Take Assessment: Exercise 3

Please answer the following question(s).
If the assessment includes multiple-choice questions, click the "Submit Answers" button when you have completed those questions.

1. Go to bottom of question.

Implementing the Gourmet Coffee System

Prerequisites, Goals, and Outcomes

Prerequisites: Before you begin this exercise, you need mastery of the following:

- *Object Oriented Programming*
  - Knowledge of class design
    - Class attributes
    - Constructors
    - Accessor methods
    - Mutator methods
  - Knowledge of inheritance
    - How to implement a specialization/generalization relationship using inheritance

Goals: Reinforce your ability to implement Java classes using inheritance.

Outcomes: You will demonstrate mastery of the following:

- Implementing the constructors, accessors, and mutators of a Java class
- Using inheritance to implement a specialization/generalization relationship

Background

This assignment asks you to implement some of the classes in the *Gourmet*

Coffee System specified on Exercise 2.

Description

In this assessment, you will implement the classes and relationships illustrated in the following class diagram:

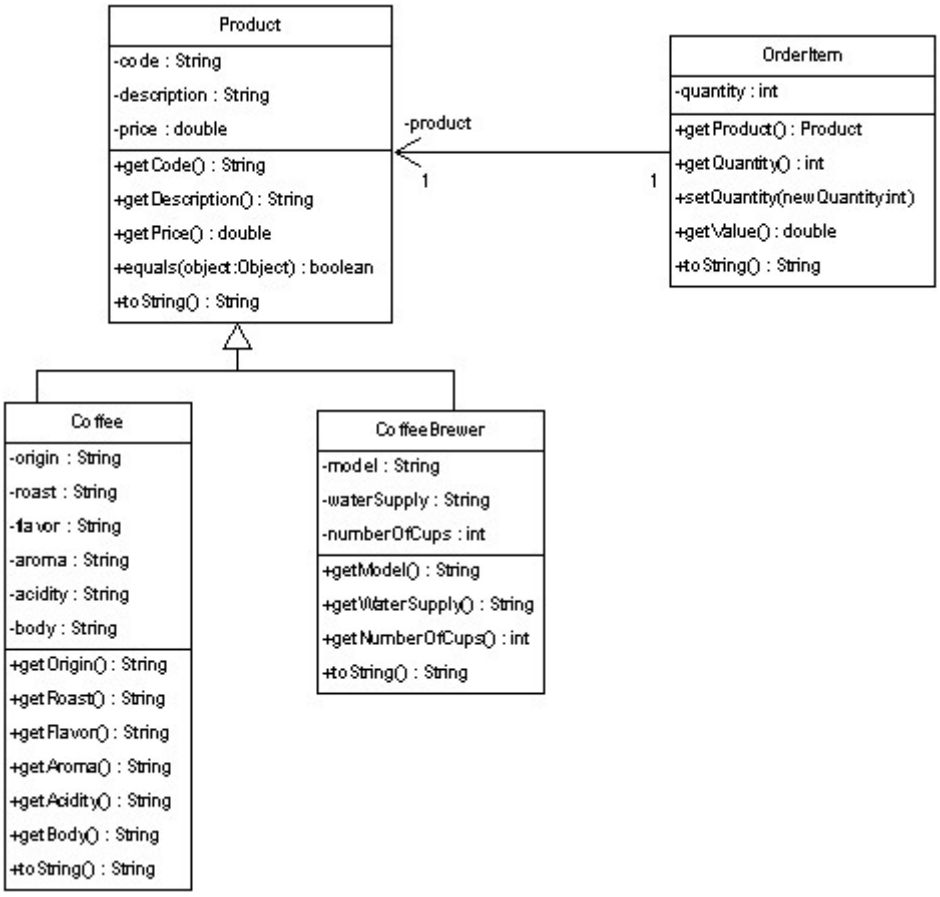

Figure 1 *Portion of Gourmet Coffee System class diagram*

The class specifications are as follows:

Class Product

The class Product models a generic product in the store.

Instance variables:

- *code*. The unique code that identifies the product
- *description*. A short description of the product
- *price*. The price of the product

Constructor and methods:

- *public Product(String initialCode,*
- *String initialDescription,*
- *double initialPrice)*

  Constructor that initializes the instance variables code, description, and price.

- *public String getCode()*. Returns the value of instance variable code.

- *public String getDescription()*. Returns the value of instance variable description.

- *public double getPrice()*. Returns the value of instance variable price.

- *boolean equals(Object object)*. Overrides the method equals in the class Object. Two Product objects are equal if their codes are equal.

- *String toString()*. Overrides the method toString in the class Object. Returns the string representation of a Product object. The String returned has the following format:

  code_description_price

  The fields are separated by an underscore (_). You can assume that the fields themselves do not contain any underscores.

Class Coffee

The class Coffee models a coffee product. It extends class Product.

Instance variables:

- *origin*. The origin of the coffee
- *roast*. The roast of the coffee
- *flavor*. The flavor of the coffee
- *aroma*. The aroma of the coffee
- *acidity*. The acidity of the coffee
- *body*. The body of the coffee

Constructor and methods:

- *public Coffee(String initialCode,*
- *String initialDescription,*
- *double initialPrice,*
- *String initialOrigin,*
- *String initialRoast,*
- *String initialFlavor,*
- *String initialAroma,*
- *String initialAcidity,*
- *String initialBody)*

  Constructor that initializes the instance variables code, description, price, origin, roast, flavor, aroma, acidity, and body.

- *public String getOrigin()*. Returns the value of instance variable origin.

- *public String getRoast()*. Returns the value of instance variable roast.

- *public String getFlavor()*. Returns the value of instance variable flavor.

- *public String getAroma()*. Returns the value of instance variable aroma.

- *public String getAcidity()*. Returns the value of instance variable acidity.

- *public String getBody()*. Returns the value of instance variable body.

- *String toString()*. Overrides the method toString in the class Object. Returns the string representation of a Coffee object. The String returned has the following format:

  code_description_price_origin_roast_flavor_aroma_acidity_body

  The fields are separated by an underscore (_). You can assume that the fields themselves do not contain any underscores.

Class CoffeeBrewer

Class CoffeeBrewer models a coffee brewer. It extends class Product.

Instance variables:

- *model*. The model of the coffee brewer
- *waterSupply*. The water supply (*Pour-over* or *Automatic*)
- *numberOfCups*. The capacity of the coffee brewer

Constructor and methods:

- public CoffeeBrewer(String initialCode,
- String initialDescription,
- double initialPrice,
- String initialModel,
- String initialWaterSupply,
- int initialNumberOfCups)

  Constructor that initializes the instance variables code, description, price, model, waterSupply, and numberOfCups.

- *public String getModel()*. Returns the value of instance variable model.

- *public String getWaterSupply()*. Returns the value of instance variable waterSupply.

- *public int getNumberOfCups()*. Returns the value of instance variable numberOfCups.

- *String toString()*. Overrides the method toString in the class Object. Returns the string representation of a CoffeeBrewer object. The String returned has the following format:

  code_description_price_model_waterSupply_numberOfCups

  The fields are separated by an underscore (_). You can assume that the fields themselves do not contain any underscores.

Class OrderItem

Class OrderItem models an item in an order.

Instance variables:

- *product*. This instance variable represents the one-way

association between OrderItem and Product. It contains a reference to a Product object.

- *quantity*. The quantity of the product in the order.

Constructor and methods:

- *public OrderItem(Product initialProduct,*
- *int initialQuantity)*

  Constructor that initializes the instance variables product and quantity.

- *public Product getProduct()*. Returns the value of the instance variable product, a reference to a Product object.

- *public int getQuantity()*. Returns the value of the instance variable quantity.

- *public void setQuantity(int newQuantity)*. Sets the instance variable quantity to the value of parameter newQuantity.

- *public double getValue()*. Returns the product of quantity and price.

- *String toString()*. Overrides the method toString in the class Object. Returns the string representation of an OrderItem object. The String representation has the following format:

  quantity product-code product-price

  The fields are separated by a space. You can assume that the fields themselves do not contain any spaces.

Test driver classes

Complete implementations of the following test drivers are provided in the student archive. Use these test drivers to verify that your code works correctly.

- Class TestProduct
- Class TestCoffee
- Class TestCoffeeBrewer
- Class TestOrderItem

Files

The following files are needed to complete this assignment:

- *student-files.zip* — Download this file. This archive contains the following:
  - *TestProduct.java*
  - *TestCoffee.java*
  - *TestCoffeeBrewer.java*
  - *TestOrderItem.java*

Tasks

Implement classes Product, Coffee, CoffeeBrewer, and OrderItem. Document using Javadoc and follow Sun's code conventions. The following steps will guide you through this assignment. Work incrementally and test each increment. Save often.

1. **Extract** the files by issuing the following command at the command prompt:

   ```
   C:\>unzip student-files.zip
   ```

2. **Then**, implement class Product from scratch. Use TestProduct driver to test your implementation.

3. **Next**, implement class Coffee from scratch. Use TestCoffee driver to test your implementation.

4. **Then**, implement class CoffeeBrewer from scratch. Use TestCoffeeBrewer driver to test your implementation.

5. **Finally**, implement class OrderItem from scratch. Use TestOrderItem driver to test your implementation.

Submission

Upon completion, submit **only** the following:

1. Product.java
2. Coffee.java
3. CoffeeBrewer.java

4. OrderItem.java

Go to top of question.

File to submit: Upload File | Forward File | Refresh | Ready for Grading

---

Go to top of assessment.