Introduction to Classes 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 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

Constructors

Signature	Description
Company()	Default constructor.

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

Method Name	Return Type	Description
getCompanySize()	String	Returns "small" if 50 or less employees, "medium" if between 51 and 250 employees, "large" if greater than 250 employees.
getProfit()	double	Returns the result of revenue - expenses.

Person

Constructors

Signature	Description
Person()	Default constructor.

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

Method Name	Return Type	Description
getFullName()	string	Returns the first and last name of the person separated by a space. For instance, "Sonny Koufax".
isAdult()	bool	Returns `true` if the person is 18 or older.

Product

Constructors

Signature	Description
Product()	Default constructor.

Data Members

Attribute	Data Type	Get	Set	Description
name	String	X	X	The name of the product.

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

Methods

The `Product` class does not have any additional methods beyond the basic getters and setters.

Difficulty - Medium

Dog

Constructors

Signature	Description
Dog()	Default constructor. **All new dogs are awake by default.**

Data Members

Attribute	Data Type	Get	Set	Description
isSleeping	boolean	X		`true` if the dog is asleep, `false` if not. **All new dogs are awake by default.**

Methods

Method Name	Return Type	Description
makeSound()	string	Returns `"Zzzzz..."` if the dog is asleep. Returns `"Woof!"` if the dog is awake.
sleep()	void	Sets `isSleeping` to `true`.
wakeUp()	void	Sets `isSleeping` to `false`.

ShoppingCart

Constructors

Signature	Description
ShoppingCart()	Default constructor.

Data Members

Attribute	Data Type	Get	Set	Description
totalNumberOfItems	int	X		The number of items in the shopping cart. **All shopping carts have 0 items by default.**
totalAmountOwed	double	X		The total for the shopping cart. **All shopping carts have 0.0 owed by default.**

Methods

Method Name	Return Type	Description
getAveragePricePerItem()	decimal	Returns the result of `totalAmountOwed / totalNumberOfItems`.
addItems(int numberOfItems, double pricePerItem)	void	Updates `totalNumberOfItems` and increases `totalAmountOwed` by `(pricePerItem * numberOfItems)`
empty()	void	Resets `totalNumberOfItems` and `totalAmountOwed` to 0.

Difficulty - Difficult

Calculator

Constructors

Signature	Description
Calculator()	Default constructor. All calculators have 0 for result by default.

Data Members

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

Methods

Method Name	Return Type	Description
add(int addend)	int	Adds `addend` to `result` and returns the current value of `result`.
getResult()	int	Returns the current value of `result`.

Method Name	Return Type	Description
multiply(int multiplier)	int	Multiplies current result by `multiplier` and returns the current value of `result`.
power(int exponent)	int	Raises `result` to power of `exponent`. Negative exponents should use the absolute value. Returns the current value of `result`
reset()	void	Resets `result` to 0.
subtract(int subtrahend)	int	Subtracts `subtrahend` from the current value of `result` and returns the current value of `result`.

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'll be provided with a series of specification for classes you need to create. 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, create the classes specified in the requirements section of this document. The unit tests you run 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. Focusing on getting a single test to pass at a time saves a lot of time, as this forces you to only focus on what is important for the test you're 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.
