PROGRAMACIÓN **28/03/2017**
NOME:

Un Hotel desexa realizar unha aplicación JAVA para levar control das reservas e ocupación das súas prazas dispoñibles. Para elo **se almacenará a información nos seguintes ficheiros**:

clientes.dat - Almacenará a información dos clientes do hotel, de xeito permanente. A información a almacenar será:
DNI, nome, apelidos, dirección, teléfono e e-mail.

habitacions.dat - Ficheiro que almacenará a información de todas as habitacións do hotel. A información será:
Número, plazas, precio/día

As habitacións do hotel cambian moi rara vez, de forma que este ficheiro permanece constante.

reservas.dat - Ficheiro que almacenará a información das reservas realizadas. O número de reserva se xenera de xeito automático cando se formaliza unha reserva a partir dun número que se almacena ao principio do ficheiro, que será 1 no inicio da aplicación e que se irá incrementando automáticamente segun se realizan as distintas reservas. A continuación dese número se almacenará información das reservas, que será a seguinte:

nº reserva, nº habitación, dni, data entrada reserva, data saída reserva, data entrada real, data saida real

A aplicación visualizará un menú coas seguintes opcións:

1.- Reserva
2.- Consulta Reserva
3.- Entrada
4.- Saida
5.- Listado Habitacións Libres
6.- Listado de Ocupacións

Opción 'Reserva'

Permite a un cliente solicitar unha reserva de habitación:

a) A aplicación solicitará a data de entrada de saída.
b) Se visualizarán as habitaciones dispoñibles indicando o seu número de plazas e prezo.
c) Se pedirá o número de habitación desexada, controlando que sexa unha das visualizadas con anterioridade.
d) Se lle solicitará o DNI ao cliente.
- Si o cliente xa existe, se visualizarán os seus datos, e se non, se solicitarán os datos que faltan e se dará de alta.

e) Se dará de alta a reserva

NOTA: Suponse que nunca se estarán dando dúas reservas ao mesmo tempo

Opción 'Consulta Reserva'

Permite consultar os datos dunha reserva a partir do seu número de reserva e anulala. Ou visualizar todas as reservas realizadas por un cliente, incluíndo as xa utilizadas.

a) Se visualizará un menú coas opcións: ***por DNI***, ***por Nº de Reserva***
- ***Por DNI:*** Se solicitará o DNI e se visualizarán todas as reservas do cliente.
- ***Por Nº de Reserva:*** Se solicitará o Nº de Reserva, si a reserva existe, se visualizará e se preguntará si se desexa anular. Si se dice que sí, se cancelará a reserva.

Opción 'Entrada'

Realiza o "check-in" dun cliente, rexistrando a data de entrada no hotel na súa reserva

a) Se solicitará o Nº de Reserva
b) Se modificará ***reservas.dat*** poñendo a data actual como data de entrada, deixando a ***null*** a data de saída

Opción 'Saída'

Realiza o "check-out" dun cliente, rexistrando a data de marcha do hotel na súa reserva

a) Se solicitará o Nº de Reserva
b) Se modificará ***reservas.dat*** poñendo a data actual como data de saída

Opción 'Listado de Habitacións Libres'

Lista as habitacións libres entre dúas datas

a) Se solicitará a data de entrada e a data de saída
b) Se visualizará a lista das habitacións libres entre esas dúas datas.

Opción 'Listado de Ocupacions'

Se visualizará unha lista de todas as habitacións ocupadas actualmente, cos datos do cliente.

Se consideran necesarias as seguintes clases:

package hotel;

public class Reserva {

```
private int numReserva;
private Cliente cliente;
private Habitacion habitacion;
private Date entradaReserva;
private Date saidaReserva;
private Date entrada=null;
private Date saida=null;
```

/ Constructor**
Crea o obxecto Reserva identificado por numreserva consultando os datos en *reservas.dat***. Si a reserva non existe, lanzará unha ReservaException.**
***/**
public Reserva(int numreserva) throws ReservaException;

/ Constructor**
Crea o obxecto reserva coa información suministrada e a almacena no ficheiro *reservas.dat* **xenerando o número de reserva apropiado. Si algo falla lanzará unha ReservaException**
***/**
public Reserva(Cliente cliente,Habitacion habitacion,Date entrada, Date saida) throws ReservaException;

/ save**
Almacena a información do obxecto no ficheiro *reservas.dat* **de xeito apropiado, reutilizando o espacio das reservas borradas (numero de reserva a 0), xenerando o número de reserva e actualizando o contador do inicio do ficheiro, si funciona correctamente devolverá o número de reserva xenerado, en caso de erro lanza unha ReservaException.**
***/**
private int save() throws ReservaException;

/ listaReservas**
Devolve un ArrayList cos obxectos Reserva correspondentes ao cliente co dni indicado
***/**
public static ArrayList <Reserva> listaReservas(String dni);

/ listaReservas**
Devolve un ArrayList cos obxectos Reserva activos correspondentes a habitación indicada unha reserva está activa, si a data de saída da habitación está a null, e non está anulada.
***/**
public static ArrayList <Reserva> listaReservas(int numhabitacion) throws ReservaException;

/ cancel**
Cancela a reserva poñendo en *reservas.dat* **(e no propio obxecto) o número de reserva a 0.**
***/**
public void cancel() throws ReservaException;

/ entrada**
Procesa a reserva creando a entrada correspondente en *ocupacion.dat*
***/**
public void entrada() throws ReservaException;

/ saida**
Modifica o rexistro correspondente en *ocupacion.dat* **para poñer a data de saída (a de hoxe)**
***/**
public void saida() throws ReservaException;

/ listaOcupantes**
Devolve un ArrayList coas reservas dos clientes que están ocupando as habitacións.
***/**
public static ArrayList <Reserva> listaOcupantes();

}

package hotel;

public class Cliente {

private String dni;
private String nome;
private String apelidos;
private String direccion;
private String telefono;
private String email;

/ Constructor**
Recupera de *clientes.dat* **o cliente co dni indicado, lanzando unha ClienteException si non se atopa.**
***/**
public Cliente(String dni) throws ClienteException;

/ Constructor**
Crea o obxecto Cliente, e o da de alta no ficheiro clientes.dat **si o dni xa existe, lanza unha ClienteException.**
***/**
public Cliente(String dni,String nome,String apelidos,String direccion,String telefono,String email) throws ClienteException;

/ save**
Almacena a información do obxecto Cliente en *clientes.dat***. Si o dni xa existe, lanza unha ClienteException**
***/**
private void save() throws ClienteException;

}

package hotel;

public class Habitacion {

int numero;
int plazas;
double precio;

/ Constructor**
Crea o obxecto Habitacion
***/**
public Habitacion(int numero, int plazas, double precio);

/ lista**
Devolve un ArrayList con todas as habitacións do ficheiro *habitacions.dat*
***/**
public static <ArrayList> Habitacion listaTodas();

/ listaLibres**
Devolve un ArrayList coas habitacións de habitacions.dat que non estan reservadas nin ocupadas (non teñen reserva) nas datas indicadas..
***/**
public static <ArrayList> Habitacion listaLibres(Date entrada, Date salida);

}

Se pide:

1.- *Indica* qué tipo de acceso realizarías a cada un dos ficheiros e razona por qué.

2..- *Codifica* na clase ***Reserva*** o método:

public void saida() throws ReservaException;

Que realiza a "saida" no hotel para a reserva actual gardando a data de hoxe no campo ***data saida real.*** En caso de erro se debe lanzar a excepción. O acceso a os ficheiros deben empregar o sistema de acceso indicado no exercicio **1.**

3.- *Codifica* na clase ***Reserva*** o método:

public static ArrayList <Reserva> listaOcupantes();

Que devolve a lista dos clientes que están ocupando habitación a data de hoxe (reservas con data de entrada que non teñen data de saída). Debes utilizar o sistema de acceso indicado no exercicio **1.**

4.- *Codifica* **os dous constructores** da clase Cliente, utilizando o sistema de acceso indicado no exercicio **1.**

5.- *Implementa* a aplicación **Hotel** mediante a clase **Hotel.java**, pertencente ao package por defecto Java, de xeito que visualice o menú principal, e nas distintas opcións indique simplemente pola pantalla "Opción (*descripción da opción)* En Desenvolvemento", salvo a opción **6** (*Listado de Ocupacións*) , que debes implementar (contando que todas as clases anteriores están completas).

RECORDA QUE PODES IMPLEMENTAR OS MÉTODOS ADICIONAIS QUE CONSIDERES OPORTUNOS PARA O TEU TRABALLO, E QUE OS MÉTODOS DESCRITOS NO ENUNCIADO SE SUPOÑEN IMPLEMENTADOS.

SERÁ NECESARIO CODIFICAR CALQUEIRA MÉTODO QUE NON ESTÉ NOS MÉTODOS INDICADOS NAS CLASES ANTERIORES OU CALQUEIRA CLASE ADICIONAL.

Criterios de Avaliación:

Exercicio 1 - 1 ptos
Exercicio 2 - 2.5 ptos
Exercicio 3 - 2 ptos
Exercicio 4 - 2.5 ptos
Exercicio 5 - 2 ptos