PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJE DE PROGRAMACIÓN 2

2da. práctica (tipo b)
(Segundo Semestre 2023)

Indicaciones Generales:

- Tiempo estimado: 1h 50 minutos
- Se les recuerda que, de acuerdo al reglamento disciplinario de nuestra institución, constituye una falta grave copiar del trabajo realizado por otro estudiante o cometer plagio para el desarrollo de esta práctica.
- Está permitido el uso de apuntes de clase, diapositivas, ejercicios de clase y código fuente. (Debe descargarlos antes de iniciar con la solución del enunciado)
- No está permitido el uso de entornos de desarrollo integrado (IDEs).
  Debe utilizar un editor de texto: Notepad++, Sublime, etc.
- Está permitido el uso de Internet (únicamente para consultar páginas oficiales de Microsoft y Oracle). No obstante, está prohibida toda forma de comunicación con otros estudiantes o terceros.

PARTE PRÁCTICA (20 puntos)

PUEDE UTILIZAR MATERIAL DE CONSULTA.
Se considerará en la calificación el uso de buenas prácticas de programación (aquellas vistas en clase).

PREGUNTA 1 (20 puntos)

El Ministerio de Transportes y Comunicaciones (MTC) en conjunto con el Grupo "Aeropuertos del Perú" (AdP) desea implementar un programa en JAVA que le permita registrar y gestionar la información relacionada a todos los vuelos aéreos que se dan lugar en los distintos aeropuertos de nuestro país. Para esto, lo ha contratado a Ud. con el objetivo de diseñar y realizar la implementación de este programa.

Con respecto a la lógica del negocio se puede mencionar lo siguiente:

Un aeropuerto tiene un nombre, una dirección y es de un determinado tipo ("NACIONAL" o "INTERNACIONAL"). Asimismo, un aeropuerto pertenece a una ciudad, la cual a su vez puede tener varios aeropuertos. De la ciudad se desea registrar únicamente el nombre de esta.

Por otro lado, dentro del funcionamiento aeroportuario, existen dos tipos de empresas: las operadoras y las aerolíneas. Ambos tipos de empresas poseen un nombre, pero también poseen datos específicos. Por ejemplo, las aerolíneas, además del nombre poseen un "*callsign*" que es el indicativo de una aerolínea usado durante una transmisión radial para referirse a ella. Por ejemplo: la aerolínea llamada "LATAM AIRLINES" utiliza como *callsign* "LATAM". Asimismo, la aerolínea llamada "SKY AIRLINES" utiliza como callsign "AEROSKY". Además del "callsign" se desea que el sistema permita indicar si la aerolínea tiene o no un programa de fidelidad.

Con respecto a las operadoras, estas guardan relación con los aeropuertos pues estas realizan la gestión del tráfico aéreo de un determinado aeropuerto. Es decir, cada aeropuerto tiene asignada una empresa operadora que realiza el monitoreo y gestión de los vuelos que se llevan a cabo en ese aeropuerto. Una misma operadora puede gestionar varios aeropuertos a la vez. De las operadoras, además de registrar el nombre, se desea registrar si es que ofrecen o no soporte logístico y si es que ofrecen o no asistencia de navegación.

Otro aspecto que también debería considerar el programa es la gestión de vehículos. Todo vehículo pertenece a una determinada aerolínea, y una aerolínea puede tener múltiples vehículos adquiridos. Los vehículos pueden ser de dos tipos: aviones y remolques. Todo vehículo es de un determinado modelo y tiene una velocidad máxima (que está dada en km/h). Sin embargo, los aviones además de estas características tienen una capacidad máxima de pasajeros que pueden llevar. Por otro lado, los remolques tienen una capacidad máxima de carga que pueden trasladar (en Kg). A modo de explicación, se puede mencionar que los remolques son unos autos pequeños esenciales en el funcionamiento aeroportuario, pues son los encargados de trasladar los aviones de un punto a otro del aeropuerto.

Con respecto a los vuelos se puede mencionar lo siguiente:

Un vuelo debe contar un identificador numérico correlativo que debe ser generado de forma automática por el sistema informático. Además, un vuelo posee un código que está conformado por 2 letras y 4 números (por ejemplo: "LA2210" o "SK5263"). Asimismo, está programado para una determinada fecha, y tiene una hora de salida y una hora de llegada (utilice las clases Date y LocalTime como tipo de dato para las fechas y horas respectivamente). Todo vuelo tiene asignado un avión (que realizará el mismo) y un remolque (que será el encargado del trasladar todo lo que sea necesario para asegurar el inicio del vuelo). Finalmente, se ha de mencionar que todo vuelo parte o sale desde un aeropuerto determinado y tiene como destino final o llegada otro aeropuerto.

Finalmente, se exige que tanto las empresas como los vehículos y los aeropuertos sean consultables, es decir, que permitan devolver a modo de cadena de caracteres algunos de sus datos. En este sentido, si se consultan los datos de una operadora, se devolverá la palabra "Operadora: ", seguido del nombre y la indicación de si ofrece soporte logístico más la indicación de si ofrece asistencia de navegación. Si se consultan los datos de una aerolínea, se devolverá "Aerolínea: ", seguido del nombre, el *callsign* y la indicación de si tiene programa de fidelidad. Si se consultan los datos de un aeropuerto, se devolverá "Aeropuerto: ", seguido del nombre del aeropuerto, el nombre de la ciudad a la que pertenece el aeropuerto, el tipo de aeropuerto y en una línea inferior, los datos de la operadora de ese aeropuerto. Si se consultan los datos de un avión, se devolverá "Avion: ", seguido del modelo, la capacidad máxima de pasajeros y en una línea inferior, los datos de la aerolínea dueña de ese avión. Si se consultan los datos de un remolque, se devolverá "Remolque: ", seguido del modelo y la capacidad máxima de carga.

Se ha realizado un análisis preliminar y se han detectado algunas clases de manera genérica:

- **TipoAeropuerto:** Clase *enumerate* que establece los valores para definir el tipo de aeropuerto.
- **Consultable:** Clase *interface* que define la obligación de consultar los datos de empresas, vehículos y aeropuertos.
- **Vuelo:** Clase que define a los vuelos. Dentro de sus principales funciones está la de devolver información detallada sobre el mismo según reporte.

Se le solicita elaborar un programa en JAVA que permita dar soporte a la lógica de negocio mencionada y además permita imprimir el reporte que se muestra en la Figura 02. El reporte debe verse igual.

Para validar el modelado de clases a realizar y como parte de las pruebas del sistema, se cuenta con el código fuente de la clase Principal cuya programación se encuentra definida en la Figura 01. La programación de la clase Principal debe ser exactamente igual al que se está colocando en la Figura 01.

```
import java.text.SimpleDateFormat;
import java.time.LocalTime;
class Principal{
	/* Coloque sus datos
	Nombre Completo:
	Código PUCP:
	*/
	public static void main(String[] args) throws Exception{
		//Se crea el objeto SimpleDateFormat
		SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
		//Se crean ciudades
		Ciudad ciudad01 = new Ciudad("Lima");
		Ciudad ciudad02 = new Ciudad("Chiclayo");
		//Se crean operadoras
		Operadora operadora01 = new Operadora("LIMA PARTNERS", true, true);
		Operadora operadora02 = new Operadora("CORPAC", false, true);
		//Se crean los aeropuertos
		Aeropuerto aeropuerto01 = new Aeropuerto("Jorge Chavez","Av. Elmer
Faucett S/N",TipoAeropuerto.Internacional,ciudad01,operadora01);
		Aeropuerto aeropuerto02 = new Aeropuerto("Jose Abelardo
Qui\u00f1ones","Av. Bolognesi
S/N",TipoAeropuerto.Internacional,ciudad02,operadora02);
		//Se crea la aerolinea
		Aerolinea aerolinea01 = new Aerolinea("Latam Airlines", "LATAM", true);
		//Se crea el avion
		Avion avion01 = new Avion(aerolinea01, "AIRBUS A320", 890.00, 186);
		//Se crea el remolque
		Remolque remolque01 = new Remolque(aerolinea01,"HD-ATT60", 158.00,
4160.00);
		//Se crea el vuelo
		Vuelo vuelo01 = new Vuelo("LA2023",sdf.parse("25-08-
2023"),LocalTime.of(13,00,00),LocalTime.of(14,30,00));
		//Establecemos el avion del vuelo
		vuelo01.setAvion(avion01);
```

```
            //Establecemos el remolque asignado al vuelo
            vuelo01.setRemolque(remolque01);
            //Establecemos el aeropuerto de salida del vuelo
            vuelo01.setAeropuertoSalida(aeropuerto01);
            //Establecemos el aeropuerto de llegada del vuelo
            vuelo01.setAeropuertoLlegada(aeropuerto02);
            //Se genera un reporte del vuelo
            String reporte = vuelo01.devolverInformacion();
            //Se imprime el reporte sobre el vuelo
            System.out.print(reporte);
        }
}
```

Fig. 01. Código fuente de la clase Principal

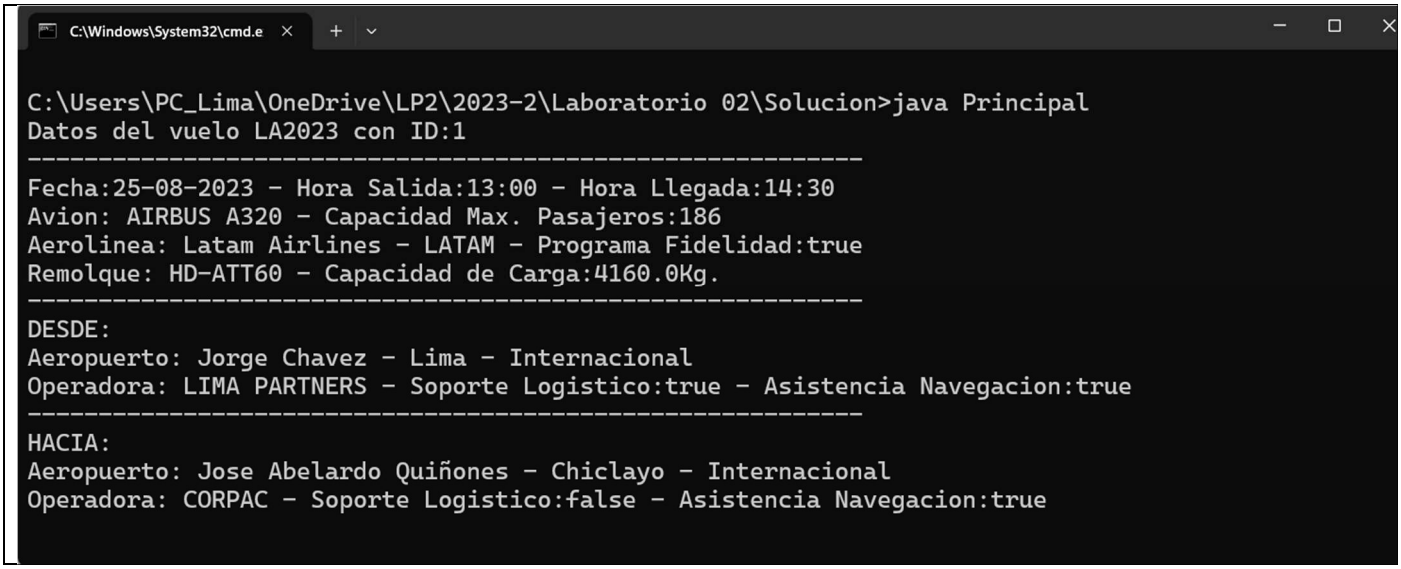

Fig. 02. Salida del sistema informático

Es indispensable que se encuentren programados todos los atributos de las clases y sus relaciones (como mínimo la programación de las relaciones que permiten la impresión del reporte). Con respecto a los constructores, *getters* y *setters*, puede programar solo aquellos que son requeridos para la salida del reporte.

Aspectos a considerar para evitar descuento de puntos:

• Colocar su nombre completo y código PUCP en la clase Principal.
• Nombrar correctamente a las clases, atributos y métodos.
• Utilice los principios de POO: abstracción (en clases y métodos donde sea requerido), polimorfismo y encapsulamiento.
• Utilice correctamente ámbitos, clases / métodos abstractas(os), clases de tipo interface, enumerados donde sea requerido.
• Colocar la etiqueta "@Override" donde corresponda.
• Respetar el orden en la estructura de una clase.
• Emplear un archivo por clase.
• El programa debe compilar correctamente, se descontarán puntos por errores de compilación.

Rúbrica de calificación:

(0.5 puntos) Correcta programación de la clase de tipo interface.
(0.5 puntos) Correcta programación de la clase enumerate.
(7 puntos) Correcta programación del método *consultarDatos()* en las clases donde es requerido el uso de la interface.
(8 puntos) Correcta programación de todas las clases con sus atributos, relaciones, constructos, *getters* y *setters* (respetando encapsulamiento, abstracción (métodos y clases), herencia, interfaces y enumerados).
(2 puntos) Correcta programación del método *devolverInformacion()* en la clase Vuelo.
(2 puntos) Correcta programación de la clase Principal y visualización del reporte.

Prof. Freddy Paz

24 de agosto del 2023