PROGRAMACION **02/12/2024**

NOME:

Desexamos programar o sistema de xestión de reservas ou "***Manager***" dun Hotel. O "Manager" debe visualizar un menú que permita:

- *Consultar e dar de alta novos clientes*
- *Facer Reservas*
- *Facer o Check-In dunha Reserva*
- *Facer o Check-Out dunha Reserva*
- *Cancelar unha reserva*

O "Manager" debe visualizar un menú con esas operacións mentres non se elixa a opción de saír e pertence ao **package hotelmanager**

Un Hotel ten un ***nome***, un ***conxunto de habitacións (Room)*** e un ***conxunto de clientes (Client)*** do hotel que son accesibles unicamente dende dentro da propia clase. Ningún dos seus atributos son modificables unha vez establecidos

O construtor do Hotel debe recibir o nome do hotel e o Array de habitacións (***Room***) das que dispón. Os Hotel teñen un número de habitacións fixa, pero o número de clientes é variable, e pode ser calquera número. Polo tanto, o conxunto de clientes se xestionará mediante un obxecto da clase ***DinamicArray*** que será descrito posteriormente.

Os Clientes (Client) teñen un ***dni*** que non pode ser modificado unha vez establecido, un ***nome (firstname)*** uns ***apelidos (lastname)*** un ***e-mail (email)*** e un ***teléfono (phone).*** Todos eles deben estar accesibles unicamente dende dentro da propia clase. *O dni non é modificable unha vez establecido.*

Client ten dous construtores, un deles recibe todos atributos como parámetros, mentres que o outro unicamente recibe o dni, o nome e os apelidos, inicializando ao string "" o resto.

Se considera que dous clientes son iguais si e soamente si os seus DNI son iguais

As habitacións (Room) teñen un ***número de habitación (number)***, un ***nome (name)***, unha ***capacidade de hospedaxe (capacity)*** e un ***conxunto de reservas (Booking) realizadas (bookings)***. Todos eles deben ser accesibles unicamente dende dentro da propia clase. O conxuo de reservas é variable polo que será tamén xestionado por un obxecto da clase ***DinamicArray***. Ningún dos seus atributos son modificables unha vez establecidos.

O construtor de Room recibe todos os atributos como parámetros.

As reservas (Booking) dispoñen dun ***número de reserva (number)***, unha ***data de comezo de tipo LocalDateTime (startdate)*** unha d***ata de finalización de tipo LocalDateTime (enddate)***, o ***cliente que fixo a reserva (guest),*** a ***habitación reservada (room)*** e un ***indicador de si está feito o checkin da reserva ou non (checkin).*** Ningún dos seus atributos é modificable unha vez establecidos

O construtor de Booking recibe: unha ***startDate***, e unha ***endDate*** de tipo LocalDate, o **Client** que fai a reserva e a **Room** reservada. A reserva se fai con hora de inicio 12:00 e hora de finalización 12:00. O número de reserva se xera de xeito automático, de modo que cada novo obxecto da clase Booking que se cree terá un número novo asociado (o primeiro obxecto creado é o 1, o segundo o 2, o terceiro o 3.... etc)

Todas estas clases relacionadas co hotel están no ***package hotel,*** mentres que o Manager está no ***package hotelmanager***

DinamicArray é unha clase pertencente ao ***package utils*** que implementa a xestión dinámica dunha colección de obxectos. Os obxectos DinamicArray teñen como atributo o **número de elementos almacenados (number)**, e ***un array Object[] (array)*** onde se almacenan os obxectos.

O seu construtor recibe como argumento o tamaño inicial do array creando o array inicial de obxectos, que estará baleiro (number=0): ***public DinamicArray(int initialcap)***

Os seus atributos so son accesibles dende dentro da propia clase, e dispón dos seguintes métodos públicos:

```
/**
 * Devólve o número de elementos actualmente no array (que será <= que o tamaño actual do array.
 *
 * @return O número de elementos almacenados.
 */
public int length()
/**
 * Devólve a capacidade actual do array (tamano físico do array subxacente).
 *
 * @return A capacidade do array.
 */
public int size()
/**
 * Convierte o array actual en un array novo, copiando todos os seus elementos.
 * Se o array está vacío, retorna un array vacío.
 *
 * @return Un novo array coas copias dos elementos actuais.
 */
public Object[] toArray()
/**
 * Engade os elementos dun array de obxectos ao array actual.
 *
 * @param array O array de obxectos que se vai engadir.
 */
public void append(Object[] array)
/**
 * Engade os elementos dun outro DinamicArray ao array actual.
 *
 * @param da O DinamicArray que se vai engadir.
 */
public void append(DinamicArray da)
/**
 * Engade un obxecto ao final do array. Se o array está cheo, incrementa o seu tamaño automaticamente.
 *
 * @param obj O obxecto a engadir.
 */
public void add(Object obj)
/**
 * Obtén o obxecto no índice especificado.
 *
 * @param idx O índice do obxecto a obter.
 * @return O obxecto no índice especificado.
 * @throws IndexOutOfBoundsException Se o índice está fóra de rango.
 */
public Object get(int idx) throws IndexOutOfBoundsException
/**
 * Substitúe o obxecto no índice especificado e devolve o valor antigo.
 *
 * @param idx O índice do obxecto a modificar.
 * @param object O novo obxecto que se gardará no índice.
 * @return O obxecto antigo no índice especificado.
 * @throws IndexOutOfBoundsException Se o índice está fóra de rango.
 */
public Object put(int idx,Object object) throws IndexOutOfBoundsException
 /**
 * Elimina o obxecto no índice especificado e despraza os elementos seguintes.
 *
 * @param idx O índice do obxecto a eliminar.
 * @return O obxecto eliminado.
 * @throws IndexOutOfBoundsException Se o índice está fóra de rango.
 */
public Object remove(int idx) throws IndexOutOfBoundsException
/**
 * Comproba se un obxecto dado está presente na lista.
 * Este método recorre a lista e devolve o índice do primeiro elemento que coincide co obxecto fornecido.
 * Se o obxecto non se atopa na lista, devolve null.
 *
 * @param obj O obxecto que se quere buscar na lista.
 * @return O índice do obxecto na lista se se atopa, ou null se non se atopa o obxecto.
 */
public Integer position(Object obj)
```

Como non desexamos a introdución de datos incorrectos, incorporamos ao ***package utils*** a clase:

```
package utils;

public class Validator {
    public boolean isValid(Object data) {
        throw new UnsupportedOperationException();
    }

    public String message() {
        return "";
    }
}
```

E ademais unha clase **Input** con métodos estáticos para a realización de entradas validadas:

```
package utils;

import java.util.Scanner;

public class Input {
    private static Scanner scn=new Scanner(System.in);
    private static final String CANCELSTR="*";

    /**
     * Para poder reutilizar esta operación en posibles diferentes entradas de datos...
     * ( string, double, dni, email, diasemana, date, time .... )
     * aforrando código
     *
     * @param msg
     * @return
     * @throws CancelException
     */
    public static String getStr(String msg) throws CancelException {
        System.out.print(msg+" ("+CANCELSTR+" para cancelar): ");
        String input=scn.nextLine();
        if (input.equals(CANCELSTR)) throw new CancelException();
        return input;
    }

    public static String validString(String msg,Validator validator) throws CancelException {
        boolean ok=false;
        String str;

        do {
            str=getStr(msg);
            if (!validator.isValid(str)) System.out.println("\t"+validator.message());
            else                         ok=true;
        } while(!ok);
        return str;
    }

    public static String getDni(String msg) throws CancelException {
        return validString(msg,new DniValidator());
    }
```

A clase ***DniValidator*** e unha clase que é un ***Validator*** capaz de indicar si un DNI español é válido ou non.

Se pide:

1.- Crear o xestor de reservas Manager. As distintas opcións do Menú simplemente amosarán unha mensaxe "Opcion <nome da opción> En Desenvolvemento". Cando o Manager se inicia se debe crear o Hotel a xestionar con 5 habitacións, 2 simples, 2 dobles e 1 triple.

2 Puntos.

Criterios de avaliación

- Atributos correctamente definidos. 20%
- Uso de programación estruturada sen rotura de bucles e con condicións correctamente estruturadas 30%
- Control de erros eficiente e completo 20%
- Funcionamento correcto do menú 20%
- Comentarios axeitados no código 10%

2.- Definir as clases ***Client, Booking, Hotel e Room***. Debe terse en conta que un Hotel ten un número indeterminado de Clientes, e que unha Room ten un número indeterminado de reservas (Booking), polo que se debe facer uso de ***DinamicArray***. Se deben definir únicamente os atributos co tipo de datos e protección de acceso axeitados e os construtores e métodos indicados no enunciado.

5 Puntos

Criterios de avaliación

- Clase Client: 1.50pts
  - Non permite a creación de Client con dni erróneos (fai uso do DniValidator do package Utils) 50%
  - Os atributos están correctamente definidos e coa protección de acceso indicada no enunciado. 12,5%
  - Os construtores están correctamente definidos. 12,5%
  - Está establecido correctamente o criterio de igualdade dos clientes 25%
- Clase Hotel 1pts
  - Os atributos están correctamente definidos e coa protección de acceso indicada no enunciado. 75%
  - O construtor está correctamente definido. 25%
- Clase Room 1pts
  - Os atributos están correctamente definidos e coa protección de acceso indicada no enunciado. 75%
  - O construtor está correctamente definido. 25%
- Claase Booking 1,50pts
  - Os atributos están correctamente definidos e coa protección de acceso indicada no enunciado. 30%
  - O construtor está correctamente definido e se xenera correctamente o número de reserva 70%

3.- Programar completamente a clase ***DinamicArray***. Para crear esta clase se pode facer uso do método System.arraycopy ou os métodos estáticos ca clase Arrays (en particular copyOf). A clave da cuestión é ter en conta que o número de elementos almacenados sempre debe ser menor ou igual que o espazo dispoñible no array. Si nun instante engadimos un elemento e non cabe, será necesario crear un novo array máis grande (o mellor é aumentalo en "bloques" por exemplo de 20 elementos para non ter que estar aumentando continuamente) e copiar os datos existentes ao novo array que substituirá ao vello.

3 puntos

Criterios de avaliación

- Os atributos están correctamente definidos e coa protección de acceso indicada 2%
- Se retornan correctamente a lonxitude (o número de elementos no array) e o tamaño (o tamaño físico do array) 3%
- Se engaden correctamente os elementos ao array, que crece cando é necesario 20%
- Se recuperan correctamente elementos polo seu índice 5%
- Se almacenan correctamente elementos na posición indicada (sobreescriben, non insertan) 5%
- Se eliminan correctamente elementos da posición indicada 20%
- Se determina correctamente a posición dun obxecto nunha lista segundo o criterio de igualdade do obxecto 20%
- Se engade correctamente un array de obxectos ao DinamicArray 10%
- Se engade correctamente un DinamicArray a este DinamicArray 10%

- Se retorna correctamente un Array do tamaño axeitado coa información do DinamicArray 5%

4.- (Adicional) – Se pide crear o ***DniValidator***, que é un Validator que valida DNI e ***CancelException***, que é a Exception que lanzan as operacións de entrada de datos si o usuario decide cancelar a operacion. Ambas pertencen ao ***package utils***

Requisitos para validar un DNI español:

1. Un DNI debe ter **8 díxitos seguidos dunha letra**.
2. A letra de control debe coincidir coa calculada a partir dos 8 díxitos. A fórmula para a letra de control é:
    - Dividir os 8 díxitos entre 23 e utilizar o resto para buscar a letra correspondente na táboa: `TRWAGMYFPDXBNJZSQVHLCKE`.

1 puntos

Criterios de avaliación

- A clase DniValidatorr está definida correctamente como un Validator 40%
- A clase DniValidator ten todos os métodos necesarios e serve para a validación dos DNI 40%
- A clase CancelException está correctamente construída 20%

NOTA: A nota máxima do exame son 10 puntos, aínda que os exercicios petrmitirían acadar máis puntuación