# *Gestió d'un hostal rural*

Professors d'IIP - DSIC - UPV

Es desitja realitzar una aplicació `GestorHostal` per gestionar l'entrada (*check in*) i l'eixida (*check out*) de clients d'un hostal rural. Per facilitar el desenvolupament d'aquest exercici, es suposen disponibles (en PoliformaT):

- La classe `Data`, una classe tipus de dades que permet representar una data mitjançant els atributs privats de tipus `int`: `dia`, `mes` i `any`. D'aquesta classe s'utilitzaran els mètodes públics que apareixen en el resum de la seua documentació que es mostra en la figura 1.

***Constructor Summary***

**Constructors**

| Constructor and Description |
|---|
| **`Data`**`()`<br>Crea una `Data` amb els valors de la data del sistema. |
| **`Data`**`(int d, int m, int a)`<br>Crea una `Data` amb els valors donats de dia, mes i any. |

***Method Summary***

**All Methods** | **Instance Methods** | **Concrete Methods**

| Modifier and Type | Method and Description |
|---|---|
| `int` | **`difDies`**`(Data d)`<br>Torna el numero de dies entre la `Data this` i un altra `Data d` donada no inclosa, es a dir, el numero de dies en `[this, d[`. |
| `boolean` | **`equals`**`(java.lang.Object o)`<br>Comprova si la `Data this` es igual a un altra donada. |
| `boolean` | **`esAnterior`**`(Data d)`<br>Comprova si la `Data this` es anterior a un altra `Data` donada, suposant ambdues correctes. |
| `boolean` | **`esCorrecta`**`()`<br>Comprova si la `Data this` és correcta. |
| `java.lang.String` | **`toString`**`()`<br>Torna un `String` amb la informacio de la `Data this` en el format dd/mm/aaaa. |

Figura 1: Documentació de la classe `Data`.

- La classe `Client`, una classe tipus de dades que permet representar un client mitjançant els atributs privats: `nif` (`String`), `nom` (`String`), `arribada`, `eixida` (objectes de la classe `Data` tals que `arribada` és anterior a `eixida`) i `regim` (`int` en el rang [0..2] indicant si el client està en règim d'allojament i desdejuni, mitja pensió o pensió completa, respectivament).

  A més, estan definides les constants públiques estàtiques: `AD = 0`, `MP = 1` i `PC = 2`, indicant, respectivament, els règims d'allojament i desdejuni, mitja pensió i pensió completa.

  En la figura 2 es mostra la representació gràfica d'un objecte de la classe `Client`.

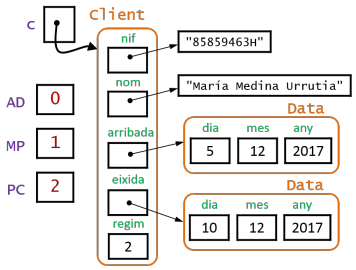
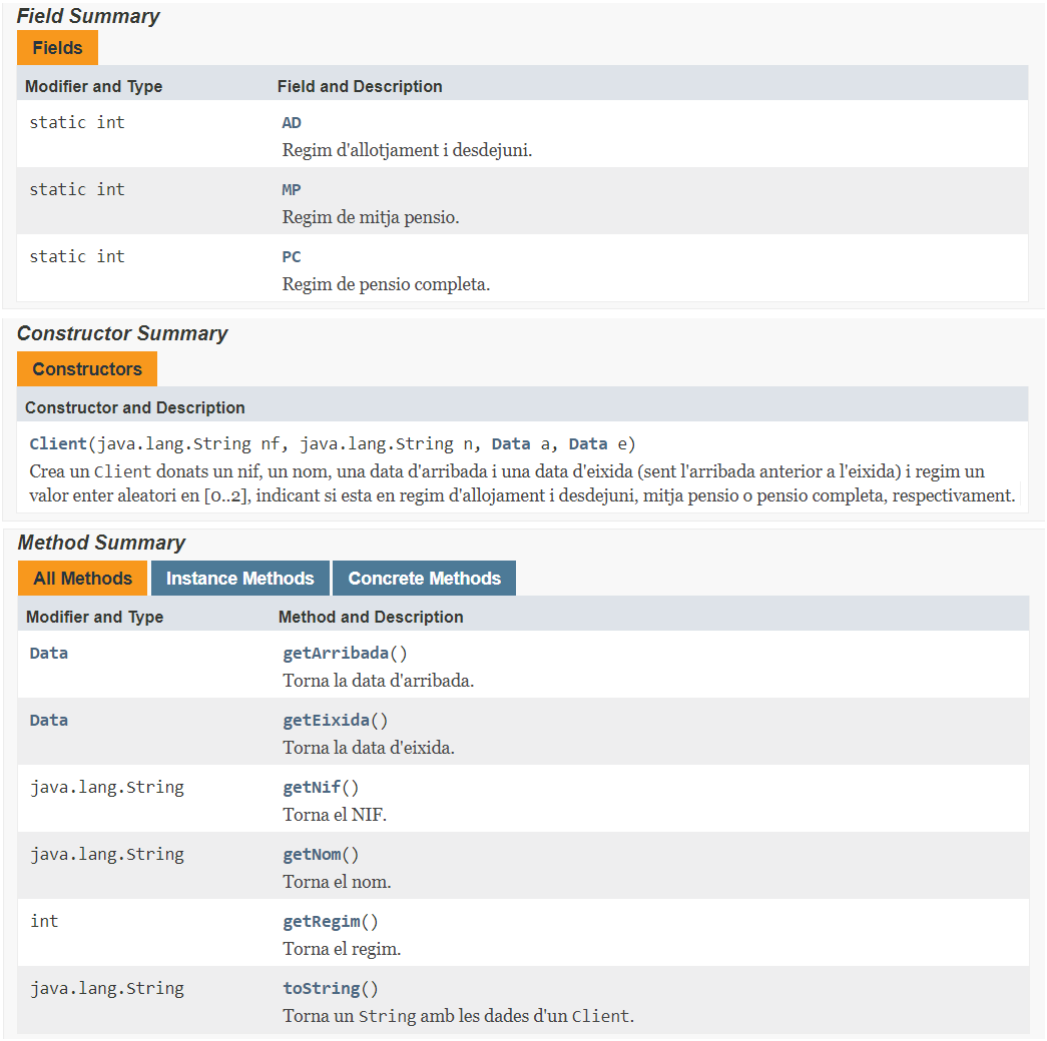



Figura 2: Representació gràfica d'un objecte `Client`.

D'aquesta classe s'utilitzaran les constants i els mètodes públics que apareixen en el resum de la seua documentació que es mostra en la figura 3.



***Field Summary***

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `static int` | **AD**<br>Regim d'allotjament i desdejuni. |
| `static int` | **MP**<br>Regim de mitja pensio. |
| `static int` | **PC**<br>Regim de pensio completa. |

***Constructor Summary***

**Constructors**

| Constructor and Description |
|---|
| **Client**(java.lang.String nf, java.lang.String n, **Data** a, **Data** e)<br>Crea un `Client` donats un nif, un nom, una data d'arribada i una data d'eixida (sent l'arribada anterior a l'eixida) i regim un valor enter aleatori en [0..2], indicant si esta en regim d'allojament i desdejuni, mitja pensio o pensio completa, respectivament. |

***Method Summary***

**All Methods** | **Instance Methods** | **Concrete Methods**

| Modifier and Type | Method and Description |
|---|---|
| **Data** | **getArribada**()<br>Torna la data d'arribada. |
| **Data** | **getEixida**()<br>Torna la data d'eixida. |
| java.lang.String | **getNif**()<br>Torna el NIF. |
| java.lang.String | **getNom**()<br>Torna el nom. |
| int | **getRegim**()<br>Torna el regim. |
| java.lang.String | **toString**()<br>Torna un `String` amb les dades d'un `Client`. |

Figura 3: Documentació de la classe `Client`.

- La classe `GestorHostal`, una clase programa que permet simular l'entrada i l'eixida de clients i consultar les dades de tots els clients i dels clients en règim de pensió completa d'un hostal rural.

**Es demana:** completar la classe tipus de dades següent, tenint en compte que els seus atributs seran privats i els seus mètodes públics o privats i només els que s'indiquen a la classe.

- La classe `Hostal` conté la informació dels clients que ocupen les habitacions d'un hostal rural. Un `Hostal` té un número màxim d'habitacions `MAX_HAB = 25` i per representar-les s'utilitza un array `habitacions` d'objectes de tipus `Client` junt amb un atribut `lliures` que indica el número d'habitacions lliures de l'hostal en un moment donat ($0 \leq$ `lliures` $\leq$ `MAX_HAB`). El número de cada habitació coincideix amb la seua posició a l'array de clients, de manera que `habitacions[i]` és el `Client` que ocupa l'habitació `i` o és `null` si l'habitació està lliure (la posició 0 no s'utilitzarà). També té un atribut `pCompleta` ($0 \leq$ `pCompleta` $\leq$ `MAX_HAB`) que indica el número de clients de l'hostal en règim de pensió completa (és a dir, aquells clients amb `regim` igual a `PC`). A més, per emmagatzemar l'històric de clients que han visitat l'hostal, s'utilitza un array `historic` amb els NIF dels clients junt amb un atribut `clients` que indica quants clients l'han visitat ($0 \leq$ `clients` $\leq$ `MAX_CLIENTS` on `MAX_CLIENTS = 1000` és el nombre màxim de clients considerat). El preu per nit (en €) d'una habitació, tenint en compte el règim, es defineix mitjançant les constants `PREU_AD = 30.0`, `PREU_MP = 40.0` i `PREU_PC = 50.0`.

  En la figura 4 es mostra la representació gràfica d'un objecte de la classe `Hostal`.

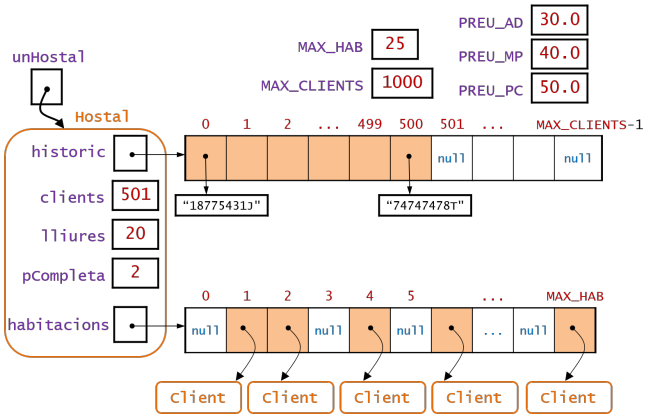



Figura 4: Representació gràfica d'un objecte `Hostal`.

Els mètodes d'aquesta classe són:

- `public Hostal()`. Crea un `Hostal` on totes les habitacions estan lliures, no hi ha clients i, per tant, no hi ha clients en règim de pensió completa ni històric de NIF.
- `public int getLliures()`. Torna el número d'habitacions lliures.
- `public int getClientsHistoric()`. Torna el número de clients en l'històric.
- `public int getPC()`. Torna del número d'habitacions ocupades per clients en règim de pensió completa.
- `public Client getClient(int i)`. Donat un número d'habitació vàlid `i` ($1 \leq i \leq$ `MAX_HAB`), torna el client que ocupa aquesta habitació o torna `null` si està lliure.
- `public boolean hiHaLliures()`. Torna `true` si hi ha habitacions lliures i torna `false` en cas contrari.
- `public int primeraLliure()`. Torna el número de la primera habitació lliure (la de número menor) si hi ha habitacions lliures o torna `-1` si no hi ha.
- `private int cercar(String nif)`. Donat un NIF `nif`, comprova si està a l'històric de NIF. Si està, torna la posició que ocupa a l'array `historic`. En cas contrari, torna `-1` indicant que no s'ha trobat.
- `private void afegirHistoric(Client c)`. Donat un client `c`, si el seu NIF no està a l'històric, l'afegeix.
- `public boolean checkIn(String nif, String nom, Data arribada, Data eixida)`. Check in d'un client de nif `nif`, nom `nom`, data d'arribada `arribada` i data d'eixida `eixida` (sent `arribada` anterior a `eixida`), tornant `true` si s'ha pogut fer i `false` en cas contrari (si no hi ha habitacions lliures). Si hi ha habitacions lliures, la primera d'elles (la de número menor) passa a estar ocupada pel client. Si el nif del client no està a l'històric, l'afegeix.
- `public double checkOut(int i)`. Check out del client que ocupa l'habitació `i` (sent $1 \leq i \leq$ `MAX_HAB`), tornant el preu a pagar o 0 si l'habitació no estava ocupada. L'habitacio `i`, si estava ocupada, passa a estar lliure.
- `public double checkOut(Data d)`. Check out de tots els clients tals que la seua data d'eixida és la data `d` donada, tornant el preu total a pagar o 0 si no hi ha cap client amb aquesta data d'eixida.
- `public String toString()`. Sobreescriptura del mètode `toString()` d'`Object`. Torna un `String` que descriu l'hostal, és a dir, quins clients ocupen quines habitacions i quines habitacions estan lliures. Si no hi ha clients, torna "`Hostal buit`" (acabat en "`\n`"). Per exemple,

```
1     María Medina Urrutia       85859463H  05/12/2017  10/12/2017  PC
2     Pepe Pérez Gutiérrez       74747474I  15/12/2017  21/12/2017  AD
3     lliure
4     Juan López Alegría         12345678J  19/12/2017  31/12/2017  MP
5     lliure
6     Germán García Santaeulalia 98765432M  24/11/2017  04/12/2017  PC
7     lliure
...
25    Andrés Sánchez Miralles    13457892A  27/11/2017  28/11/2017  AD
```

  on el número d'habitació va seguit d'un tabulador "`\t`".
- `public int[] pensioCompleta()`. Torna un array de `int` amb els números d'habitacions ocupades pels clients en règim de pensió completa. La longitud d'aquest array serà igual al número de clients en règim de pensió completa, o 0 si no hi ha cap client en aquest règim a l'hostal.

El resum de la documentació de la classe (constants i mètodes públics) es mostra en la figura 5.

### *Field Summary*

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `static int` | **`MAX_CLIENTS`**<br>Numero maxim de clients. |
| `static int` | **`MAX_HAB`**<br>Numero maxim d'habitacions. |
| `static double` | **`PREU_AD`**<br>Preu/nit d'una habitacio en regim d'allojament i desdejuni. |
| `static double` | **`PREU_MP`**<br>Preu/nit d'una habitacio en regim de mitja pensio. |
| `static double` | **`PREU_PC`**<br>Preu/nit d'una habitacio en regim de pensio completa. |

### *Constructor Summary*

**Constructors**

| Constructor and Description |
|---|
| **`Hostal`**`()`<br>Crea un `Hostal` amb totes les habitacions lliures, es a dir, no hi ha clients i, per tant, no hi ha clients en regim de pensio completa ni historic de NIF. |

### *Method Summary*

**All Methods** | **Instance Methods** | **Concrete Methods**

| Modifier and Type | Method and Description |
|---|---|
| `boolean` | **`checkIn`**`(java.lang.String nif, java.lang.String nom, `**`Data`**` arribada, `**`Data`**` eixida)`<br>Check in d'un client de nif `nif`, nom `nom`, data d'arribada `arribada` i data d'eixida `eixida`, tornant `true` si s'ha pogut fer i `false` en cas contrari (si no hi ha habitacions lliures). |
| `double` | **`checkOut`**`(`**`Data`**` d)`<br>Check out de tots els clients tals que la seua data d'eixida es la `Data d` donada, tornant el preu total a pagar o `0` si no hi ha cap client amb aquesta data d'eixida. |
| `double` | **`checkOut`**`(int i)`<br>Check out del client que ocupa l'habitacio `i` (sent `i` un numero d'habitacio valid), tornant el preu a pagar o `0` si l'habitacio no estava ocupada. |
| **`Client`** | **`getClient`**`(int i)`<br>Torna el **`Client`** que ocupa l'habitacio `i` (sent `i` un numero d'habitacio valid) o `null` si l'habitacio esta lliure. |
| `int` | **`getClientsHistoric`**`()`<br>Torna el numero de clients en l'historic. |
| `int` | **`getLliures`**`()`<br>Torna el numero d'habitacions lliures. |
| `int` | **`getPC`**`()`<br>Torna el numero d'habitacions ocupades per clients en regim de pensio completa. |
| `boolean` | **`hiHaLliures`**`()`<br>Torna `true` si hi ha habitacions lliures i torna `false` en cas contrari. |
| `int[]` | **`pensioCompleta`**`()`<br>Torna un array amb els numeros d'habitacions ocupades pels clients en regim de pensio completa. |
| `int` | **`primeraLliure`**`()`<br>Torna el numero de la primera habitacio lliure (la de numero menor) si hi ha habitacions lliures o torna un `-1` si no hi ha. |
| `java.lang.String` | **`toString`**`()`<br>Torna un `String` que descriu l'Hostal, es a dir, quins clients ocupen quines habitacions i quines habitacions estan lliures. |

Figura 5: Documentació de la classe `Hostal`.

## Validació de la classe `Hostal`

Per tal de comprovar el correcte funcionament del codi de la classe `Hostal`, s'ha preparat un test que realitza un conjunt de proves sobre aquest codi. Perquè aquest test s'execute correctament és indispensable que:

- En la definició dels atributs i mètodes de la classe, utilitzes sempre els noms d'atributs i mètodes proposats a l'enunciat i documentació de l'exercici i en els arxius `.java` proporcionats, respectant a més les característiques descrites per a cadascun d'ells en quant a modificadors i paràmetres.
- Canvies, provisionalment, el valor de les constants `MAX_CLIENTS` i `MAX_HAB` a 10 i 5, respectivament.
- En els mètodes que tornen com a resultat un `String`, seguisques el format i el text indicats.

A continuació es descriuen els passos a seguir per a executar aquest test (veure figura 6).

- Tria l'opció *Comprova-ho tot* (*Test All*) del menú contextual que apareix en fer clic amb el botó dret del ratolí sobre la icona de la classe *Unit Test*. S'executarà una bateria de proves sobre els mètodes de la classe, comparant els resultats esperats amb els realment obtinguts.
- Si els mètodes són correctes, en la finestra *Resultats del test* (*Test Results*) de *BlueJ*, apareixeran marcats amb un ✓ (de color verd). Si, pel contrari, algun mètode no funciona correctament, en la finestra *Resultats del test*, el test de cada mètode incorrecte apareixerà marcat amb una `X`. Si selecciones qualsevol de les línies marcades amb `X`, en la part inferior de la finestra, es mostra un missatge orientatiu sobre la possible causa de l'error.

  Si el mètode que no funciona correctament s'usa en altres mètodes, l'error es propaga a aquests mètodes i, per això, apareixen també marcats com incorrectes en la finestra *Resultats del test*.

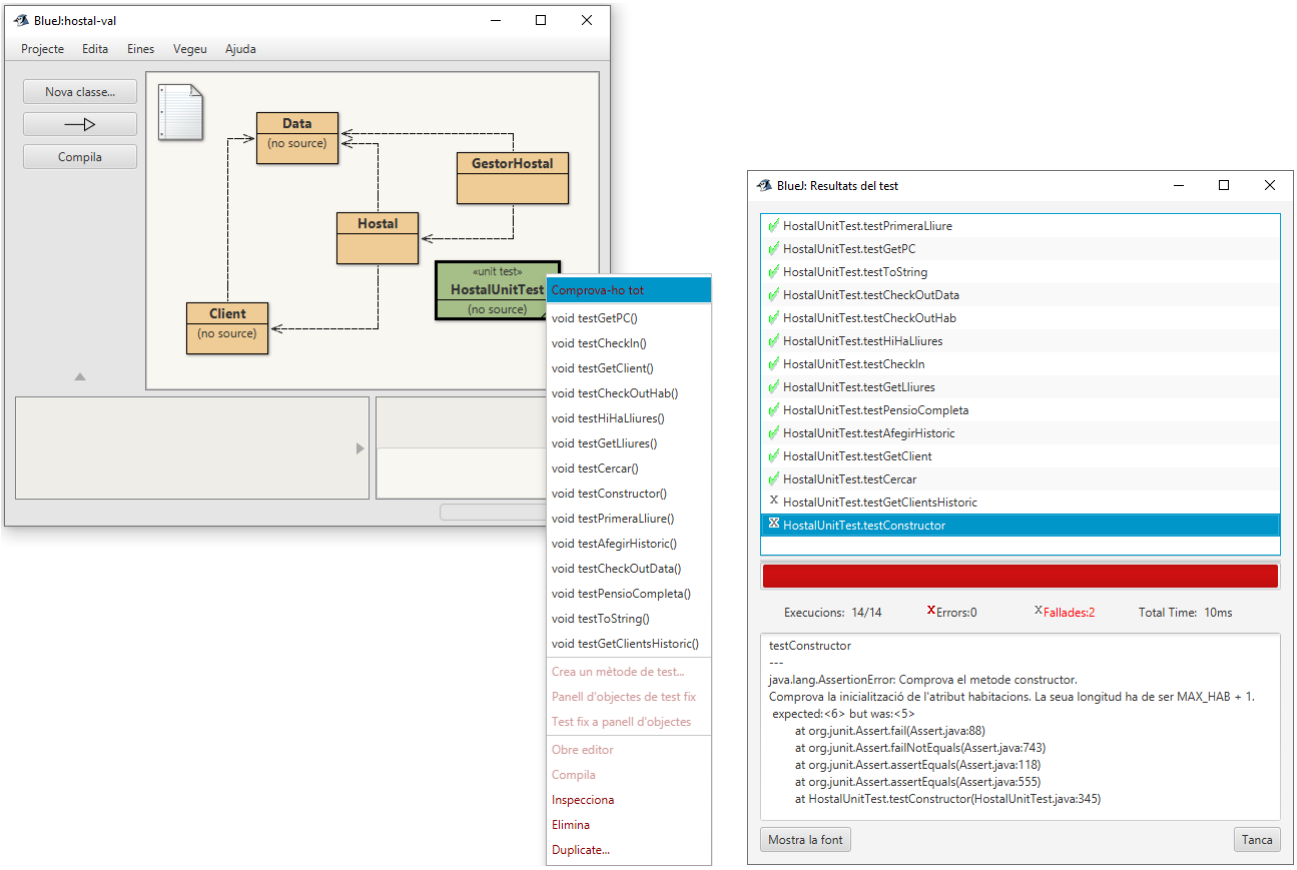



Figura 6: Execució de la *Unit Test* de la classe `Hostal` i *Resultats del test* en *BlueJ*.

- Per a tornar a executar el test, després de corregir els errors i compilar de nou la teua classe, si la icona de la *Unit Test* apareix ratllada, has de tancar i tornar a obrir el projecte *BlueJ*.