**ESCOLA POLITÈCNICA SUPERIOR**

**UNIVERSITAT DE GIRONA**



**PROJECTE: Tipus abstractes i mòduls funcionals (v2.0)**

Elaborat per:

**David Martínez, u1939690**

**Roger Barnés, u1939667**

**Matèria:** Projecte de Programació

**Grup:** Pràctiques L1, GEINF

**Versió:** 2.0

**Professor:** Dr. Francesc Castro & Dr. Miquel Bofill

Montilivi, Girona, 30 de març de 2017

1. **Tipus Abstractes de dades**

**Tipus Client**

Descripció general: Conté informació d’un client i les seves preferències.

*Pre: --*

*Post: Es crea un client amb nom “nom” i preferències “prefs”*

Client(String nom, Collection<String> prefs)

*Pre: --*

*Post: Retorna el nom del Client*

String obtenirNom()

*Pre: --*

*Post: Retorna cert si la característica “car” es troba entre les preferències del Client*

Boolean tePreferencia(String car)

**Tipus GrupClients**

Descripció general: Conté un grup de clients, amb una categoria d’allotjament preferent, uns llocs que s’han de visitar prefixats, data i hora que s’iniciarà el seu viatge, lloc d’origen i de destí, i duració màxima.

*Pre: --*

*Post: Es crea un conjunt de clients amb tants clients com té “clients”, categoria desitjada i punts d’interès a visitar*

GrupClients(Collection<Client> clients, Integer catDesit, Collection<PuntInteres> pI, Lloc origen, Lloc desti, Data inici, Double duracioMax)

*Pre: --*

*Post: Retorna un enter que representa el nombre de clients del conjunt que tenen la preferència “pref” entre les seves preferències personals*

Integer obtenirSatisfaccioPreferencia(String pref)

*Pre: --*

*Post: Retorna cert si l’allotjament “hotel” és de la categoria que desitja el conjunt de clients*

Boolean categoriaDesitjada(Allotjament hotel)

*Pre: --*

*Post: Retorna el lloc d’origen del viatge del grup de clients*

Lloc obtenirOrigen()

*Pre: --*

*Post: Retorna el lloc de destí del viatge del grup de clients*

Lloc obtenirDesti()

*Pre: --*

*Post: Retorna la data (amb hora inclosa) de sortida del grup de clients*

Data obtenirInici()

*Pre: --*

*Post: Retorna un iterador als punts d’interès prefixats que s’han de visitar sí o sí*

Iterator<PuntInteres> obtenirInteressos()

**Tipus HoraDia**

Descripció general: Conté informació d’un instant (hh24:mm) d’un dia de la setmana.

*Pre: dia ha d’existir dins de «dl,dm,dc,dj,dv,ds,dg»,*

*0 <= hora <= 23. 0 <= minuts <= 59*

*Post: Crea una Hora d’un dia a partir del dia de la setmana “dia” i les hores i minuts en format 24h*

HoraDia(String dia, Integer hora, Integer minuts)

**Tipus FranjaHoraria**

Descripció general: Conté una franja horària compresa entre dos HoraDia determinades.

*Pre: --*

*Post: Crea una franja horària a partir d’un HoraDia inici i final*

FranjaHoraria(HoraDia inici, HoraDia final)

*Pre: --*

*Post: Retorna cert si l’HoraDia “instant” es troba dins de l’horari de la franja*

Boolean pertanyFranja(HoraDia instant)

**Tipus PuntInteres**

Descripció general: Representa un punt on els clients tenen interès, bé sigui per visitar o bé per allotjar-se.

*Pre: --*

*Post: Es crea un punt d’interès de nom “nom” amb les activitats que ofereix “acts” i el preu “cost”*

PuntInteres(String nom, Collection<String> acts, Double cost)

*Pre: --*

*Post: Retorna cert si el punt d’interès satisfà la preferència “pref”*

Boolean satisfaPreferencia(String pref)

*Pre: --*

*Post: Retorna el nom del punt d’interès*

String obtenirNom()

*Pre: --*

*Post: Retorna el codi del punt d’interès*

String obtenirCodi()

*Pre: --*

*Post: Retorna el lloc on està vinculat el punt d’interès*

Lloc obtenirLloc()

**Tipus PuntVisitable refina PuntInteres**

Descripció general: Representa un punt visitable, amb les seves franges horàries i temps de visita.

*Pre: –*

*Post: Es crea un punt visitable a partir de la informació del punt d’interès i el temps mitjà de visita*

PuntVisitable (String nom, Collection<String> acts, Double cost, Integer tempsVisita)

*Pre: --*

*Post: Retorna el cost del punt visitable. Si és gratis retorna 0*

Double obtenirPreu()

*Pre: –*

*Post: S’ha afegit la franja horària “fh” al punt visitable*

void afegirFranja(FranjaHoraria fh)

*Pre: –*

*Post: Retorna el temps mitjà de visita*

Integer obtenirTempsVisita()

*Pre: –*

*Post: Retorna cert si el punt visitable està obert en un DiaHora determinat*

Boolean estaObert(DiaHora inst)

**Tipus Allotjament refina PuntInteres**

Descripció general: Representa un allotjament, amb la seva categoria.

*Pre: 1<= categoria <= 5*

*Post: Es crea un allotjament a partir de les dades del punt d’interès i la categoria del allotjament*

Allotjament(String nom, Collection<String> acts, Double cost, Integer categoria)

*Pre: --*

*Post: Retorna la categoria de l’allotjament*

Integer obtenirCat()

*Pre: --*

*Post: Retorna el cost per allotjar-se en una habitació doble. Si és gratis retorna 0*

Double obtenirPreu()

**Tipus Coordenades**

Descripció general: Representa una localització amb latitud i longitud, juntament a la zona horària on està compromesa.

*Pre: -12 <= zH <= 14*

*Post: Es crea unes coordenades amb latitud, longitud i zona horària UTC (Representada amb un nombre real)*

Coordenades(String latitud, String longitud, Double zH)

*Pre: --*

*Post: Retorna la zona horària de les coordenades*

Double obtenirZonaHoraria()

**Tipus Lloc**

Descripció general: Representa un lloc amb les seves coordenades, generalment una ciutat, que conté punts d’interès\*.

*Pre: --*

*Post: Es crea un lloc de nom “nom” i coordenades “coor”*

Lloc(String nom, Coordenades coor)

*Pre: --*

*Post: Retorna el nom del lloc*

String obtenirNom()

*Pre: --*

*Post: Retorna les coordenades del Lloc*

Coordenades obtenirCoordenades()

*Pre: --*

*Post: Afegeix la estació “est” a la llista d’estacions del lloc*

void afegirEstacio(Estacio est)

*Pre: --*

*Post: Afegeix el punt d’interès “pI” als punts d’interès del lloc*

void afegirPuntInteres(PuntInteres pI)

*Pre: --*

*Post: Retorna l’estació associada al lloc*

Estacio obtenirEstacio()

**Tipus MitjaTransport**

Descripció general: Conté informació del mitjà de transport.

*Pre: origen i destí han de ser llocs o punts d’interès*

*Post: Crea un mitjà de transport amb origen, destí, preu, durada i descriptor*

MitjaTransport(String descriptor, Object o, Object d, Double preu, Integer durada)

*Pre: --*

*Post: Retorna el Lloc/Punt d’interès d’origen del transport*

Object getOrigen()

*Pre: --*

*Post: Retorna el Lloc/Punt d’interès de destí del transport*

Object getDesti()

*Pre: --*

*Post: Retorna el preu del transport*

Double getPreu()

*Pre: –*

*Post: Retorna la durada del transport*

Double getDurada()

*Pre: –*

*Post: Retorna el descriptor del transport*

String getDescriptor()

**Tipus MTDirecte refina MitjaTransport**

Descripció general: Conté informació del mitjà de transport directe.

*Pre: –*

*Post: Crea un MTDirecte amb els paràmetres de MitjaTransport + Boolea que indica si és urbà o interurbà*

MTDirecte(<MT>, Boolean urba)

*Pre: –*

*Post: Retorna cert si el MTDirecte és urbà*

Boolean esUrba()

**Tipus MTIndirecte refina MitjaTransport**

Descripció general: Conté informació del mitjà de transport indirecte.

*Pre: –*

*Post: Crea un MTIndirecte amb els paràmetres de MitjaTransport + Estació vinculada*

MTIndirecte(<MT>, Estacio estVinculada)

*Pre: –*

*Post: Retorna la estació a la que està vinculada el mitjà de transport indirecte*

Estacio getEstacio()

**Tipus Estacio**

Descripció general: Representa un HUB d’un mitjà de transport indirecte en una ciutat.

*Pre: --*

*Post: Crea una estació al lloc “ciutat”, amb un temps d’origen i destí de desplaçament en minuts*

Estacio(Lloc ciutat, Double tOrigen, Double tDesti)

**Tipus Trajecte**

Descripció general: Conté informació d’un desplaçament entre Llocs/Punts d’interès, juntament amb les hores de sortida i arribada.

*Pre: --*

*Post: Es crea un Trajecte amb el transport i les hores de sortida i arribada*

Trajecte(MitjaTransport mT, HoraDia sortida, HoraDia arribada)

Pre: --

Post: Retorna el Lloc/Punt d’interès d’origen del Trajecte

Object getOrigen()

*Pre: --*

*Post: Retorna el Lloc/Punt d’interès de destí del Trajecte*

Object getDesti()

*Pre: --*

*Post: Retorna la durada del Trajecte en minuts*

Double getDurada()

*Pre: --*

*Post: Retorna el preu del Trajecte*

Double getPreu()

*Pre: --*

*Post: Retorna l’hora de Sortida del Trajecte*

HoraDia getSortida()

*Pre: --*

*Post: Retorna l’hora d’Arribada del Trajecte*

HoraDia getArribada()

**Tipus Ruta**

Descripció general: Conté el conjunt de subtrajectes dels viatges, juntament amb els PuntInteres i Llocs on s’ha passat del mapa. També conté un trajecte absolut de Ruta.

*Pre: --*

*Post: Crea una Ruta inicial buida*

Ruta()

*Pre: --*

*Post: Retorna cert si el Trajecte absolut de la ruta té* *“origen” com origen i “desti” com destí, i també opcionalment passa per tots els “punts”. És a dir, si la Ruta està completada.*

Boolean esCompleta(PuntInteres origen, PuntInteres desti, Collection<PuntInteres> punts)

*Pre: --*

*Post: Afegeix un subtrajecte a la Ruta*

void afegeixSubtrajecte(Trajecte traj)

*Pre: --*

*Post: Retorna cert si la Ruta es buida, fals altrament*

Boolean isEmpty()

**Tipus Mapa**

Descripció general: Mapa de Llocs amb les seves estacions (amb els seus TrajecteExtern) i Punts d’interès amb els seus TrajecteInterns, on els punts d’interès estan interconnectats amb els llocs.

*Pre: --*

*Post: Crea un mapa buit*

Mapa()

*Pre: --*

*Post: Afegeix un lloc al mapa, i també (si en té), els seus punts d’interès i les seves estacions.*

void afegeixLloc(Lloc ll)

*Pre: Lloc on està l’estació ha d’existir*

*Post: Afegeix una estació al mapa*

void afegeixEstacio(Estacio est)

*Pre: Lloc on està el punt d’interès ha d’existir*

*Post: Afegeix un punt d’interès al mapa*

void afegeixPuntInteres(PuntInteres pI)

*Pre: --*

*Post: Afegeix un desplaçament al mapa entre dos llocs o entre dos punts d’interès a partir d’un trajecte entre ells*

void afegeixTrajecte(Trajecte traj)

*Pre: --*

*Post: Retorna cert si existeix el punt d’interès*

Boolean existeixPuntInteres(PuntInteres pI)

*Pre: --*

*Post: Retorna el nombre de punts d’interès del mapa*

Integer nPuntsInteres()

*Pre: tipus == “temps” || tipus == “dist” || tipus == “cost”*

*Post: Retorna un Map amb els punts d’interès des d’on es pot anar a partir de pI i el seu Trajecte (El de mínim temps, mínima distància o mínim cost depenent de “tipus”)*

Map<PuntInteres,Trajecte> getDesplsMin(PuntInteres pI, String tipus)

**Tipus CalculRutaMapa**

Descripció general: Mòdul funcional que conté algorismes per el càlcul de rutes o circuits a partir del mapa.

*Pre: --*

*Post: Calcula una Ruta mitjançant backtraking*

Ruta calcularRutaBack(grupClients clients, Mapa mon)

*Pre: --*

*Post: Calcula una Ruta mitjançant un algorisme voraç*

Ruta calcularRutaGreedy(grupClients clients, Mapa mon)

*Pre: --*

*Post: Concatena les rutes a i b*

Ruta concatenarRutes(Ruta a, Ruta b)

**Tipus EntradaSortida**

Descripció general: Mòdul funcional que conté mètodes per a demanar/mostrar dades per pantalla.

*Pre: --*

*Post: Demana la ruta del fitxer a carregar dades i el retorna en lectura preparat per llegir*

Fitxer entrarNomFitxer()

*Pre: --*

*Post: Demana les dades per a poder crear un GrupClients a partir dels Clients de l’agència*

GrupClients crearGrup(Collection<Client> clients)

*Pre: --*

*Post: Mostra la Ruta “resultat” per la sortida estàndard*

void mostrarRuta(Ruta resultat)

*\*Aclariment: En el cas de tenir un punt d’interès no associat a cap lloc (Lloc Primari sense Lloc Secundari), com també se n’ha de tenir constància de les seves coordenades i pot usar transport indirecte juntament amb altres característiques dels llocs, considerarem que es troba dins d’un lloc artificial per a tenir-hi constància (Amb el mateix nom).*

1. **Mòduls funcionals**

Consultar Imatge mòduls.png