**Section 1: Define / Answer**

Parameter:

Parameters are the variables that are listed as part of a method declaration. Each parameter must have a unique name and a defined data type.

Argument:

Arguments is a list of Parameters that can be passed to your Java Programm at start up.

In Java, what is the difference between an object and a class?

A **class** is a blueprint which you use to create **objects**. An object is an **instance** of a class

Explain the difference between Procedural Programming and Object Orientated programming-

Procedural programming is a programming paradigm, derived from structured programming, based upon the concept of the procedure call.

Object-oriented programming (OOP) is a programming language model organized around "objects" rather than "actions" and data rather than logic.

Pg. 577**,** Java Programming *A comprehensive Introduction*

[**http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format**](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format)(Detailed explanation of Java documentation)

[**http://www.tutorialspoint.com/java/java\_documentation.htm**](http://www.tutorialspoint.com/java/java_documentation.htm)

[**http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments**](http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments)

**Javadoc tags (Examples)**

|  |  |  |
| --- | --- | --- |
| **Tag** | **Description** | **Syntax** |
| @author | Adds the author of a class. | @author name-text |
| {@code} | Displays text in code font without interpreting the text as HTML markup or nested javadoc tags. | {@code text} |
| {@docRoot} | Represents the relative path to the generated document's root directory from any generated page | {@docRoot} |
| @deprecated | Adds a comment indicating that this API should no longer be used. | @deprecated deprecated-text |
| @exception | Adds a **Throws** subheading to the generated documentation, with the class-name and description text. | @exception class-name description |
| {@inheritDoc} | Inherits a comment from the **nearest** inheritable class or implementable interface | Inherits a comment from the immediate surperclass. |
| {@link} | Inserts an in-line link with visible text label that points to the documentation for the specified package, class or member name of a referenced class. T | {@link package.class#member label} |
| {@linkplain} | Identical to {@link}, except the link's label is displayed in plain text than code font. | {@linkplain package.class#member label} |
| @param | Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section. | @param parameter-name description |
| @return | Adds a "Returns" section with the description text. | @return description |
| @see | Adds a "See Also" heading with a link or text entry that points to reference. | @see reference |
| @serial | Used in the doc comment for a default serializable field. | @serial field-description | include | exclude |
| @serialData | Documents the data written by the writeObject( ) or writeExternal( ) methods | @serialData data-description |
| @serialField | Documents an ObjectStreamField component. | @serialField field-name field-type field-description |
| @since | Adds a "Since" heading with the specified since-text to the generated documentation. | @since release |
| @throws | The @throws and @exception tags are synonyms. | @throws class-name description |
| {@value} | When {@value} is used in the doc comment of a static field, it displays the value of that constant: | {@value package.class#field} |
| @version | Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used. | @version version-text |

Task 1- Start to construct complete programs and an introduction to Object Orientated programming. Think about the overall functioning of the program. Use Assignment #12 Task1 as the bases for this exercise.

Create a **do**-**while** loop / with **switch case** statements that operate the program.

You will a multi-level menu operation using **do-while** implementation.

Present the user with a menu and options. Based upon the options selected by the user the program should operate correctly.

Create a computer program that will calculate the range for 3 different vehicles.

Use object orientated programming design to solve the problem.

Set-up the program so the user can manually input the values for passengers, fuel capacity, mpg for the 3 created vehicles.

Create a **void or return** method inside the “programmer created “ class to calculate vehicle range**.**

**range =** **fuel capacity \* miles per gallon**.

Each Vehicle type should have unique values for number of passengers, fuel capacity, and miles per gallon.

Attach Snipping photos as the program operates, including menu prompts, outputs etc.

**Sample Output: // Create similar output for 3 Vehicle Types**

**On next page-**

**Change input values now that we are creating the same program multiple times.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Main Menu: \***

**\* Enter # to run program or Quit \***

**\* 1) Enter Fuel Capacity \***

**\* 2) Enter Miles Per Gallon \***

**\* 3) Calculate Range \***

**\* 4) Quit \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1**

**You Selected Option 1:**

**Enter fuel capacity in Integers Please**

**25**

**You entered: 25**

**2**

**You Selected Option 2:**

**Enter Miles Per Gallon Please**

**29**

**You entered: 29**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Main Menu: \*

\* Enter # to run program or Quit \*

\* 1) Enter Fuel Capacity \*

\* 2) Enter Miles Per Gallon \*

\* 3) Calculate Range \*

\* 4) Quit \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1

You Selected Option 1:

Enter fuel capacity in Integers Please

15

You entered: 15

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Main Menu: \*

\* Enter # to run program or Quit \*

\* 1) Enter Fuel Capacity \*

\* 2) Enter Miles Per Gallon \*

\* 3) Calculate Range \*

\* 4) Quit \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2

You Selected Option 2:

Enter Miles Per Gallon Please

45

You entered: 45

