes [Strawberry Protein Shake (Premium 60.0)]
```

2. หลังจากเสร็จสิ้นการ implemented ตามการออกแบบแรกแล้ว ทีมต้องการเพิ่มความสะดวกให้ลูกค้าสามารถกรองเลือกสินค้าที่ต้องการได้ จากความพยายามออกแบบให้ยืดหยุ่น เพื่อให้สามารถเพิ่มการกรองแบบต่าง ๆ ตามแต่ความต้องการ จึงสร้างอินเทอร์เฟสสำหรับการกรอง และกำหนดคลาสตัวกรองเพื่อเป็นต้นแบบไว้ 2 อย่างคือ กรองด้วยชนิดที่เป็น Premium และกรองตามราคา ตามไดอะแกรมเพิ่มเติม และมีรายละเอียดเพิ่มดังนี้

- **`Filter`** เป็นอินเทอร์เฟสที่กำหนดให้มีเมท็อด `match(Sellable item)` โดยมีเจตนาเพื่อใช้ตอบว่าวัตถุ `item` ที่เป็นพารามิเตอร์เข้า ตรงตามค่าการกรองที่ต้องการหรือไม่ ถ้าตรงคืนค่า `true` ถ้าไม่ตรงคืน `false`
- สร้าง 2 filter classes เพื่อใช้สำหรับการกรอง Sellable ตามเงื่อนไขที่กำหนด
  1) คลาส **`FilterPremium`** มีเมท็อด `match` ที่จะกรองวัตถุ Sellable ที่ค่า `isPremium()` เป็น `true`
  2) คลาส**`FilterPrice`** มีเมท็อด `match` ที่จะกรองวัตถุ Sellable ที่ค่า `getPrice()` ตรงกับค่า price ของมัน (Note: คลาสตัวกรองนี้จะได้ค่า price เมื่อตอนสร้าง) โดยมี
     a) **Constructor** ให้ค่าเริ่มต้นกับตัวแปรวัตถุ `price` ตามพารามิเตอร์ที่ได้รับมา
     b) **`match`** method คืนค่า true ถ้า `getPrice()` ของ `Sellable` มีค่าเท่ากับตัวแปรวัตถุ `price` ของมัน
- แก้ไขคลาส `FlyingRabbit` โดยเพิ่มเมท็อด **`print(Filter f)`** เพื่อให้พิมพ์แบบกรองค่าได้ โดยมันจะพิมพ์ค่าที่ match ตามที่ตัวกรองได้ตรวจสอบ (อีกนัยคือ match ของ Filter f คืนค่า true มาให้)

ไดอะแกรมคลาสของร่างสอง เป็นดังนี้



ตัวอย่างคลาสทดสอบเป็นดังนี้

```
package q2;
import java.util.Scanner;
public class FlyingRabitTest {
	public static void greetingScreen() {
		System.out.println("  1. list all");
		System.out.println("  2. list only premium items");
		System.out.println("  3. list only items matched my price");
		System.out.println("  4. sort items by category and list them: ");
		System.out.println("  5. exit");
		System.out.print("Please choose your choice:");
	}

	public static FlyingRabbit setupFlyingRabbit() {
		FlyingRabbit machine = new FlyingRabbit();
		machine.add(new Drink("Double Espresso", "Hot Coffee", 45, false));
		machine.add(new Drink("Chocolate Protein Shake", "Protein Shakes", 60, true));
		machine.add(new Drink("Hot Cappucino", "Hot Coffee", 40, false));
		machine.add(new Drink("Hot Caramel Latte", "Hot Coffee", 45, true));
		machine.add(new Drink("Dirty", "Iced Coffee", 35, false));
		machine.add(new Drink("Iced Cappucino", "Iced Coffee", 45, false));
		machine.add(new Drink("Matcha Protein Shake", "Protein Shakes", 55, true));
		machine.add(new Drink("Hot Americano", "Hot Coffee", 35, false));
		machine.add(new Drink("Hot Cafe' Latte", "Hot Coffee", 40, false));
		machine.add(new Drink("Strawberry Protein Shake", "Protein Shakes", 60, true));
		machine.add(new Drink("Iced Mocha", "Iced Coffee", 40, false));
		machine.add(new Drink("Iced Matcha Cafe Latte", "Iced Coffee", 45, true));
		return machine;
	}

	public static void main(String[] args) {
		FlyingRabbit machine = setupFlyingRabbit();

		System.out.println("Welcome to Flying Rabbit @TU station");
		Scanner scan = new Scanner(System.in);
		boolean done = false;

		FilterPremium premium = new FilterPremium();
		CompareCategory cc = new CompareCategory();

		do {
			greetingScreen();
			System.out.print("Select your choice (1 to 5):");
			switch (scan.nextLine()) {
			case "1":
				machine.print();
				break;
			case "2":
				machine.print(premium);
				break;
			case "3":
				System.out.print("Filter price. Price that you want: ");
				double price = scan.nextDouble();
				scan.nextLine();
				machine.print(new FilterPrice(price));
				break;
			case "4":
				machine.sort(cc);
				machine.print();
				break;
			case "5":
				System.out.println("Thank you. Bye!");
				done = true;
				break;
			}
```

```
            } while (!done);
        }
}
```

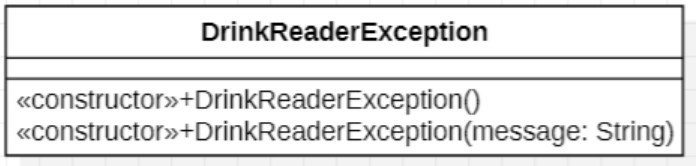
ตัวอย่างผลการทำงานที่ได้

```
Welcome to Flying Rabbit @TU station
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):1
Item list:
        Hot Coffee      [Double Espresso (Regular 45.0)]
        Protein Shakes  [Chocolate Protein Shake (Premium 60.0)]
        Hot Coffee      [Hot Cappucino (Regular 40.0)]
        Hot Coffee      [Hot Caramel Latte (Premium 45.0)]
        Iced Coffee     [Dirty (Regular 35.0)]
        Iced Coffee     [Iced Cappucino (Regular 45.0)]
        Protein Shakes  [Matcha Protein Shake (Premium 55.0)]
        Hot Coffee      [Hot Americano (Regular 35.0)]
        Hot Coffee      [Hot Cafe' Latte (Regular 40.0)]
        Protein Shakes  [Strawberry Protein Shake (Premium 60.0)]
        Iced Coffee     [Iced Mocha (Regular 40.0)]
        Iced Coffee     [Iced Matcha Cafe Latte (Premium 45.0)]
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):2
Filter list:
        Protein Shakes  [Chocolate Protein Shake (Premium 60.0)]
        Hot Coffee      [Hot Caramel Latte (Premium 45.0)]
        Protein Shakes  [Matcha Protein Shake (Premium 55.0)]
        Protein Shakes  [Strawberry Protein Shake (Premium 60.0)]
        Iced Coffee     [Iced Matcha Cafe Latte (Premium 45.0)]
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):3
Filter price. Price that you want: 45
Filter list:
        Hot Coffee      [Double Espresso (Regular 45.0)]
        Hot Coffee      [Hot Caramel Latte (Premium 45.0)]
        Iced Coffee     [Iced Cappucino (Regular 45.0)]
        Iced Coffee     [Iced Matcha Cafe Latte (Premium 45.0)]
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):3
Filter price. Price that you want: 70
Filter list:
--No match!
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):4
Item list:
        Hot Coffee      [Double Espresso (Regular 45.0)]
        Hot Coffee      [Hot Cappucino (Regular 40.0)]
        Hot Coffee      [Hot Caramel Latte (Premium 45.0)]
```

```
        Hot Coffee      [Hot Americano (Regular 35.0)]
        Hot Coffee      [Hot Cafe' Latte (Regular 40.0)]
        Iced Coffee     [Dirty (Regular 35.0)]
        Iced Coffee     [Iced Cappucino (Regular 45.0)]
        Iced Coffee     [Iced Mocha (Regular 40.0)]
        Iced Coffee     [Iced Matcha Cafe Latte (Premium 45.0)]
        Protein Shakes  [Chocolate Protein Shake (Premium 60.0)]
        Protein Shakes  [Matcha Protein Shake (Premium 55.0)]
        Protein Shakes  [Strawberry Protein Shake (Premium 60.0)]
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):1
Item list:
        Hot Coffee      [Double Espresso (Regular 45.0)]
        Hot Coffee      [Hot Cappucino (Regular 40.0)]
        Hot Coffee      [Hot Caramel Latte (Premium 45.0)]
        Hot Coffee      [Hot Americano (Regular 35.0)]
        Hot Coffee      [Hot Cafe' Latte (Regular 40.0)]
        Iced Coffee     [Dirty (Regular 35.0)]
        Iced Coffee     [Iced Cappucino (Regular 45.0)]
        Iced Coffee     [Iced Mocha (Regular 40.0)]
        Iced Coffee     [Iced Matcha Cafe Latte (Premium 45.0)]
        Protein Shakes  [Chocolate Protein Shake (Premium 60.0)]
        Protein Shakes  [Matcha Protein Shake (Premium 55.0)]
        Protein Shakes  [Strawberry Protein Shake (Premium 60.0)]
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):5
Thank you. Bye!
```

3. ในเฟส 3 ทีมตกลงว่าแทนการกำหนดค่าเครื่องดื่มแบบแน่นอน จะให้โปรแกรมสามารถอ่านเครื่องดื่มจากไฟล์ได้ จึงได้ออกแบบคลาสตัวอ่าน และคลาสความผิดพลาดเฉพาะ เมื่อเกิดการผิดพลาดจากการอ่านจะได้โยนให้เมท็อดที่เรียกใช้จัดการความผิดพลาดอย่างเหมาะสม โดยออกแบบไว้ดังในไดอะแกรม และมีรายละเอียดเพิ่มดังนี้

   - คลาส **`DrinkReaderException`** ที่เป็น **checked** exception และมี constructor อย่างน้อย 2 ตัว คือแบบ default และแบบที่รับ String เป็นพารามิเตอร์ โดยข้อความปริยาย ให้กำหนดได้เองตามความเหมาะสม



   - คลาส **`DrinkReader`** ทำหน้าที่เป็นส่วนต่อประสานกับผู้ใช้เพื่อรับชื่อไฟล์ข้อมูลเครื่องดื่ม โดยไฟล์ข้อมูลเป็นเท็กซ์ไฟล์ แต่ละบรรทัดแทนรายการเครื่องดื่ม 1 รายการ ประกอบด้วยชื่อเครื่องดื่ม ประเภท ราคา และการเป็น premium (ซึ่งอาจมีหรือไม่ก็ได้ ถ้าไม่มีให้ premium เป็น *`false`*) แต่ละส่วนของข้อมูลคั่นด้วยเครื่องหมาย `::` เมื่ออ่านได้แล้วสร้างเป็น `Drink` เพื่อเพิ่มในอาร์เรย์ลิสต์ `readerList` ของมัน ในกรณีที่

ข้อมูลในบรรทัดนั้นมีรูปแบบไม่ถูกต้องจะข้ามรายการ แสดงให้ผู้ใช้ทราบว่ามีข้อมูลไม่ถูกต้อง แต่ยังคงอ่านไฟล์และเพิ่มรายการที่ถูกต้องต่อจนหมด

รูปแบบของแต่ละบรรทัดของรายการเครื่องดื่ม

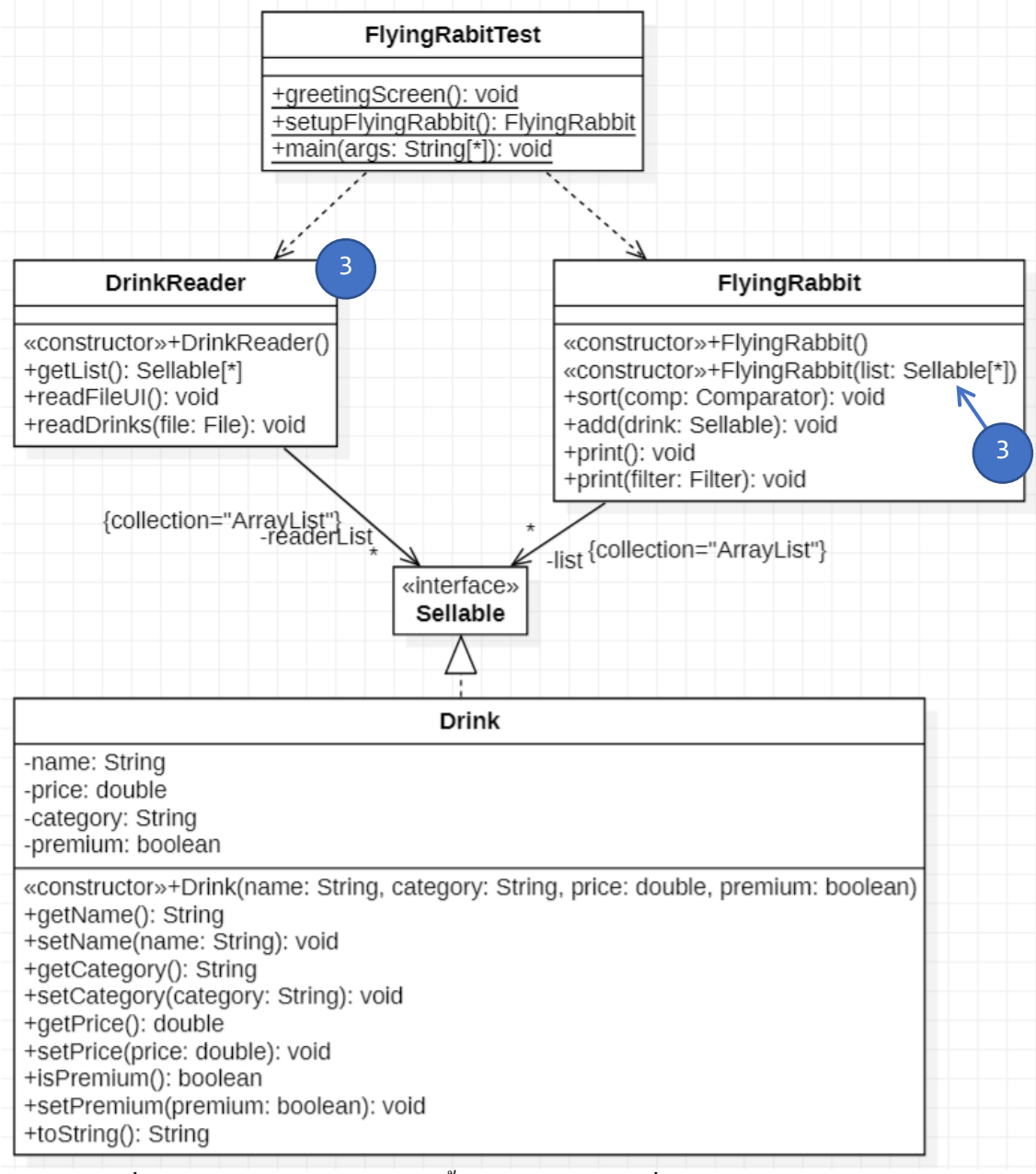
ชื่อเครื่องดื่ม`::`*ประเภท*`::`*ราคา*`::`*ความเป็น* `Premium`

ตัวอย่างเช่น ดับเบิ้ลเอสเปรสโซ่ ประเภทกาแฟร้อน ราคา 45 บาท ไม่ใช่เครื่องดื่ม Premium (อาจไม่ระบุได้ จะถือว่าใช้ค่าปริยาย false)

```
Double Espresso::Hot Coffee::45::false
```

ตัวอย่างเช่น ชอคโกเลตโปรตีนเชค ประเภท โปรตีนเชค ราคา 60 บาท เป็นเครื่องดื่ม Premium

```
Chocolate Protein Shake::Protein Shakes::60::true
```



Note: คลาสบางตัวที่ไม่ปรากฏในภาพไดอะแกรมนี้ ยังคงต้องมีอยู่แต่เนื่องจากไม่ได้แก้ไขจากเดิมจึงไม่แสดงในที่นี้

คลาส DrinkReader มีตัวแปรวัตถุ readerList ชนิด ArrayList<Sellable> และมีเมท็อดดังนี้

- **getList** คืนค่า ArrayList<Sellable> จาก readerList
- **readUI** ทำหน้าที่เป็นส่วนต่อประสานรับชื่อไฟล์จากผู้ใช้เพื่ออ่านข้อมูลเครื่องดื่มจากไฟล์ โดยเครื่องดื่มที่ถูกต้องจะเพิ่มเข้าในลิสต์ของ readerList ในการรับชื่อไฟล์หากพบข้อผิดพลาด (ไม่พบไฟล์) จะให้ผู้ใช้กรอกใหม่ โดยจะรับค่าไม่เกิน 3 ครั้ง หากยังผิดอยู่โยนความผิดพลาด DrinkReaderException หากพบไฟล์อ่านรายการเพิ่มเป็นรายการเครื่องดื่มให้กับ readerList โดยเรียกเมท็อด readDrinks
- **readDrinks** อ่านไฟล์ที่ละบรรทัด แปลงข้อมูลรายการที่ถูกต้องให้เป็น Drink เพิ่มลงในลิสต์ของ readerList กรณีอ่านพบความผิดพลาดอ่านข้าม

- แก้ไขคลาส FlyingRabbit ให้มี Constructor ที่รับค่า ArrayList<Sellable> เพื่อกำหนดค่าให้กับ list ได้จากภายนอก จากนั้นเมท็อด setupFlyingRabbit ของคลาสทดสอบเพื่อให้ได้การตั้งเครื่องจากการอ่านไฟล์แทนการให้ค่าแน่นอน

```
	public static FlyingRabbit setupFlyingRabbit() throws DrinkReaderException {
		DrinkReader reader = new DrinkReader();
		reader.readFileUI();
		FlyingRabbit machine = new FlyingRabbit(reader.getList());
		return machine;
	}
```

สมมติให้ในไฟล์ src\Drink.txt มีเนื้อหาดังนี้ (2 รายการตัวหนังสือสีแดงคือรายการที่ไม่ถูกต้อง)

```
Name::Category::0::true
Espresso::Hot Coffee::35::false
Chocolate Milk Shake::Cool Drink::60::true
Hot Plain Milk::Hot Milk::false
Honey Lemon Iced Tea::Ice Tea::40::false
Double Chocolate Milk Shake::Iced Milk::50::true
Hot Chinese Tea::Hot Tea::30:: false
Iced Caramel Milk::Iced Milk::40
Iced Mocha::Iced Coffee::40::false
Iced Coffee::30
Hot Americano::Hot Coffee::40::false
```

ตัวอย่างผลการทำงาน กรณีที่อ่านไฟล์ได้

```
File containing drinks: Drink.txt
Try again! Cannot find the file Drink.txt
File containing drinks: src/Drink.txt
--Incorrect price skip: Hot Plain Milk::Hot Milk::false
--Skip the item: Iced Coffee::30
Welcome to Flying Rabbit @TU station
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):1
```

```
Item list:
       Category       [Name (Premium 0.0)]
       Hot Coffee     [Espresso (Regular 35.0)]
       Cool Drink     [Chocolate Milk Shake (Regular 60.0)]
       Ice Tea        [Honey Lemon Iced Tea (Regular 40.0)]
       Iced Milk      [Double Chocolate Milk Shake (Premium 50.0)]
       Hot Tea        [Hot Chinese Tea (Regular 30.0)]
       Iced Milk      [Iced Caramel Milk (Regular 40.0)]
       Iced Coffee    [Iced Mocha (Regular 40.0)]
       Hot Coffee     [Hot Americano (Regular 40.0)]
  1. list all
  2. list only premium items
  3. list only items matched my price
  4. sort items by category and list them:
  5. exit
Please choose your choice:Select your choice (1 to 5):5
Thank you. Bye!
```

ตัวอย่างผลการทำงาน กรณีที่พยายามอ่านไฟล์ 3 ครั้งแล้วไม่สำเร็จ

```
File containing drinks: drink.txt
Try again! Cannot find the file drink.txt
File containing drinks: drinks.txt
Try again! Cannot find the file drinks.txt
File containing drinks: drink.txt
Cannot find! drink.txt gave up!
```