# PROGRAMACIÓ AVANÇADA
# PRÀCTICA 1. TARDOR 2015

---

**Objectiu 1:** Repassar els conceptes típics de la P.O.O.: classes, atributs, mètodes, constructors, herència, polimorfisme, redefinició (sobreescriure) de mètodes.
**Objectiu 2:** Col·leccions de Java. API de Java. Taules.
**Objectiu 3:** Primeres implementacions usant variable dinàmica.
**Durada:** Una sessió.
**Lliurament:** Llistat imprès dels fonts i penjar el projecte al Moodle.
**Data Lliurament:** El dia previ a la sessió de la pràctica 2.

## Enunciat

Una companyia d'assegurances vol informatitzar-se, encarregant un aplicatiu per gestionar tota la informació referent a les assegurances que ofereix, la cartera de clients i els agents d'assegurances que treballen per la companyia.

La companyia necessita tenir les següents dades emmagatzemades:

- Dels agents: nom, DNI, adreça, població, telèfon, anys d'antiguitat a l'empresa, número compte corrent i sou.
- Dels clients: nom, DNI, adreça, població, telèfon, número compte corrent.
- De les assegurances: número de pòlissa, client tomador, data d'emissió, import, tipus d'assegurança i agent tramitador. En funció del tipus d'assegurança calen d'altres dades.

Aquesta companyia ofereix 3 tipus d'assegurances:

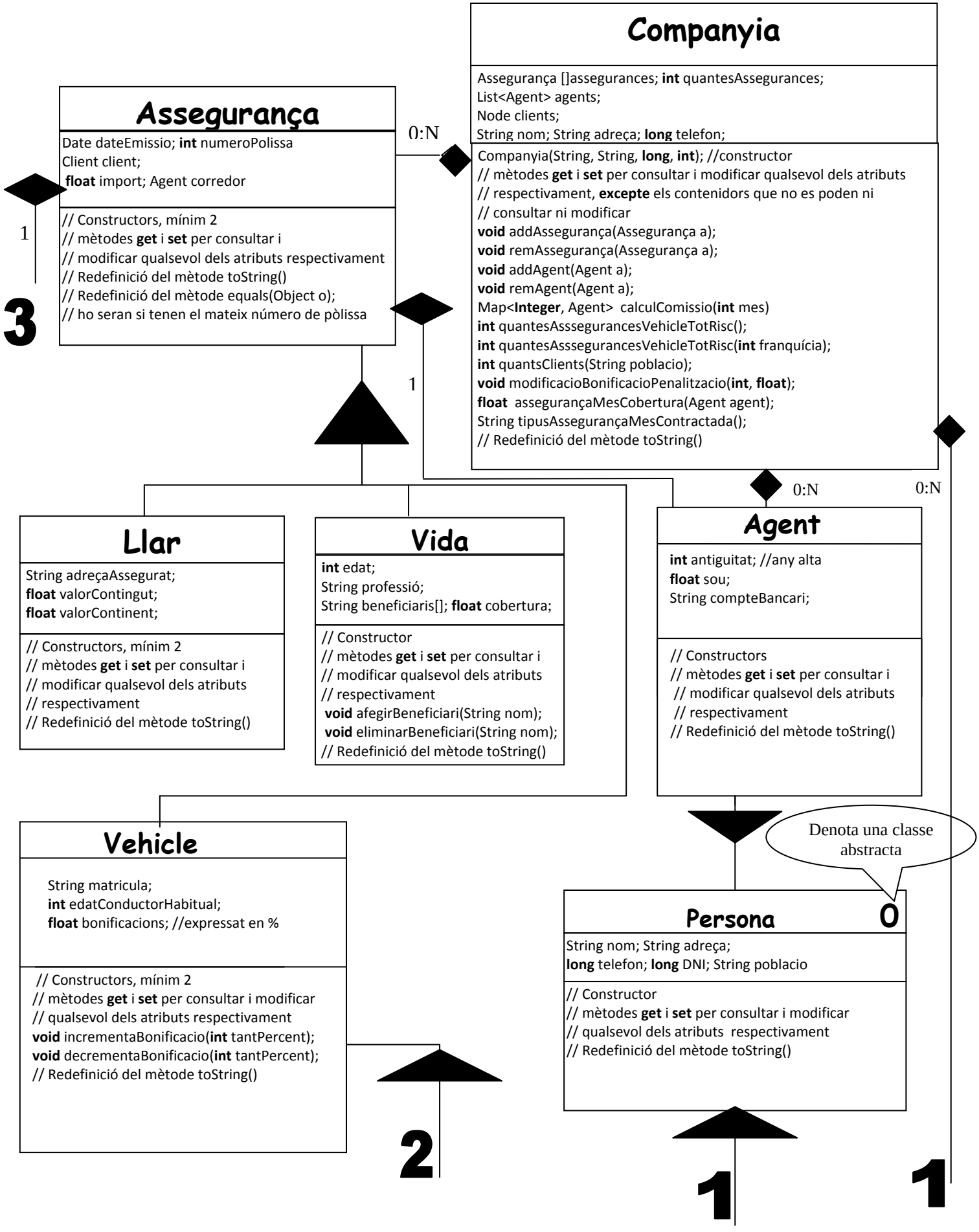
- → Assegurança de la Llar
- → Assegurança de Vida
- → Assegurança de Vehicle

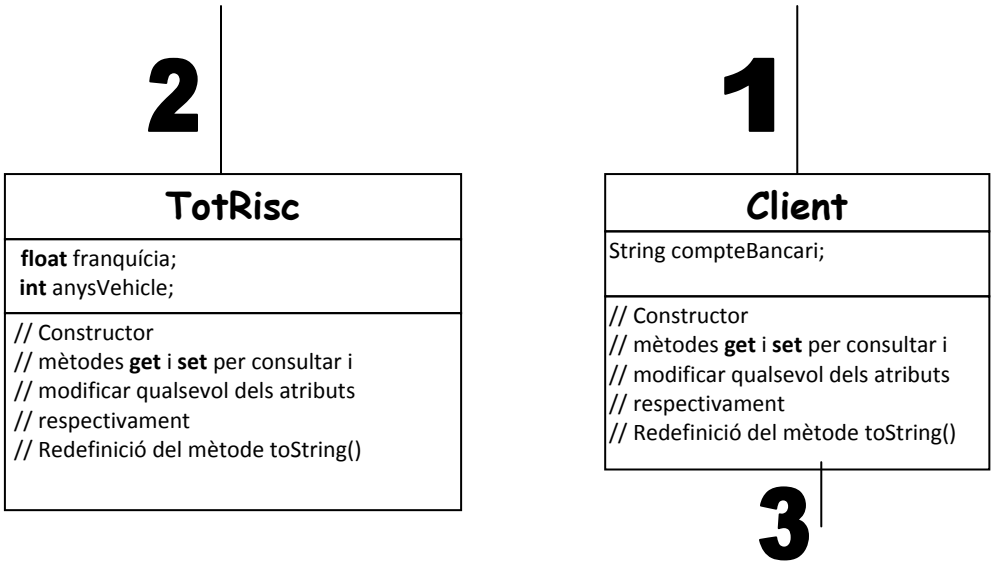
A més d'emmagatzemar dades s'han d'oferir mètodes per poder realitzar les altes, baixes, modificacions i consultes que requereix l'empresa.

Així doncs, després de la fase de presa de requeriments i anàlisi del problema, el disseny de classes que s'ha de codificar queda reflectit en el següent diagrama de classes, on la simbologia:

Indica una relació d'herència entre classes

Indica una relació de continença entre dues classes, la continguda i el contenidor. Aquest tipus de relació s'ha d'implementar com una agregació, és a dir la classe contenidora té un o més atributs d'objectes del tipus classe continguda


Companyia
Assegurança []assegurances; int quantesAssegurances;
List<Agent> agents;
Node clients;
String nom; String adreça; long telefon;
Companyia(String, String, long, int); //constructor
// mètodes get i set per consultar i modificar qualsevol dels atributs
// respectivament, excepte els contenidors que no es poden ni
// consultar ni modificar
void addAssegurança(Assegurança a);
void remAssegurança(Assegurança a);
void addAgent(Agent a);
void remAgent(Agent a);
Map<Integer, Agent> calculComissio(int mes)
int quantesAsssegurancesVehicleTotRisc();
int quantesAsssegurancesVehicleTotRisc(int franquícia);
int quantsClients(String poblacio);
void modificacioBonificacioPenalitzacio(int, float);
float asseguraçaMesCobertura(Agent agent);
String tipusAsseguraçaMesContractada();
// Redefinició del mètode toString()
Assegurança
Date dateEmissio; int numeroPolissa
Client client;
float import; Agent corredor
// Constructors, mínim 2
// mètodes get i set per consultar i
// modificar qualsevol dels atributs respectivament
// Redefinició del mètode toString()
// Redefinició del mètode equals(Object o);
// ho seran si tenen el mateix número de pòlissa
0:N
1
3
1
0:N
0:N
Llar
String adreçaAssegurat;
float valorContingut;
float valorContinent;
// Constructors, mínim 2
// mètodes get i set per consultar i
// modificar qualsevol dels atributs
// respectivament
// Redefinició del mètode toString()
Vida
int edat;
String professió;
String beneficiaris[]; float cobertura;
// Constructor
// mètodes get i set per consultar i
// modificar qualsevol dels atributs
// respectivament
void afegirBeneficiari(String nom);
void eliminarBeneficiari(String nom);
// Redefinició del mètode toString()
Agent
int antiguitat; //any alta
float sou;
String compteBancari;
// Constructors
// mètodes get i set per consultar i
// modificar qualsevol dels atributs
// respectivament
// Redefinició del mètode toString()
Vehicle
String matricula;
int edatConductorHabitual;
float bonificacions; //expressat en %
// Constructors, mínim 2
// mètodes get i set per consultar i modificar
// qualsevol dels atributs respectivament
void incrementaBonificacio(int tantPercent);
void decrementaBonificacio(int tantPercent);
// Redefinició del mètode toString()
Denota una classe abstracta
Persona
0
String nom; String adreça;
long telefon; long DNI; String poblacio
// Constructor
// mètodes get i set per consultar i modificar
// qualsevol dels atributs respectivament
// Redefinició del mètode toString()
2
1
1




La classe Companyia té tres magatzems:

- 1.- **assegurances**: és un taula d'una dimensió on s'emmagatzemen **seqüencialment** totes les assegurances que la companyia té contractades. Haureu d'implementar vosaltres tot el manteniment d'aquest vector per donar implementació a tots els mètodes demanats. L'atribut **quantesAssegurances** emmagatzema el número d'assegurances contractades, és a dir la dimensió real de la taula.
- 2.- **agents**: és un objecte de la col·lecció parametritzada List<Agent>. En aquest cas féu ús dels mètodes que us dona la llibreria per implementar el manteniment sol·licitat d'aquest magatzem.
- 3.- **clients**: seqüència enllaçada de nodes, aquesta no té node capçalera i és lineal. La classe node s'ha de declarar privada dins de la classe Companyia. La classe Node té 2 atributs, l'un per emmagatzemar l'objecte Client i altre per referenciar a d'altre Node i crear així la seqüència enllaçada. Ha de tenir mètodes constructors.

Codifiqueu el disseny de classes a Java tenint en compte:

- Totes les classes han de tenir mètode(s) **constructor(s), un obligatòriament** amb la funcionalitat afegida d'inicialització de **tots** els atributs de l'objecte construït amb les dades especificades per l'usuari de l'aplicació. Si en la classe se us demana més d'un constructor heu de determinar el(s) més adient(s) en funció dels mètodes que oferta la classe per consultar i modificar atributs.
- En el cas de la classe *Companyia* el constructor a part de crear la l'objecte amb nom, adreça i telèfon amb valors indicats en els tres primers paràmetres, ha de crear els magatzems que de moment romandran buits. El darrer paràmetre del constructor indica el nombre màxim d'assegurances que la companyia pot tenir contractades, valor a usar en la creació del contenidor de les assegurances.
- Els atributs de totes les classes **han** de ser **privats**. Tots els mètodes demanats han de ser **públics.**
- Implementeu tots els mètodes de totes les classes.

→ els mètodes anomenats set*** permeten **modificar** l'atribut de nom ***.

→ els mètodes de nom get*** permeten **consultar** l'atribut de nom ***.

➔ **Totes** les classes han de redefinir el mètode toString(), una representació en format cadena de **tots** els atributs (també els heretats) de la classe.

➔ Classe *Companyia*. Mètodes:

**void** addAssegurança(Assegurança a);

afegeix al magatzem corresponent la nova contractació. Si la persona que fa la contractació no era client de la companyia també s'ha d'afegir al corresponent magatzem de clients.

**void** remAssegurança(Assegurança a);

es dona de baixa l'assegurança, cal eliminar-la del magatzem. Recordeu que totes les posicions plenes han d'estar en posicions consecutives. Si l'assegurança que es dona de baixa era l'única que el client tenia contractada aquest deixa de ser client, per tant s'haurà de donar de baixa del magatzem de clients.

**void** addAgent(Agent a);

afegeix al magatzem corresponent el nou treballador.

**void** remAgent(Agent a);

es dona de baixa l'agent, deixa de treballar per l'empresa, cal eliminar-lo del magatzem.

Map<**Integer**, Agent> calculComissio(**int** mes)

Aquest mètode crea, omplena i retorna un Map que emmagatzema i relaciona cada pòlissa contractada durant el mes indicat en el paràmetre amb l'agent que ha fet l'operació, a partir d'aquestes dades l'aplicatiu de nòmines de l'empresa en calcularà la comissió mensual de cadascun dels agents.

La clau ve determinada pel número de pòlissa de l'assegurança, que és identificador, i el valor és l'agent que ha realitzat la operació.

**int** quantesAsssegurancesVehicleTotRisc();

ha de comptabilitzar el nombre de contractacions a tot risc sense franquícia que té la companyia.

**int** quantesAsssegurancesVehicleTotRisc(**int** franquícia);

Aquest mètode és una sobrecarrega de l'anterior, observeu que té el mateix nom de mètode, en aquest cas el mètode només ha de comptabilitzar el nombre de contractacions a tot risc amb una franquícia superior a la indicada en el paràmetre del mètode.

**int** quantsClients(String poblacio);

comptabilitza i retorna el nombre total de clients de la companyia que són de la població indicada en el paràmetre, si un mateix client té més d'una assegurança només s'ha de comptabilitzar una única vegada.

**void** modificacioBonificacioPenalitzacio(**int**, **float**);

aquest mètode només és aplicable si l'assegurança és de tipus Vehicle, el mètode modifica la bonificació/penalització de l'assegurança identificada amb el número de pòlissa indicat en el primer paràmetre. Modifica la bonificació de l'assegurança amb el valor indicat en el paràmetre, si té un valor positiu la incrementa, en cas contrari la decrementa.

**float** assegurançaMesCobertura(Agent agent);

troba i retorna la cobertura màxima de les assegurances de Vida tramitades per l'agent especificat en el paràmetre.

String tipusAssegurançaMesContractada();

determina quin tipus d'assegurança és la més contractada, Vehicle, Llar ó Vida, en retorna el string corresponent.

- **Tots els mètodes que poden donar situacions "anormals**" ho han de controlar mitjançant l'ús d'**excepcions**. Per exemple: el mètode de donar de baixa una assegurança, no pot fer-ho d'una assegurança inexistent.
- **Es valorarà l'eficiència de les implementacions. Per una banda,** si amb un únic recorregut i/o cerca es pot determinar el que ha de calcular el mètode no se'n faran ni dos, ni més. **Per altra banda**, s'ha d'aplicar anàlisi descendent en la implementació d'aquells mètodes que poden reaprofitar codi o senzillament són llargs i es trossegen per millorar la seva bona entesa.
- En tota l'aplicació podeu suposar que el dni és un atribut identificador de les persones i el nombre de pòlissa per les assegurances.

## Organització

**Els noms dels vostres projectes, en les pràctiques d'aquesta assignatura han de seguir el següent patró: PràcticaXCognomNom del propietari de la pràctica. En cas de fer-la en parella cal que poseu PràcticaXCognom1Nom1&&Cognom2Nom2. En aquesta pràctica X és 1. Creeu com a mínim un paquet per desar totes les classes que heu de dissenyar en la seva confecció.**

## Què se us subministra?

Fitxer amb l'enunciat de la pràctica.

## Què s'ha de lliurar i com?

S'ha de lliurar la carpeta que conté el **projecte** Eclipse amb el vostre desenvolupament de la pràctica. La carpeta s'ha de lliurar amb tot el seu contingut i comprimida amb ZIP o RAR.

També s'ha de lliurar el **llistat en paper** de tot el codi desenvolupat.

## On s'ha de lliurar?

El lliurament del projecte es farà a través de la plataforma Moodle i no s'acceptarà cap altra via. Feu atenció a la data i hora límit.

**El lliurament en paper es farà directament a la professora al inici de la pràctica 2.**

## Quan s'ha de lliurar?

El lliurament a la plataforma es podrà fer fins el dia anterior a la sessió de la pràctica 2. Tingueu present que a partir d'aquesta hora el sistema bloquejarà, de manera automàtica, la possibilitat de lliurament.

**Grup 101 → 21 d'octubre a les 23:50h**
**Grup 102 → 14 d'octubre a les 23:50h**