Pràctica 1

1. Descripció de la Pràctica

L'objectiu d'aquesta pràctica és desenvolupar un programa, utilitzant el paradigma de Programació Orientat a Objectes, per modelar la informació corresponent a un càmping anomenat *Càmping del Mar* i gestionar els allotjaments que ofereix el càmping. En concret, el programa es centra en gestionar les reserves que els clients poden fer dels allotjaments del càmping i mostrar alguna informació sobre aquests allotjaments.

El càmping disposa d'allotjaments de tipus **Parcel·la**, per allotjar-se amb la pròpia tenda, caravana o autocaravana, i de tipus **Casa.** Les cases poden ser dels següents tipus: **Bungalow** que pot ser normal o **Bungalow Premium**, **Glamping** i **Mobil-Home**.

Els allotjaments s'identifiquen amb un nom i un identificador i tenen assignats un temps d'estada mínima que han de ser reservats (en dies) segons el tipus d'allotjament i la temporada. La **parcel·la** té una mida (en m^2) i en alguns casos un punt de connexió elèctrica disponible. La **casa** té diferents mides (petita, mitjana i gran), un número d'habitacions i una capacitat de places per persones. El **bungalow** té pàrquing amb 1 o 2 places, terrassa, televisió i sistema d'aire fred. A més, el **bungalow Premium** té serveis extres com els llençols i tovalloles incloses i un codi Wifi gratuït. El **glamping** és un altre tipus d'allotjament tipo tenda de campanya de gama alta. Poden ser de tela o de fusta i tenir casa per mascotes. Finalment, la **Mobil-Home** és un allotjament més petit i bàsic, però còmode. És una casa que pot tenir terrassa amb barbacoa.

Un **client** es defineix pel seu nom i DNI (format per un String de 9 caràcters).

Una **reserva** es defineix pel client que la fa i l'allotjament que es reserva per un dia d'entrada i un dia de sortida. La reserva només es podrà realitzar si l'estada sol·licitada (dies des de la data d'entrada fins la data de sortida) és major o igual que l'estada mínima predeterminada segons el tipus d'allotjament i segons si és temporada baixa (del 21 de setembre al 20 de març) o alta (del 21 de març al 20 de setembre).

La taula 1 inclou 5 exemples d'allotjaments dels 5 tipus diferents amb tota la seva informació, inclosa l'estada mínima (en número de dies) per ambdues temporades.

Per una altra banda, els allotjaments queden no operatius segons uns criteris diferents definits per cadascun dels allotjaments:

- Es considera que una parcel·la té un funcionament correcte si té punt de connexió elèctrica.
- Es considera que un Bungalow té un funcionament correcte si té aire fred.
- Es considera que un Bungalow Premium té un funcionament correcte si té aire fred i el codi Wifi assignat té entre 8 i 16 caràcters.
- Es considera que un Glamping té un funcionament correcte si té una casa per a mascota.
- Es considera que una Mobil-Home té un funcionament correcte si té terrassa amb barbacoa.

Les funcionalitats bàsiques disponibles del programa seran:

1. Afegir un allotjament nou.
2. Afegir un client nou.
3. Afegir una reserva d'un allotjament fet per part d'un client per una data i número de dies determinats sempre i quan l'allotjament estigui disponible.
4. Calcular la mida total de les parcel·les (en m^2) del càmping.
5. Calcular el número d'allotjaments del càmping.
6. Calcular el número d'allotjaments operatius mitjançant la comprovació dels criteris definits per cadascun dels allotjaments.

Tipus	**Nom**	**Id**	**Estada mínima (dies) Temp. baixa**	**Estada mínima (dies) Temp. alta**	**Propietats**
Parcel·la	Parcella Nord	100P	2	4	Mida 64 m^2 amb punt de connexió elèctrica.
Parcel·la	Parcella Sud	101P	2	4	Mida 64 m^2 amb punt de connexió elèctrica.
Bungallow	Bungallow Est	102B	4	7	Mida mitjana, 2 habitacions, per 6 persones, 1 plaça de pàrquing, terrassa, televisió, aire fred.
Bungallow	Bungallow Oest	103B	4	7	Mida mitjana, 2 habitacions, per 4 persones, 1 plaça de pàrquing, terrassa, televisió, aire fred.

Bungallow Premium	Bungallow Nord	104B	4	7	Mida gran, 2 habitacions, per 6 persones, 2 places de pàrquing, terrassa, televisió, aire fred, llençols i tovalloles incloses, codi Wifi.
Bungallow Premium	Bungallow Sud	105B	4	7	Mida gran, 2 habitacions, per 6 persones, 1 plaça de pàrquing, terrassa, televisió, aire fred, llençols i tovalloles incloses, codi Wifi.
Glamping	Glamping Nord	106G	3	5	Mida petita, 1 habitacions, per 2 persones, de tela, té casa per mascota.
Glamping	Glamping Sud	107G	3	5	Mida petita, 1 habitacions, per 2 persones, de tela, no té casa per mascota.
Mobil-Home	Mobil-Home Nord	108MH	3	5	Mida petita, 1 habitacions, per 2 persones, amb terrassa amb barbacoa.
Mobil-Home	Mobil-Home Sud	201B	3	5	Mida mitjana, 2 habitacions, per 4 persones, sense terrassa amb barbacoa.

Taula 1: Exemple d'informació dels allotjaments del Càmping del Mar.

2. Material pel lliurament

Per aquesta pràctica proporcionem un codi base que permetrà facilitar el seu desenvolupament i execució.

El codi base conté les següents classes i interfícies:

- ***GestorCamping***
- ***InAllotjament***
- ***InCamping***
- ***InLlistaReserves***
- ***ExcepcioReserva***

I les següents classes de test:

- ***ParcelaTest***
- ***AllotjamentTest***
- ***CampingTest***

- ***ClientTest***
- ***ReservaTest***
- ***LlistaReservesTest***

Trobareu aquest codi al Campus Virtual junt amb aquest enunciat en la secció de PRÀCTIQUES: "Pràctica 1 (enunciat i codi base)". L'heu de descarregar i afegir-lo al vostre projecte.

Sabem que gestionar la creació d'un projecte entre dues persones pot arribar a ser difícil. Per això us recomanem que seguiu la metodologia **Revisió per parells** (o **Pair Programming**) present en el món laboral de la creació de software. Per conèixer aquesta metodologia podeu llegir el següent article: https://www.scrumio.com/blog/que-es-el-pair-programming/

D'altra banda, per poder organitzar les tasques repartides entre els dos companys, podeu aplicar la **Revisió de codi** (o **Code Review**) dels codis que heu fet. Podeu consultar el següent article per informar-vos:
https://es.wikipedia.org/wiki/Revisión_de_código

Com a últim consell, per poder tenir una clara estructura de com es troben distribuïts els codes reviews de les tasques entre vosaltres recomanem l'ús de la següent aplicació:
https://trello.com/es
https://youtu.be/gGcLLDRVYcc?si=499SaH2xooTIzCVR

3. Descripció del Projecte Base

Al Campus Virtual de l'assignatura podràs trobar les classes amb el codi que et servirà de base (totes a la carpeta CodiBase). Un cop hagis creat el teu projecte *IntelliJ* (veure secció **4.1**) i hi hagis afegit aquestes classes, podràs obrir-les i executar el codi de l'aplicació.

3.1 Creació del Projecte Base amb IntelliJ

El primer pas serà crear un projecte al *IntelliJ*, al qual se li ha de posar com a nom Cognom1Nom1Cognom2Nom2, on 1 i 2 fa referència als dos membres de la parella, i tenint en compte les següents consideracions:

- La primera lletra de cada part en majúscula i la resta en minúscula.
- Eviteu utilitzar accents i caràcters del tipus ñ o ç.

  Per exemple, una estudiant amb nom Dolça Martínez Castaña, hauria de crear un projecte amb el nom MartinezCastanaDolca.

Al projecte hauràs de crear dos paquets per organitzar el codi, un anomenat **prog2.vista** que contindrà les classes que implementen la vista del programa proporcionada a l'usuari i un altre paquet anomenat **prog2.model**, que contindrà les classes que implementen les dades del model del Camping.

Quan es crea el projecte en el *IntelliJ* s'ha d'indicar la classe principal (la que conté el main) com **prog2.vista.*GestorCamping***.

3.2 Codi Base: Classe GestorCamping

La classe principal ***GestorCamping*** té com a responsabilitat llançar el bucle principal de l'aplicació i per tant conté el mètode estàtic *main()*. El codi d'aquesta classe, proporcionada en el codi base, està per completar amb comentaris que serveixen d'ajuda. Les línies de codi que falten estan indicades mitjançant un comentari del tipus:

```
// Per completar
```

L'estructura del codi disponible simplifica el desenvolupament de la pràctica, ja que permet interpretar el que s'ha de fer i guiar-vos en la implementació necessària.

En el mètode *main* es crea un objecte de tipus ***Camping*** anomenat *campingMar*. Després es crida als mètode de classe *omplirDadesModel* i *ferReserves* que omplen les dades necessàries del model. Per últim s'ha de completar el codi per mostrar diferents informacions sobre algunes característiques del càmping: la mida total de les parcel·les del camping, el número total d'allotjaments del càmping i el número d'allotjaments operatius.

3.3 Codi Base: Interfícies InCamping, InAllotjament i InLlistaReserves

Una interfície conté la llista de mètodes que haurà d'implementar la classe que implementi la interfície (que faci un "implements" de la interfície). L'ús de la paraula reservada "implements" es detalla en les següents seccions 5.1 i 5.2. Les interfícies ***InCamping***, ***InAllotjament*** i ***InLlistaReserves*** donen forma a les classes ***Camping***, ***Allotjament*** i les seves subclasses i ***LlistaReserves***, respectivament.

3.4 Codi Base: ExcepcioReserva

En aquesta pràctica, heu d'utilitzar excepcions per fer la gestió dels errors, com per exemple, l'error d'afegir una reserva quan l'allotjament no està disponible pel període sol·licitat.

Disposeu de la classe ***ExcepcioReserva***, que anirà dins el [paquet Vista] (***prog2.vista***), per representar aquests errors particulars. A la part del codi on es produeix un problema a controlar, heu de crear un objecte d'aquesta excepció i llançar-la, fent servir ***throw*** dins del mètode corresponent. La paraula ***throws*** al final de la capçalera del mètode ens permetrà delegar la captura d'aquesta excepció fins al mètode on es vol gestionar. Utilitzeu el mateix constructor de l'excepció per introduir-hi el missatge d'error adequat i específic de cada error. Finalment, feu servir *try-catch* per capturar la excepció i llavors informar a l'usuari de l'error produït.

4. Desenvolupament del Projecte

Les classes necessàries per a desenvolupar el sistema de gestió del càmping hauran de formar part del paquet **prog2.model** (que hauràs de crear).

Important: totes les classes del model han de tenir un constructor, i un conjunt de mètodes *getters* y *setters* per accedir i modificar els atributs.

4.1 Classe Camping

La classe **Camping** guardarà un nom, una llista d'allotjaments que emmagatzemarà tots els allotjaments disponibles, una llista de clients que emmagatzemarà els clients del càmping i una llista de reserves.

Les llistes d'allotjaments i de clients s'han d'implementar fent servir la classe ***ArrayList*** i per iterar sobre les llistes de tipus ***ArrayList*** s'ha de fer servir ***Iterators***. Per més informació sobre els iteradors consulteu el document de suport a la pràctica 1 penjat al Campus Virtual.

A més, s'ha d'utilitzar la interfície ***InCamping*** per definir la classe ***Camping*** de la següent manera:

```
public class Camping implements InCamping{

}
```

En la interfície ***InCamping*** estan definits i explicats (en els comentaris) els mètodes que s'han d'implementar. Cadascun dels mètodes *afegirParcela*, *afegirBungalow*, *afegirBungalowPremium*, *afegirGlamping i afegirMobilHome* crea un objecte del tipus d'allotjament corresponent amb la informació necessària rebuda com a paràmetres i l'afegeix a la llista d'allotjaments.

A més, a la classe ***Camping*** s'han d'implementar els següents mètodes de servei:

- Un mètode de suport anomenat *buscarAllotjament* que retornarà l'allotjament corresponent a un identificador passat com a paràmetre. Aquest mètode s'utilitzarà al mètode *afegirReserva* de la mateixa classe ***Camping***.
- Un mètode de suport anomenat *buscarClient* que retornarà el client corresponent a un DNI passat com a paràmetre. Aquest mètode s'utilitzarà al mètode *afegirReserva* de la mateixa classe ***Camping***.

I el següent mètode de classe:

```
public static InAllotjament.Temp getTemporada(LocalDate data);
```

Que retornarà la temporada corresponent a la data passada com a paràmetre. S'ha d'analitzar si la data està en el període del 21 de març al 20 de setembre (temporada alta) o en el període del 21 de setembre al 20 de març (temporada baixa).

4.2 Classe Allotjament i les seves subclasses: Parcela, Casa, Bungalow, BungalowPremium, Glamping, MobilHome.

Podeu deduir els atributs de la classe ***Allotjament*** i les seves subclasses a partir de la descripció de la secció 1. El número de dies de l'estada mínima per cadascun dels allotjaments i per cada temporada (baixa i alta) es pot consultar a la Taula 1.

A més, s'ha d'utilitzar la interfície ***InAllotjament*** per definir la classe ***Allotjament*** fent servir:

```
public abstract class Allotjament implements InAllotjament{

}
```

Les subclasses ***Parcela***, ***Bungalow***, ***BungalowPremium, Glamping*** i ***MobilHome*** han d'implementar el mètode *correcteFuncionament* necessari per aconseguir la funcionalitat 6 definida en la descripció de la pràctica a la secció 1.

Els constructors de les subclasses d'***Allotjament*** han de rebre els seus paràmetres en el mateix ordre que els reben els mètodes *afegirParcela*, *afegirBungalow*, *afegirBungalowPremium*, *afegirGlamping i afegirMobilHome* de la interfície ***InCamping***.

Per últim, s'ha de pensar com s'ha d'implementar el mètode *toString* d'aquestes classes perquè aparegui tota la seva informació, seguint el següent format de l'exemple: "Nom=Parcel·la Nord, Id=100P, estada mínima en temp ALTA: 4, estada mínima en temp BAIXA: 2. Parcela{mida=64.0, connexioElectrica=true}"

4.3 Classe Client

La classe **Client** tindrà dos atributs de tipus String, el nom i el DNI i el seu constructor ha de rebre els paràmetres en el mateix ordre que els rep el mètode *afegirClient* de la classe ***Camping***.

El mètode setter del DNI ha de comprovar que el DNI té 9 caràcters i llançar una excepció de tipus ***ExcepcioReserva*** si no es compleix. És important que el constructor cridi a aquest mètode setter per assignar el valor a l'atribut DNI.

A part, haurà d'implementar el mètode *toString* per donar el format desitjat a les impressions per pantalla amb la informació de la classe.

4.4 Classe Reserva

La classe ***Reserva*** guardarà els atributs que s'han descrit a l'enunciat. Dos dels atributs seran la data d'entrada i data de sortida de la reserva, fent servir el tipus ***LocalDate*** (consultar la secció 5.4.1).

El seu constructor ha de rebre els paràmetres en el mateix ordre que els rep el mètode *afegirReserva* de la classe ***LlistaReserves***.

4.4.1 Classe LocalDate

La classe ***LocalDate*** és una classe de la llibreria de Java que permet treballar amb dates.

Per tal d'utilitzar la classe us donem diversos exemples:

- Per definir la data i hora 20 de Febrer del 2025 es fa de la següent manera:

```
LocalDate date = LocalDate.of (2025, 2, 20);
```

- Donat el dia d'entrada i dia de sortida d'una reserva (***LocalDate*** dataEntrada i dataSortida) es pot calcular l'estada (en número de dies) de la reserva fent servir la classe ***ChronoUnit*** de Java de la següent manera:

```
long estada = ChronoUnit.DAYS.between(dataEntrada, dataSortida);
```

- Haureu de comparar si l'estada compleix que és més llarga o igual que l'estada mínima predefinida per l'allotjament en la temporada corresponent a la data d'entrada (consultar següent secció).

- Haureu de comparar dates i comprovar si les dates sol·licitades per la reserva d'un allotjament estan dins d'un interval d'una altra reserva del mateix allotjament (consultar següent secció).

4.5 Classe LlistaReserves

La classe ***LlistaReserves*** encapsula la llista de reserves del càmping. Ha de definir com a atribut un *ArrayList<Reserva>* i s'ha de definir fent servir la interfície ***InLlistaReserves***.

```
public class LlistaReserves implements InLlistaReserves{

}
```

A més a més, s'han d'implementar els següents mètodes de suport:

- Un mètode anomenat *allotjamentDisponible*. Aquest mètode rep com a paràmetres un objecte de tipus ***Allotjament*** i dos objectes de tipus ***LocalDate*** (les dates de entrada i de sortida) i retornarà un booleà per indicar si l'allotjament passat està disponible per fer la reserva en el dia indicat.

- Un mètode *isEstadaMinima.* Aquest mètode rep com a paràmetres un objecte de tipus ***Allotjament*** i dos objectes de tipus ***LocalDate*** (les dates d'entrada i de sortida) i retornarà un booleà per indicar si l'estada sol·licitada compleix que és més llarga o igual que l'estada mínima de l'allotjament en la temporada corresponent a la data d'entrada.

Aquests dos mètodes s'utilitzaran al mètode *afegirReserva* de la mateixa classe. El mètode *afegirReserva* rep un allotjament, un client i la data d'entrada i de sortida per fer la reserva (com a tipus ***LocalDate***) i ha de fer el següent:

- Comprovar que l'allotjament estigui disponible pel període entre el dia d'entrada i de sortida. En cas negatiu, llança una excepció de tipus ***ExcepcioReserva*** amb el missatge: "L'allotjament amb identificador ID no està disponible en la data demanada DATA pel client CLIENT amb DNI: DNI.", a on ID, DATA, CLIENT i DNI canviaran en funció de l'execució. Un exemple és: "L'allotjament amb identificador 100P no està disponible en la data demanada 2025-02-20 pel client Lluís Plans amb DNI: 789101A."

- Comprovar que l'estada que s'està sol·licitant compleixi que és d'un número de dies major o igual que l'estada mínima per aquell tipus d'allotjament i per la temporada corresponent a la data d'entrada. En cas negatiu, llança una excepció de tipus ***ExcepcioReserva*** amb el missatge: "Les dates sol·licitades pel client CLIENT amb DNI: DNI no compleixen l'estada mínima per l'allotjament amb identificador ID.", a on ID, DATA, CLIENT i DNI canviaran en funció de l'execució.

- En cas afirmatiu de les dues comprovacions, crea la reserva i l'afegeix a la llista de reserves del càmping.

4.6 Classes de test

Per comprovar el correcte funcionament del codi implementat és necessari córrer un conjunt de casos de prova (Unit Tests) que us hem proporcionat en les classes de test del codi base.

Un cas de prova (Unit Test Case) és una part de codi, que assegura que una altra part del codi (un mètode) funciona com s'esperava. Un cas de prova es caracteritza per una entrada coneguda i un resultat esperat.

JUnit és un marc de proves unitàries per al llenguatge de programació Java (https://junit.org/junit5/, https://www.jetbrains.com/help/idea/junit.html).

Teniu un document de suport sobre JUnit al Campus Virtual.

5. Format del Lliurament

El lliurament consistirà en tot el codi generat en els diferents punts de l'enunciat, juntament amb la documentació especificada en aquest apartat.

En concret, cal generar un fitxer comprimit (ZIP) amb el nom dels dos membres de la parella: **Cognom1Nom1Cognom2Nom2_Pr1**, que contingui:

A. El projecte sencer de IntelliJ

Tot el codi generat ha d'estar correctament comentat.

B. La memòria del lliurament.

La memòria ha de contenir els punts descrits en la normativa de pràctiques i els punts següents:

1. Explicar les classes implementades.
2. Dibuixar el diagrama de relacions entre les classes que has utilitzat a la teva pràctica. No cal incloure la llista d'atributs i mètodes.

3. Explicar quins són els atributs de la classe Reserva i perquè.
4. Expliqueu on s'implementen els mètodes *correcteFuncionament* i perquè.
5. Expliqueu en quines classes heu implementat el mètode *toString* i perquè.
6. Expliqueu com has definit els atributs de les classes (public o private) i perquè.
7. Expliqueu si seria correcte que es creïn el client i l'allotjament quan es crea una reserva. Es a dir, seria correcte canviar el mètode *afegirReserva* de la classe ***Camping*** de la següent manera?:

```
public boolean afegirReserva(String nomAllotjament, String  idAllotjament, long
estadaMinimaALTA_, long estadaMinimaBAIXA_, String nomClient, String dni,
LocalDate dataEntrada, LocalDate dataSortida){

    Allotjament allotjament = new Allotjament(nomAllotjament, idAllotjament,
    tempsSlotReserva, estadaMinimaALTA_, estadaMinimaBAIXA_);

    Client client = new Client(nomClient , dni);

    llistaReserves.afegirReserva(allotjament, client, data Entrada, dataSortida);

}
```

8. Detallar les proves realitzades amb JUnit tests per comprovar el correcte funcionament de la pràctica, resultats obtinguts i accions derivades.
9. Observacions generals.

6. Data Límit del Lliurament

Consultar el calendari de lliuraments al Campus Virtual.