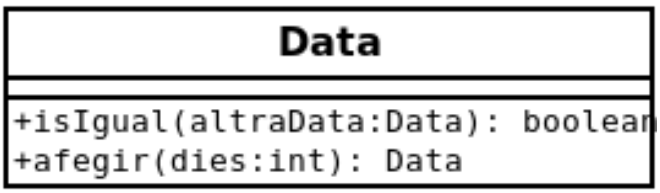
ÍNDEX

ACTIVITAT 6.14 Relació reflexiva

Enunciat

Una de les relacions entre classes més simples que podem trobar en la POO, és dona quan **una classe s'utilitza a si mateixa** per a fer una tasca. En realitat, este tipus de relació ja l'hem treballada en la activitat anterior. Si ens fixem detalladament, podem apreciar-la, en el mètode `isIgual` de la classe `Data`. També, fem servir aquesta relació en els mètodes de `afegir` i `restar` dies de la mateixa classe:



Fixa't que el mètode `isIgual` de la classe `Data` treballa amb un paràmetre de la mateixa classe. Fixa't també que el mètode `afegir` retorna un objecte de la mateixa classe.

Per a practicar més aquest tipus de relació, en aquesta activitat, definirem i implementarem una classe `Punt` que ens permeta representar un punt en un espai bidimensional.

La classe ha de complir com a mínim amb els següents requeriments:

- Cada punt es definix per les seues coordenades **x** e y.
- **Implementa un constructor** amb paràmetres per a assignar valors a les variables d'instància.
- Implementa un mètode public `getDistancia(Punt punt)` que calcule la distància entre el propi objecte i l'objecte `punt` passat com a argument. La fórmula per a calcular la distància entres dos punts (A i B, on A serà `this` i B l'objecte `punt`) és:

$$A(x_1, y_1) \qquad B(x_2, y_2)$$

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Implementa un mètode public `getPuntMitja(Punt punt)` que retorne el punt que es troba

enmig dels dos proporcionats. Per a calcular-lo s'ha de obtindre la meitat de la suma dels valors de la coordenada X i la meitat dels valors de la coordenada Y.

- Implementa el mètode **`toString()`** que crearà i retornarà un String que contindrà les coordenades del punt seguint el següent format: **(coordX, coordY). (Aquest tipus de mètodes son mètodes standard de Java).**

Una vegada implementada la classe **`Punt`**, escriu un programa en una classe **`TestPunt`** que cree els següents punts:

- Un primer punt situat en l'origen (0,0)
- Un segon situat en (5,3).
- Un tercer en (2, -1)
- Un quart estarà situat enmig del segon i del tercer
- El quint serà un punt en (4,3).

Visualitza:

- La distància del punt (4,3) creat anteriorment a l'origen de coordenades (primer punt).
- La informació dels cinc punts creats. *En aquest cas, necessitaries imprimir per consola la informació retornada pel mètode toString. No obstant això, el llenguatge permet imprimir directament l'objecte (en aquest cas, el punt), obtenint el mateix resultat ja que indirectament s'està invocant al mètode* `toString`.

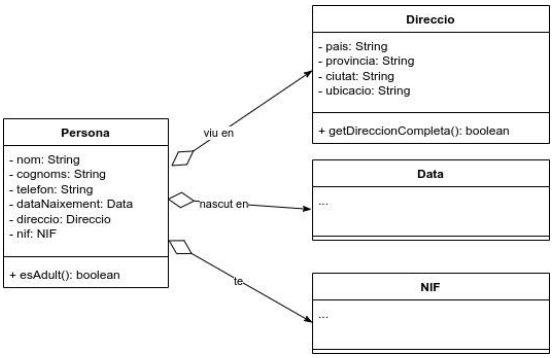
  Per exemple:

```
Punt punt = new Punt(0, 0);

System.out.println(punt);  // equivaldria a System.out.println(punt.toString)
```

ACTIVITAT 6.15 Relació d'agregació

Enunciat

Es pot identificar una associació (relació) d'agregació quan un atribut d'una classe a la qual podem anomenar "tot" té com a atribut una dada pertanyent a una altra classe a la qual anomenarem "part". La particularitat de l'agregació residix en què **la "part" pot existir independentment del "tot"**. A nivell de codi la classe "tot" manté com a atribut una referència a un objecte de la classe "part" que se subministra des d'un constructor o mètode per a afegir l'agregat. L'agregació es representa amb un diamant blanc en el costat del "tot".




```
…
private Direccio direccio;
…
public Persona(Direccio direccio...) {
…
 this.direccio = direccio;
…
}
```

Crea una classe **`Persona`**. La classe contindrà:

- Un atribut **`dataNaixement`** que serà un objecte de la classe **`Data`** creada en l'activitat 13.
- Un atribut **`direccio`** que serà un objecte del tipus **`Direccio`**. Hauràs de crear una classe `Direccio` que agrupe el país, província, ciutat i adreça postal.
- Atributs per al nom, cognoms i telèfon.
- Un atribut **`nif`**, seran objectes de la classe **`Nif`** creada en l'activitat per parelles anteriorment realitzada.
- Un mètode **`public boolean esMajorDeEdat()`** que retornarà true/false indicant si la persona és major d'edat. Una persona és major d'edat si la seua edat és >= 18 anys.
- Un mètode **`public boolean estaJubilat()`** que retornarà true/false indicant si la persona està jubilada. Una persona està jubilada si la seua edat és >= 65 anys.

⚠ Hauràs d'afegir un nou mètode `public int getAnysTranscorreguts(Data data)` en la classe `Data` que retorne el nombre d'anys que han passat entre ella mateixa i un altre objecte de tipus `Data` rebut com a argument.

Crea una classe **`TestPersona`** que continga el mètode **`main()`** i crea objectes que representen a

tota la teua família. A continuació, es presenta un exemple de creació d'un objecte de tipus **`Persona`**.

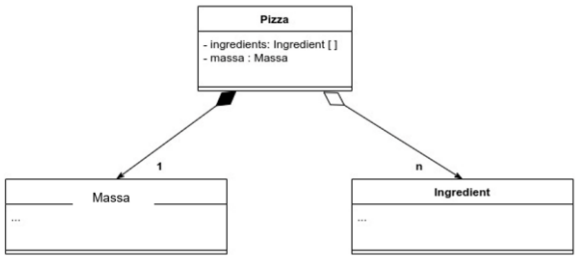
```
public class TestPersona {

public static void main(String[] args) {

 Data dataNaixement = new Data("12/08/1990");

 Nif nif = new Nif("216525612");

 Direccio direccio = new Direccio("Espanya", "Alacant", "Alcoi", "Av.
Hispanitat, 2");

 Persona persona = new Persona(nif, "Pepe", "García", "Sánchez", dataNaixement,
direccio);

 System.out.print(persona.esMajorDeEdat());

 System.out.print(persona.estaJubilat());

 }

}
```

C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

ACTIVITAT 6.16 Relació de composició

Enunciat

De la mateixa manera que amb les associacions d'agregació, podem identificar una associació de composició quan un atribut d'una classe a la qual podem anomenar " tot" té com a atribut una dada pertanyent a una altra classe a la qual anomenarem "part". A diferència de l'agregació, **la classe "part" no pot existir independentment del "tot"**. A nivell de codi, la classe "tot" crea un objecte de la classe "part". La composició es representa amb un diamant negre en el costat del "tot".




```
…
private Massa massa;
…
public Pizza() {
…
this.massa = new Massa();
this.ingredientes = new Ingredient[5];
}
…
// Es crea una pizza només amb la massa i amb espai per a ficar, a posteriori, 5 ingredients.
```

Desitgem crear una aplicació que ens permeta gestionar els preus de les pizzes que anem cuinant. Per a això, necessitarem:

- Una classe **`Ingredient`**: vindrà definida per un nom identificatiu, una descripció i un preu de l'ingredient. Disposarà de 2 constructors un amb tots els paràmetres i un altre solo amb el nom identificatiu establint per defecte un preu de l'ingredient de 1€ i una descripció genèrica.
  - Crea un getter que ens permeta obtindre el preu de l'ingredient.
  - Crea un setter que ens permeta canviar la descripció.

- Una classe **`Massa`**: vindrà definida per un preu base, una descripció i si té la vora farcida de formatge o no. Disposarà de 2 constructors un amb tots els paràmetres i un altre en el qual s'establirà un preu per defecte de 5€, sense la vora farcida de formatge i la següent descripció **"La massa més fina i deliciosa del món".**
  - Definix un setter que ens permeta establir que la vora de la massa està farciment de formatge.
  - Definix un getter que ens permeta obtindre el preu de la massa. Si la vora està farcida de formatge s'incrementarà el preu en 1€.

C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

- Una classe **`Pizza`**: vindrà definida per un nom identificatiu, un conjunt d'ingredients i una massa.
  - Tota pizza tindrà (almenys 2 ingredients) i podrem afegir ingredients extra.
  - Definix un constructor que reba 2 ingredients i el nom identificatiu de la pizza.
  - Afig un mètode **`afegirIngredient(Ingredient ingredient)`** que ens permeta afegir els ingredients extra. El màxim d'ingredients per **`Pizza`** és de 5.
  - La massa es crearà en el propi constructor de la **`Pizza`** fent ús del constructor per defecte de la classe **`Massa`**.
  - Definix un mètode consultor **`getPreu()`** en la classe **`Pizza`** que ens permeta obtindre el preu d'una pizza a partir dels seus components. El preu vindrà determinat per la suma del preu de tots els seus ingredients i de la massa.
  - Definix un mètode **`getNumeroIngredients()`** que retorne el número d'ingredients que té una pizza;
  - Definix un mètode **`setVoraFarcida(boolean isVoraFarcida)`** que ens permeta establir que la massa de la pizza està farcida de formatge. *Este mètode cridarà directament al setter de la massa de la pizza.*
  - Definix un mètode **`toString`** que retorne un `String` composat pel seu nom i els ingredients que conté.

Crea una classe **`TestPizzeria`** i realitza les següents operacions:

- Definix els següents ingredients; tomaca, formatge mozzarella, formatge blau, pernil york, cheddar i formatge base.
- A continuació crea **3 possibles comandes** amb diferents pizzes i mostra el seu preu total (cada comanda vindrà representada per un array de pizzes). Quins tipus de relacions s'estan duent a terme entre les classes proposades?

Per a una millor comprensió, es presenta un exemple d'uºtilització de les classes :

```
public class TestPizzeria {
    public static void main(String[] args) {
        //Ingredients
        Ingredient tomato = new Ingredient("tomato");
        Ingredient baseCheese = new Ingredient("base cheese", "cheddar", 1.5f);
        Ingredient yorkHam = new Ingredient("york Ham");

        //Fabricació pizza margarida
        Pizza prosciutto = new Pizza("prosciutto", tomato, baseCheese);

        prosciutto.afegirIngredient(yorkHam);

        System.out.println("El preu d'una pizza prosciutto és de " + prosciutto.getPreu()); // 8.5€

        //Creació de una comanda amb 3 pizzes
        Pizza[] comanda = {
            prosciutto,
            new Pizza("prosciutto amb pernil", tomato, yorkHam),
            new Pizza("margarida", tomato, baseCheese)};

        //Impresió de la comanda
        for (Pizza pizza : comanda) {
            System.out.println(pizza);
        }
    }
}
```