

Extended Project (70%)

|  |  |
| --- | --- |
| **Course Identification** | |
|  |  |
| Name of programs – Codes: | COMPUTER SCIENCE TECHNOLOGY – VIDEO GAME PROGRAMMING – 420.AX – 420.BX |
| Course title: | OBJECT-ORIENTED PROGRAMMING AND DATABASE APPLICATION |
| Course number: | 420-B35-AS |
| Group: | 07127 |
| Teacher’s name: | Houria Houmel |
| Duration: | Extended Project |
| Semester: | Fall 2021 |
| **Student Identification** | |
| Name: Oussama Arouch Student number: 201911899  Date: December 14, 2021. Result: \_\_\_\_\_\_\_\_\_\_\_\_\_\_  I declare that this is an original work, and that I credited all content sources of which I am not the author (online and printed, images, graphics, films, etc.), in the required quotation and citation style for this work. | |
| **Standard of the Evaluated Competency** | |
|  | |

**Statement of the evaluated competency – Code** Ensure the quality of an application – 0177

Evaluated elements of the competency

1. Plan tests
2. Run the various tests.
3. Verify the quality of the application.

Design and develop an application in a database environment. (017B)

Evaluated elements of the competency

1. Define the functions of the application.
2. Define the technological requirements
3. Lay the groundwork for the application
4. Model the application.
5. Produce a user interface using a prototype.
6. Develop programs
7. Produce documentation for the application

|  |
| --- |
| **Instructions** |
|  |
| • Plagiarism, attempts at plagiarism or complicity in plagiarism during a summative evaluationresults in a mark of zero (0). In the case of recidivism, in the same course or in another course,the student will be given a grade of '0' for the course in question. (IPEL – Article 5.16).• Any screenshot sent in place of your work will be refused.• Make sure to save the latest version before handing over.• Make sure you have compressed the FleetOfVehicles project• Make sure you have sent the compressed FleetOfVehicles in the Assignment Section before the deadline. |
| **Mark Breakdown** |
|  |
| This evaluation is on 100 points, distributed as follows:   * **Part 1**: Fo**r a total of 60 points**   Q1, Q9 (10 points each). For a total of 20 points  Q2, Q3. (2.5 points each). For a total of 5 points  Q4. Q5, Q6, Q7, Q8, Q10, Q11 (5 points each). For a total of 35 points     * **Part 2**: **For a total of 10 points**   Q1, Q2. (5 points each). For a total of 10 points   * **Part 3**: **For a total of 30 points**   Q1, Q2, Q3. (10 points each). For a total of 30 points        **TOTAL: 100 POINTS** |
|  |
|  |

Case study: ***Vehicles*** ***Fleet Factory***

*Vehicles*’ Fleet Factory manufactures different types of vehicles.  
A ***vehicle*** is identified by a serial number, make, model, date of manufacture, and trip counter where trip counter is accessible by the subclasses.

A ***gasoline vehicle*** is identified by a ***fuel consumed*,** while an ***electric vehicle*** is identified by a ***kilowatt power consumed****.*

The list of features is not exhaustive, you must add at least one feature to each class.

A vehicle could run on gasoline or electric batteries. Therefore, we have two different classes of vehicles.

Each type would have its own ***fuel-efficiency measure*.**For **gas vehicles**, it is the ***gasoline consumed per kilometer***, and for **electric vehicles** it is the ***kilowatts of power consumed per kilometer.***  
Because this is the common functionality and must be implemented by every type of vehicle, including those that will come on the market in the future such as hybrid vehicles, you must create a standard interface that every vehicle will implement. You will call this interface ***IMileageEfficiency****.*

*Any vehicle that implements this interface will get a standard set of methods for obtaining the* ***vehicle’s efficiency****. To keep thing short and simple, you will define this interface with a single method, as follows:***floatgetMilesPerEnergyConsumed*().***

Therefore, the Vehicle class will implement this interface and the Gasoline Vehicle and Electric Vehicle subclasses will provide an appropriate implementation for its method  
 << getMilesPerEnergyConsumed >>

Applying the polymorphism.

The two subclasses have common behavior: get miles per energy consumed   
The ***Gasoline Vehicle*** named **GasVehicle** must implement the getMilesPerEnergyConsumed () method as follows:  
 double getMilesPerEnergyConsumed () {return tripCounter / fuelConsumed;}

The ***Electric Vehicle*** named **ElecVehicle** must implement the getMilesPerEnergyConsumed () method as follows:  
 double getMilesPerEnergyConsumed () {return tripCounter / kwPowerConsumed;}  
  
You have been assigned the job to design and implement a multi-tiers application called Fleet of vehicles.

**Part 1. The Business Layer:**

**Class Library Business Layer**

**Using Eclipse**:

* Create a workspace named **workspace-vehicle-fleets**
* Create a class library project named **VehicleFleeLibrary**
* Configure the build path to add the **ojdbc driver** to the classpath.
* Create the packages: bus and data that must contain business and database classes

**Question 1: (10 points)**

**Class Diagram Design**

Design the **class diagram** using **visual paradigm**

**Question 2: (2.5 points)**

**Interface:**

Create an **interface** called **IMileAgeEffficiency** with the following signatures:

* **public abstract void MakeTrip()**
* **public abstract void MakeTrip(int tripCounter, double energyConsumed)**
* **public abstract double GetMilesPerEnergyConsumed** **()**

**Question 3: (2.5 points)  
Exception class:**

Create an **exception class** called **RaiseException** that uses the super mechanism to send the error message to the super class. The methods are the following:

* **public RaiseException()**
* **public RaiseExeption(String message)**

**Question 4: (5 points)**

**Validation class:**

Create a validation **class** named **Validator** that contain the following methods:

* **public static boolean isInRange (String value) throws RaiseException**
* **public static boolean isDigit (String value) throws RaiseException**
* **public static boolean isChar (String value) throws RaiseException**
* **public static boolean isDouble (String value) throws RaiseException**

**Question 5: (5 points)**

**Predicate classes:**

Create a predicate class named **SerialNumberPredicate** that implements the Comparator interface and override the compare method as follows:  
 **public int compare (Vehicle vehicle1, Vehicle vehicle2)**

Create a predicate class named **MileageEfficiencyPredicat**e that implements the Comparator interface and override the compare method as follows:  
 **public int compare (Vehicle vehicle1, Vehicle vehicle2)**

**Question 6 : (5 points)**

**Abstract class:**

Create an **abstract** **class named Vehicle** with the following attributes:   
**serial number** of a **Long type**, **make** of a **String type, model** of **String type, date of manufacture** of **DateTime type, trip counter** where trip counter is accessible by the subclasses.

The **Vehicle** class **implements the interface** **IMileAgeEfficiency,** as follows:  
 The method **double getMilesPerEnergyConsumed** **()** remains ***abstract.***

Validate data in the setters. The set method must call the public static method of the Validator class. **If an exception occurs**, **an exception object encapsulatingthe error message will be returned and propagated through the runtime stack from the class to the user**.

**The constructors with parameters must invoke the setter.**

**Question 7: (5 points)**

**Extended classes:**

All electric vehicles and gasoline vehicles **share common features**, and **common behaviors**.

Create two subclasses **ElectricVehicle** and **GasVehicle**, which inherits from **Vehicle** class**.**

* Define a corresponding attribute to each subclass.
* Define the overloaded constructors, the get and set methods
* Override the method that returns the vehicle’s state.
* ***Define the public static services (CRUD operations) to communication with database classes (*ElectricVehicleDB** and **GasVehicleDB**,***)***

**Question 8: (5 points)**

**File Manager class**

Create a class named **FileManager2** that contains the following operations:

* **public void serialize (HashMap<key, value> map) throws IOException,**
* **public HashMap<key, value> deSerialize() throws IOException,**

**Question 9: (10 points)**

**Singleton collection class**

Create a **singleton vehicle fleet** named **SingletonVehicleFleet** that contains the following:

* **private SingletonVehiclesFleet()**
* **public static SingletonVehiclesFleet getSingleInstance()**
* **public HashMap<key, value> getVehicles()**
* **public HashMap<key, value> getElectricVehicles()**
* **public HashMap<key, value> getGasolineVehicles()**
* **public void add (Vehicle object)**
* **public void remove (Vehicle object) `**
* **public void sort (MileageEfficiencyPredicate predict)**
* **public void sort (SerialNumberPredicate predict)**
* **public void sortByMileageEfficiency()**
* **public void sortBySerialNumber()**
* **public void serialize (HashMap<key, value> map) throws IOException,**
* **public HashMap<key, value> deSerialize() throws IOException,**
* **public void search (Long serialNumber)**
* **public void search (String model)**

**Question 10: (5 points)**

**VehicleFleet API and jar file**

Generate **the vehicle fleet API** that documents the user on business classes

Export **the class library into a jar file** named: **vehiclefleet.jar**

**Part II: Managing the Data File – Flat file (10 points)**

**Using Eclipse**:

**Question1: (5 points)**

Managing the fleet of vehicles using the serialized file named **SerVehicleFleet.ser**

* Create a user application named **VehicleFleeFactory**
* Configure the build path to add the external jar file (**vehiclefleet.jar**) to the classpath
* Create a class named VehicleFleet
* Create a user-friendly interface to manage the fleet of vehicles

**Question2 (5 points)**

To effectively manage the fleet of vehicles, you must implement the following operations:

1. Add new vehicles (electric and gasoline vehicles) into the fleet of vehicles
2. Search for a vehicle by serial number.
3. Remove a vehicle from the fleet of vehicles
4. Print the fleet of vehicles
5. Save the fleet of vehicle into a serialized file
6. Read the fleet of vehicles from the serialized file
7. Print the list of gasoline vehicles
8. Print the list of electric vehicles.
9. Test the vehicles: Simulate a trip

* Electric Vehicle: **make trip** and **getMilesPerEnergyConsumed**
* Gasoline vehicle: **make trip** and **getMilesPerEnergyConsumed**

1. Sort the fleet of vehicles ***by mileage efficiency***
2. Print the vehicle having the efficient mileage

**Part III: Managing the database (30%)**

**Using SQL\*Plus**:

Implementing the Fleet of Vehicles Database

**Question1: (10 points)**

* Connect as sys/sysdba
* Create the **vehiclefleetdb** user and grant all the privileges
* Create the **ElecVehicle** and **GasVehicle** **tables**
* Alter the tables and add the **table constraints**
* Alter the tables and add the **domain constraints**
* Insert data into the tables (15 lignes per table)

**Using Eclipse 2021**:

Managing the Fleet of Vehicles Application

**Question 2: (10 points)**

**Database classes:**

**1-** Create the database classes named **ElectricVehicleDB** and **GasolineVehicleDB** that contain the following SQL statements:

* public static int **insert** (Vehicle aVehicle) throws SQLException
* public static int **update** (Vehicle aVehicle) throws SQLException
* public static int **delete** (String serialNumber) throws SQLException
* public static Vehicle **search** (String serialNumber)
* public static ArrayList<Vehicle> **getlist** ()

**2-** Create a Singleton database connection class named ***SingletonConnection*** with the following:

* **private SingletonConnection()**
* **public static SingletonConnection GetInstance()**

**Question3: (10 points)**

* Create a user interface named **VehicleFleeFactory**
* Configure the build path to add the jar file **(vehiclesfleet.jar**) to the classpath
* Create a class named **VehicleFleet**
* Create a friendly user interface to input valid data into the database. If non valid data an exception message must warn the user to input a valid data

To effectively manage the fleet of vehicles, you must implement the following operations:

1. Add a vehicle into the database
2. Search for a vehicle by serial number
3. Update a vehicle
4. Remove a vehicle from the database
5. List the electric vehicles
6. List the gasoline vehicles
7. Test the vehicles: Simulate a trip

* Electric Vehicle: **make trip** and **getMilesPerEnergyConsumed**
* Gasoline vehicle: **make trip** and **getMilesPerEnergyConsumed**

1. Sort a vehicle by mileage efficiency
2. Print the vehicle having the efficient mileage

**Part I – Questions : Q1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Element of competency:** 1. Define the functions of the application. 3. Lay the groundwork for the application. | | | | | |
| **Performance criteria** | **Highly satisfactory** | **Satisfactory** | **Unsatisfactory** | **Highly unsatisfactory** | **Total** |
| **1.1 Determination of the customer’s needs. 1.2 Proper use of data collection techniques. 1.3 Complete analysis of the characteristics of activities related to the application to be developed. 1.4 Deduction and justification of the application’s functions. 1.5 Global representation of the functions 3.1 Creation of a realistic work schedule. 3.2 Appropriate use of planning methods and tools. 3.3 Acquisition of the proper environment in which to develop the application** | 10-9 | 8 - 7 | 6-5 | 4-0 | **/10** |

**Part I – Questions : Q2, Q3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Element of competency: 6. Develop programs** | | | | | |
| **Performance criteria** | **Highly satisfactory** | **Satisfactory** | **Unsatisfactory** | **Highly unsatisfactory** | **Total** |
| 6.1 Appropriate coding of functions and procedures.  6.2 Proper use of function libraries and procedures specific to the database management system  6.3 Appropriate programming of modules.  6.4 Optimal use of the programming language | 2.5-2.25 | 2-1.75 | 1.5-1.25 | 1-0 | **/2.5** |

**Part I – Questions: Q4. Q5, Q6, Q7, Q8, Q10, Q11**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Element of competency - 6. Develop programs - 7. Produce documentation for the application.** | | | | | |
| **Performance criteria** | **Highly satisfactory** | **Satisfactory** | **Unsatisfactory** | **Highly unsatisfactory** | **Total** |
| 6.1 Appropriate coding of functions and procedures. 6.2 Proper use of function libraries and procedures specific to the database management system 6.3 Appropriate programming of modules.  6.4 Optimal use of the programming language  **7.2 Appropriate creation of an on-line help function.**  **7.3 Production of a clear and comprehensive user manual for the application.** | 5-4.5 | 4-3.5 | 3-2.5 | 2-0 | **/5** |

**Part I – Questions : Q9**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Element of competency:** 1. Define the functions of the application. | | | | | |
| **Performance criteria** | **Highly satisfactory** | **Satisfactory** | **Unsatisfactory** | **Highly unsatisfactory** | **Total** |
| **1.1 Determination of the customer’s needs. 1.2 Proper use of data collection techniques.** | 10-9 | 8 - 7 | 6-5 | 4-0 | **/10** |

**Part II - Questions : Q1, Q2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Element of competency: 5. Produce a user interface using a prototype.** | | | | | |
| **Performance criteria** | **Highly satisfactory** | **Satisfactory** | **Unsatisfactory** | **Highly unsatisfactory** | **Total** |
| 5.2 inputs and outputs consistent with data type and customer needs and requirements.  5.3 Complete validation of interface performance. 5.4 Clear presentation | 10-9 | 8 - 7 | 6-5 | 4-0 | **/10** |

**Part III – Questions : Q1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Element of competency:** Design and develop an application in a database environment - 017B | | | | | |
| **Performance criteria** | **Highly satisfactory** | **Satisfactory** | **Unsatisfactory** | **Highly unsatisfactory** | **Total** |
| 4.1 Appropriate modelling of the data.  4.2 Creation of an appropriate data dictionary.  4.3 Proper modelling of processes. 4.5 Proper creation of tables. | 10-9 | 8 - 7 | 6-5 | 4-0 | **/10** |

**Part III – Questions : Q2, Q3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Element of competency: 5**. Produce a user interface using a prototype. | | | | | |
| **Performance criteria** | **Highly satisfactory** | **Satisfactory** | **Unsatisfactory** | **Highly unsatisfactory** | **Total** |
| 5.2 Creation of menus inputs and outputs consistent with data type and customer needs and requirements.  5.3 Complete valid of menus | 20-18 | 17 - 13 | 12-10 | 9-0 | **/20** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CORRECTION GRID FOR LANGUAGE**   |  |  |  |  | | --- | --- | --- | --- | | Clear Communication | Clear Communication, **most of the time** | Vague Communication | Unclear Communication | | - 0 | - 0,5 | - 1,5 | - 2 | | (Word Choice)  Use of precise and rich vocabulary | (Word Choice)  Use of precise vocabulary | (Word Choice)  Use of imprecise vocabulary | (Word Choice)  Use of inappropriate vocabulary | | - 0 | - 0,5 | - 1,5 | - 2 | | (Format/Type of work)  Respect of norms | (Format/Type of work)  Respect of **most of the** norms | (Format/Type of work)  Non-respect of the norms | (Format/Type of work)  Inappropriate in relation to the required norms | | - 0 | - 0,5 | - 1,5 | - 2 | | (Linguistic Code)  (≤2 mistakes / page) | (Linguistic Code)  (3-7 mistakes/page) | (Linguistic Code)  (8-10 mistakes/ page) | (Linguistic Code)  (>10 mistakes/  page) | | - 0 | - 0,5 - 2.5 | - 2.5 - 3.5 | - 4 | |