Introduction to Classes Individual Exercises

The purpose of this exercise is to provide you the opportunity to define and use classes. The names of the classes and methods are specified in the Evaluation Criteria & Functional Requirements section.

Learning Objectives

After completing this exercise, students will understand:

- How to create classes.
- How to create attributes.
- How to create "getters" and "setters".
- How to override a default constructor.
- How to manage the state of an object.
- How to limit access to attributes through the use of access modifiers.

Evaluation Criteria & Functional Requirements

- The project must not have any build errors.
- Unit tests pass as expected.
- Appropriate variable names and data types are being used.
- Code is presented in a clean, organized format.
- If an attribute does not have `set` marked, then a setter should not be defined.
- The code meets the specifications defined below.

Difficulty - Easy

Company

Data Members

Attribute	Data Type	Get	Set	Description
name	string	X	X	The company name.
numberOfEmployees	int	X	X	The number of employees at the company.
revenue	double	X	X	The annual revenue of the company.
expenses	double	X	X	The annual expenses of the company.

Methods

```
public String getCompanySize()
public double getProfit()
```

Notes

- `getCompanySize()` returns "small" if 50 or less employees, "medium" if between 51 and 250 employees, "large" if greater than 250 employees.
- `getProfit()` returns the result of revenue - expenses.

Constructor

The `Company` class uses the default constructor.

Person

Data Members

Attribute	Data Type	Get	Set	Description
firstName	String	X	X	The first name of the person.
lastName	String	X	X	The last name of the person.
age	int	X	X	The age of the person.

Methods

```
public String getFullName()
public boolean isAdult()
```

Notes

- `getFullName()` returns the `lastName + ", " + firstName`.
- `isAdult()` returns `true` if the person is 18 or older.

Constructors

The `Person` class uses the default constructor.

Product

Data Members

Attribute	Data Type	Get	Set	Description
name	String	X	X	The name of the product.
price	double	X	X	The price of the product.
weightInOunces	double	X	X	The weight (in ounces) of the product.

Methods

Notes

- There are no additional methods beyond the basic getters and setters.

Constructors

The `Product` uses the default constructor.

Difficulty - Medium

Dog

Data Members

Attribute	Data Type	Get	Set	Description
sleeping	boolean	X		`TRUE` if the dog is asleep. `FALSE` if not.

Notes

- All new dogs are awake by default.

Methods

```
public String makeSound()
public void sleep()
public void wakeUp()
```

Notes

- `makeSound()` returns `"Zzzzz..."` if the dog is asleep. Returns `"Woof!"` if the dog is awake.
- `sleep()` sets `sleeping` to `true`.
- `wakeUp()` sets `sleeping` to `false`.

Constructor

The `Dog` class uses the default constructor.

ShoppingCart

Data Members

Attribute	Data Type	Get	Set	Description
totalNumberOfItems	int	X	X	The number of items in the shopping cart.
totalAmountOwed	double	X	X	The total amount owed.

Notes

- All shopping carts have total 0 items and 0.0 amount owed by default.

Methods

```
public double getAveragePricePerItem()
public void addItems(int numberOfItems, double pricePerItem)
public void empty()
```

Notes

- `getAveragePricePerItem()` returns the result of `totalAmountOwed / totalNumberOfItems`.
- `addItems(int numberOfItems, double pricePerItem)` updates `totalNumberOfItems` and increases `totalAmountOwed` by `(pricePerItem * numberOfItems)`
- `empty()` resets `totalNumberOfItems` to 0 and `totalAmountOwed` to 0.0.

Constructor

The `ShoppingCart` class uses the default constructor.

Difficulty - Difficult

Calculator

Data Members

Attribute	Data Type	Get	Set	Description
currentValue	int	X		The current calculated value.

Notes

- All calculators have 0 as currentValue by default.

Methods

```
public int add(int addend)
public int multiply(int multiplier)
```

```
public int subtract(int subtrahend)
public int power(int exponent)
public void reset()
```

Notes

- `add(int addend)` returns the new `currentValue` after performing the addition.
- `multiply(int multiplier)` returns the new `currentValue` after performing the multiplication.
- `subtract(int subtrahend)` returns the new `currentValue` after performing the subtraction.
- `power(int exponent)` returns the new `currentValue` after raising the `currentValue` by the exponent.
- `void reset()` resets the `currentValue` to 0.

Constructor

The `Calculator` class uses the default constructor.

Getting Started

- Import the introduction-to-classes-exercises project into Eclipse.
- Right-click on the project, and select the **Run As -> JUnit Test** menu option.
- Click on the **JUnit** tab to see the results of your tests and which passed / failed.
- Provide enough code to get a test passing.
- Repeat until all tests are passing.

Tips and Tricks

- **Note, If you find yourself stuck on a problem for longer than fifteen minutes, move onto the next, and try again later.**
- In this exercise, you will be provided with a series of specification for classes you will be responsible for creating. Each class comes with its own challenge and nuance, and you may find that some of these classes are more challenging than others to implement.
- In this exercise, you will be creating the classes specified in the requirements section of this document. The unit tests you run will verify if you have defined the classes correctly. As you work on creating the classes, be sure to run the tests, and then provide enough code to pass the test. For instance, if you are working on the Product class, provide enough code to get one of the Product tests passing. By focusing on getting a single test to pass at a time, you will save yourself a lot of time, as this forces you to only focus on what is important for the test you are currently working on. This is commonly referred to as **Test Driven Development**, or **TDD**.
- What happens if you don't define any constructors in a class? Does your code still work as expected? Why or why not?
- When you provide an explicit constructor with arguments in a class, what happens to the default constructor (a constructor with no arguments)?
- Be mindful of your access modifiers.