

*<Nom de l’alumne>*

CAS PRÀCTIC: PR02

0485-Programació

0487-entorns

1708-sostenibilitat

entrega - ORDinària

Nom: Alex Jiménez Quiñonero

Curs: M03-UF4

Data: 22/03/2024

Introducció

Aquest exercici es divideix en tres fases incrementals, on s'anirà construint i millorant el sistema de manera progressiva:

• **Fase 1**: Creació de l’estructura del projecte – Definició de les classes i implementació del codi seguint les especificacions donades.

• **Fase 2**: Disseny i documentació UML – Elaboració del diagrama de classes, casos d’ús, requisits funcionals i diagrames d’activitat per representar visualment el sistema.

• **Fase 3**: Implementació i demostració – Desenvolupament final, afegint millores i realitzant una demo funcional del projecte.

El projecte està integrat amb testos unitaris i validació automàtica mitjançant **GitHub Actions**. Cada cop que es faci un push a la branca main, s'executaran aquests testos per garantir el bon funcionament del codi.

Aquesta pràctica afecta als següents RAs:

* 0485. Programació (RA2, RA4, RA6, RA7)
* 0487. Entorns de desenvolupament (RA4, RA5, RA6)
* 1708. Sostenibilitat aplicada al sistema productiu (tots els RAs)

Explicació prèvia

Benvingut al **projecte JavaCar**, **un sistema de gestió de lloguer de vehicles** desenvolupat en Java seguint els principis de la **Programació Orientada a Objectes (POO).** Aquest sistema permet administrar diferents tipus de vehicles disponibles per al lloguer, calcular els preus i gestionar els ingressos generats.

Com a programador júnior, t’incorpores a l’equip de desenvolupament en una fase a on l’arquitectura del projecte ja està definida i s’han establert les especificacions principals.

La teva tasca serà seguir l’estructura del projecte, implementar noves funcionalitats i assegurar-te que el codi compleix els requisits especificats.



PART A: Estructura del projecte

Segons les notes que t’han deixat, saps que el projecte està organitzat en les següents classes i mòduls:

1. Interfície Llogable (hecho)

* calcularPreu(int dies): double

1. Classe Abstracta Vehicle (implementa Llogable) (hecho)
   * Atributs: matricula, marca, model, preuBase, motor, rodes, etiquetaAmbiental **(Veure NOTA SOSTENIBILITAT)**
   * Constructor amb tots els atributs.
   * Getters per atributs
2. Classes Concretes de Vehicles (hecho)
   * Cotxe (nombrePlaces)
     1. Ordre d’atributs en el constructor:

Cotxe(matricula, marca, model, preuBase, nombrePlaces, motor, rodes)

* + Moto (cilindrada):
    1. Ordre d’atributs en el constructor:

Moto(matricula, marca, model, preuBase, cilindrada, motor, rodes)

* + 1. CalculPreu tenir en compte que si la cilindrada > 500 llavors afegim 5€ al preuBase
  + Furgoneta (capacitatCarga)
    1. Ordre d’atributs en el constructor:

Furgoneta(matricula, marca, model, preuBase, capacitatCarga, motor, rodes)

* + 1. CalculPreu tenir en compte que si la capacitatCarga > 1000 llavors afegim 10€ al preuBase

1. Classes Motor i Roda (hecho)
   * Motor (tipus, potencia)
   * Roda (marca, diametre)
2. Classe Estàtica GestorLloguers (hecho)
   * calcularIngressosTotals(List<Vehicle>, int dies): double
   * filtrarPerPreu(List<Vehicle>, double preuMax): List<Vehicle>

****NOTA SOSTENIBILITAT**:

Caldrà afegir els atributs i mètodes necessaris a les clases anteriors per tal de poder gestionar correctament l’etiqueta mediAmbiental que han de tenir ara tots els vehicles.

L’atribut de l’etiquetaAmbiental s’haurà de recalcular al constructor segons la normativa de distintius ambientals de la DGT.

[*https://ajuntament.barcelona.cat/qualitataire/ca/zona-de-baixes-emissions/el-distintiu-ambiental-de-la-dgt*](https://ajuntament.barcelona.cat/qualitataire/ca/zona-de-baixes-emissions/el-distintiu-ambiental-de-la-dgt)

Ens interessa poder guardar la informació necessària per al càlcul, calcular l’etiqueta que pertoca al vehicle i la possibilitat de recalcular en cas de canviar les característiques d’un vehicle.

PART B: Disseny del UML

En aquesta fase caldrà dissenyar la documentació UML següent:

• Diagrama de clases (hecho)

• Diagrama de casos d’ús (mínim 3 casos d’ús directes i 3 d’indirectes)

• Taula de requisits funcionals (mínim sobre 3 casos d’ús)

• Diagrames d’activitat (mínim sobre 3 casos d’ús)

Cada alumne haurà de proposar funcionalitats addicionals interessants. Es valorarà positivament la creativitat i aportació de funcionalitats extres com historial de lloguers, descomptes, nous vehicles o integració amb fitxers.

1. ***Diagrama de clases UML***

Realizado en <https://www.planttext.com>

**A screenshot of a computer

AI-generated content may be incorrect.**

Codigo a mano:

@startuml

interface Llogable {

+ calcularPreu(int): double

}

abstract class Vehicle implements Llogable {

- matricula: String

- marca: String

- model: String

- preuBase: double

- motor: Motor

- rodes: Roda

- etiquetaAmbiental: String

+ Vehicle(...)

+ getMatricula(): String

+ getMarca(): String

+ getModel(): String

+ getPreuBase(): double

+ getMotor(): Motor

+ getRodes(): Roda

+ getEtiquetaAmbiental(): String

+ calcularPreu(int): double

}

class Cotxe extends Vehicle {

- nombrePlaces: int

+ Cotxe(...)

+ calcularPreu(int): double

}

class Moto extends Vehicle {

- cilindrada: int

+ Moto(...)

+ calcularPreu(int): double

}

class Furgoneta extends Vehicle {

- capacitatCarga: int

+ Furgoneta(...)

+ calcularPreu(int): double

}

class Motor {

- tipus: String

- potencia: int

+ Motor(...)

+ getTipus(): String

+ getPotencia(): int

}

class Roda {

- marca: String

- diametre: int

+ Roda(...)

+ getMarca(): String

+ getDiametre(): int

}

class GestorLloguers {

+ calcularIngressosTotals(List<Vehicle>, int): double

+ filtrarPerPreu(List<Vehicle>, double): List<Vehicle>

}

Vehicle "1" \*-- "1" Motor

Vehicle "1" \*-- "4" Roda

GestorLloguers --> Vehicle

@enduml

1. **Diagrama de casos d’ús (mínim 3 casos d’ús directes i 3 d’indirectes)**
2. **Taula de requisits funcionals (mínim sobre 3 casos d’ús)**
3. **Diagrames d’activitat (mínim sobre 3 casos d’ús)**

PART C: Implementació i Demostració

En aquesta última fase cal completar la implementació, assegurar que totes les funcionalitats estan operatives i preparar una demostració funcional.

* Finalitzar i revisar totes les classes. hecho
* Validar tots els testos unitaris. hecho
* Preparar una demo funcional explicant clarament les funcionalitats implementades. hecho

LLIURAMENT I AVALUACIÓ

El lliurament ha de contenir:

* Totes les classes implementades. I l’aplicació funcional hecho
* Documentació UML completa.
* Una presentació amb la guia d'ús del programa i funcionalitats. hecho

Els criteris per a l’avaluació són:

* Correcció i completitud de la implementació.
* Qualitat i organització del codi.
* Resultats correctes en els testos unitaris.
* Implementació coherent i encertada del distintiu ambiental de sostenibilitat
* Originalitat i aportació de millores extres.
* Qualitat dels dissenys implementats
* Qualitat de la demostració i claredat en l’explicació.
* **EXTRA**: Aportació de proves sobre l’aplicació d’SCRUM en la realització de la pràctica
* **PENALITZACIÓ**: Es penalitzarà no utilizar correctament el Sistema de Control de Versions ubicat en el Github del projecte.