PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1
2do.Examen
(Segundo Semestre 2023)

Indicaciones Generales:

- Duración: 3 horas.

  SOLO ESTÁ PERMITIDO EL USO DE APUNTES DE CLASE. NO PUEDE UTILIZAR FOTOCOPIAS NI MATERIAL IMPRESO, TAMPOCO PODRÁ EMPLEAR HOJAS SUELTAS.

- No se pueden emplear **variables globales, estructuras. La clase (o el tipo de datos) string solo podrá utilizarse donde se le indique**. Tampoco se podrán emplear las funciones malloc, realloc, strdup o strtok, igualmente no se puede emplear cualquier función contenida en las bibliotecas stdio.h, cstdio o similares y que puedan estar también definidas en otras bibliotecas. **SOLO SE PODRÁ HACER USO DE PLANTILLAS Y STL EN AQUELLAS PREGUNTAS QUE ASÍ LO SOLICITEN, DE LO CONTRARIO SE ANULARÁ LA PREGUNTA.**
- Deberá modular correctamente el proyecto en archivos independientes. LAS SOLUCIONES DEBERÁN DESARROLLARSE BAJO UN ESTRICTO DISEÑO DESCENDENTE. Cada método **NO** debe sobrepasar las **20** líneas de código aproximadamente **(una línea de código no es una línea de texto, es una instrucción que termina con un punto y coma. LAS COMAS DEFINEN UNA LISTA DE INSTRUCCIONES POR LO QUE, SI EN UNA LÍNEA COLOCA INSTRUCCIONES SEPARADAS POR COMAS, SE CONTARÁ POR DOS CADA INSTRUCCIÓN)**.
- El archivo **main.cpp** solo podrá contener la función **main** de cada proyecto y el **código contenido en él solo podrá estar conformado por tareas implementadas como métodos de clases**. En el archivo main.cpp o en el archivo que contenga el método main, deberá colocar un comentario en el que coloque claramente su nombre y código, **de no hacerlo se le descontará 0.5 puntos en la nota final. (NO SE HARÁN EXCEPCIONES)**.
- El código comentado NO SE CALIFICARÁ **y esto incluye el comentar el llamado a la función que lo contiene**.
- La cláusula **friend** solo se podrá emplear en el caso de clases auto referenciadas para ligar el nodo con la clase **inmediata** que encapsula la lista, **en ningún caso adicional**. No se considerará en la nota las clases que violen esto. Tampoco se podrá emplear la cláusula **protected**.
- **Deberá mantener en todo momento el encapsulamiento de todos los atributos de las clases, así como guardar los estándares en la definición y uso de todas las clases desarrolladas.**
- Salvo en la sobrecarga de los operadores >> y <<, **no se podrán definir funciones (ni plantillas de funciones) independientes que no estén ligadas como métodos a alguna de las clases planteadas.**
- **Todos los métodos y funciones ligados a una clase deben estar desarrollados en el mismo archivo.**
- **NO PUEDE EMPLEAR ARREGLOS NI ARCHIVOS AUXILIARES PARA COLOCAR PARTE O TODOS LOS DATOS DE LOS ARCHIVOS DADOS**
- Los programas que presenten errores de sintaxis o de concepto se calificarán en base al 40% de puntaje de la pregunta. Los que no den resultados coherentes en base al 60%.
- Se tomará en cuenta en la calificación el uso de comentarios relevantes.
- **NO PUEDE UTILIZAR VARIABLES ESTÁTICAS.**

  SE LES RECUERDA QUE, DE ACUERDO AL REGLAMENTO DISCIPLINARIO DE NUESTRA INSTITUCIÓN, CONSTITUYE UNA FALTA GRAVE COPIAR DEL TRABAJO REALIZADO POR OTRA PERSONA O COMETER PLAGIO.

- **NO SE HARÁN EXCEPCIONES ANTE CUALQUIER TRASGRESIÓN DE LAS INDICACIONES DADAS EN LA PRUEBA**

Puntaje total: 20 puntos

INDICACIONES INICIALES

- La unidad de trabajo será **t:** (Si lo coloca en otra unidad, no se calificará su laboratorio y se le asignará como nota cero)
- Cree allí una carpeta con el nombre **"CO_PA_PN_EX_FINAL_2023_2"** donde **CO** indica: Código del alumno, **PA** indica: Primer Apellido del alumno y **PN** primer nombre (de no colocar este requerimiento se le descontará 3 puntos de la nota final). **Allí colocará los proyectos solicitados en la prueba.**

PREGUNTA 1 (6 puntos)

Crees una carpeta denominada "Pregunta01_Programa_En_JAVA", dentro de la carpeta anteriormente descrita, y en él desarrollará el programa que dé solución al problema planteado. DE NO COLOCAR ESTE REQUERIMIENTO SE LE DESCONTARÁ DOS (2) PUNTOS DE LA NOTA FINAL.

EN ESTA PREGUNTA SOLO PODRÁ UTILIZAR EL NotePad++ PARA DESARROLLAR EL PROGRAMA. DE EMPLEAR OTRO ENTORNO SE LE DESCONTARÁ LA MITAD DEL PUNTAJE

CADA CLASE SE DEBE DESARROLLAR EN UN ARCHIVO INDEPENDIENTE .java, SI EN UN ARCHIVO .java COLOCA DOS O MÁS CLASES ÉSTAS NO SERÁN CALIFICADAS.

DEBERÁ MANTENER EN TODO MOMENTO EL ENCAPSULAMIENTO DE LOS ATRIBUTOS, POR LO QUE DEBE DECLARAR TODO ATRIBUTO COMO PRIVADO, SINO SE DESCONTARÁ 0.5 EN CADA ACASO.

Una empresa de transporte de pasajeros maneja un archivo como los que se muestran a continuación:

```
MOG-547    Lopez/Reyes/Juan-Carlos     Lima  Ica  Arequipa  Moquegua  Tacna  10  50
A5R-175    Garcés/Carpio/Luciana       Lima  Huaraz  Chiclayo  5  20
…
FIN
P  78346512   Martel/Suarez/Patricia-Susana      Ayacucho  4  Almohada  Whisky  Gafas  Pantuflas
T  22990375   Quispe/Torres/José        Moquegua  S  N
…
```

En la primera parte del archivo viene los datos de la flota de ómnibus que partirán en una fecha determinada. En cada línea se ha colocado la placa y el nombre del chofer del vehículo, luego viene la lista de ciudades por las que pasará a dejar pasajeros y se termina con la cantidad de asientos para pasajeros de primera clase y para los de clase turista. La información de los vehículos termina con la palabra FIN.

En la segunda parte se coloca la lista de pasajeros que han comprado un boleto. En cada línea se ha colocado el tipo de pasajero (P: Primera clase, T: Clase turista), el DNI y el nombre del pasajero seguido de la ciudad de destino. Luego, si es un pasajero de primera clase viene la lista de artículos que desea que se le proporcionen durante el viaje. Si es un pasajero de clase turista, vienen dos letras (S o N) que indican si el pasajero llevará o no una valija y si desea o no que le proporcionen alimentos durante el viaje.

Se desea elaborar una aplicación en lenguaje JAVA que permita manejar la información de este archivo, la aplicación definirá las clases que se describen a continuación:

- **Clase Empresa**: la clase debe contener los siguientes atributos: 1) un atributo denominado **flota** (`ArrayList`) que contendrán los ómnibus que partirán en la fecha.

- **Clase Omnibus**: la clase debe contener los siguientes atributos: 1) un atributo denominado **placa** (`String`), 2) un atributo denominado **chofer** (`String`), 3) un atributo denominado **ruta** (`ArrayList`) que guardará las ciudades por las que pasará el bus, 4) un atributo denominado **asientosPC** (`int`) que guardará el número de asientos libres que tiene el bus para primera clase, 5) un atributo denominado **asientosCT** (`int`) que guardará el número de asientos libres que tiene el bus para la clase turista, 6) un atributo denominado **pasajeros** (`ArrayList de clase Pasajero`) que guarde los pasajeros que subirán al bus.

- **Clase Pasajero**: clase abstracta que debe contener los siguientes atributos: 1) un atributo denominado **dni** (`int`), 2) un atributo denominado **nombre** (`String`), 3) un atributo denominado **detino** (`String`)

- **Clase PrimeraClase**: que herede de la clase Pasajero, debe contener los siguientes atributos: 1) un atributo denominado **articulosSolicitados** (`ArrayList`).

- **Clase claseTurista**: que herede de la clase Pasajero, debe contener los siguientes atributos: 1) un atributo denominado **valija** (`Boolean`) que indique si llevara o no una maleta, 2) atributo denominado **almuerzo** (`Boolean`) que indique si comerá en el ómnibus, 3) atributo denominado **tarifaExtra** (`double`) que guarde lo que debe pagar extra antes de subir al ómnibus (Valija: 85.5, Almuerzo: 55.90).

"DEBE EMPLEAR OBLIGATORIAMENTE LOS NOMBRES DE LAS CLASES Y SUS ATRIBUTOS"
"NO PUEDE AGREGAR MÁS ATRIBUTOS A LAS CLASES"

La aplicación deberá leer, a través de la clase Empresa, las características de cada ómnibus que saldrá en la fecha y luego, también a través de la clase Empresa, los pasajeros que compraron un boleto. Se deberá colocar cada pasajero en un ómnibus que pase por la ciudad de destino del pasajero y que tenga asientos disponibles en su categoría. Finalmente, la aplicación deberá emitir, también a través de la clase Empresa, un reporte que muestre, de manera muy clara, la información completa de cada ómnibus que saldrán, con los datos completos de cada pasajero que llevará. La información debe aparecer en forma de ficha por cada ómnibus con una separación entre ellos.

Deberá definir por cada clase, los métodos selectores que se requieran y el constructor por defecto si se necesita. NO DEBE DEFINIR CONSTRUCTORES CON PARÁMETROS. También será obligatorio definir y utilizar métodos de lectura y escritura en cada clase que trabajen de manera polimórfica. NO SE PODRÁ LEER O IMPRIMIR DATOS QUE PERTENEZCA A OTRA CLASE.

PREGUNTA 2 (6 puntos)

Se solicita que desarrolle un proyecto **"ExamenFinal_PREG02"** dentro de la carpeta correspondiente, DE NO COLOCAR ESTE REQUERIMIENTO SE LE DESCONTARÁ 2 PUNTOS DE LA NOTA FINAL, en la cual se declaren las clases descritas con las relaciones necesarias, que permitan manipularlas empleando herencia:

- **Para manejar los pedidos realizados**: La clase se denominará "**Pedido**" y deberá contener lo siguiente: 1) un atributo denominado **codigo** (**string**) definido por una cadena dinámica de caracteres que representa el código del producto, 2) un atributo denominado **cantidad** (**int**) que representa la cantidad de productos solicitado, 3) un atributo denominado **peso** (**double**) que representa el peso de todo el pedido.
- **Para manejar los vehículos**: La clase se denominará "**Vehiculo**" y deberá contener lo siguiente: 1) un atributo denominado **dni** (**int**) que representa el código del cliente al cual pertenece el vehículo, 2) un atributo denominado **placa** (**string**) definido por una cadena dinámica de caracteres, 3) un atributo denominado **carga_maxima** (**double**) que representa el peso máximo que puede transportar un vehículo, 4) un atributo denominado **carga_actual** (**double**) que representa el peso que está transportando el vehículo de acuerdo a la carga que lleva, recuerde que nunca puede exceder a **carga_maxima**.
- **Para manejar los vehículos tipo camión**: La clase se denominará "**Camion**" y deberá contener lo siguiente: 1) un atributo denominado **ejes** (**int**) que representa la cantidad de ejes que tiene el vehículo, 2) un atributo denominado **llantas** (**int**) que representa la cantidad de llantas que tiene el vehículo, 3) un atributo denominado **depositos**, este es un STL-Vector de la clase **Pedido**, donde se guardarán los pedidos que el camión puede transportar, se sabe que cada camión puede llevar 5 pedidos como máximo, debe ignorar pedidos posteriores. Esta clase posee datos heredados de la clase **Vehiculo**.
- **Para manejar los vehículos tipo furgón**: La clase se denominará "**Furgon**" y deberá contener lo siguiente: 1) un atributo denominado **filas** (**int**) que representa la cantidad de filas que tiene el vehículo, 2) un atributo denominado **puertas** (**int**) que representa la cantidad de puertas que tiene el vehículo, 3) un atributo denominado **depositos**, este es un STL-List de la clase **Pedido**, donde se guardarán los pedidos que lleva el furgón. Debido al tipo de vehículo los pedidos se cargan ordenadamente por peso. Esta clase posee datos heredados de la clase **Vehiculo**.
- **Para manejar la flota de vehículos**: La clase se denominará "**Flota**" y deberá contener lo siguiente: 1) un atributo denominado **vehiculos**, este es un STL-Map de la clase **Vehiculos**, donde se guardarán todos los vehículos, el índice a utilizar será la placa, recuerde que los vehículos pueden ser Furgon o Camion.

"DEBE EMPLEAR OBLIGATORIAMENTE LOS NOMBRES DE LAS CLASES Y SUS ATRIBUTOS"

Con las clases indicas debe realizar las siguientes operaciones:

- En la clase **Flota** implementar el método **cargar_vehiculos,** que se encarga de la lectura del archivo "**Vehiculos.csv**" y cargar la información en el STL-Map denominado **vehiculos**. Para la lectura de los datos correspondiente a cada vehículo debe utilizar el método polimórfico **leer**, ya que los mismos

pueden ser furgones o camiones. Para diferenciar cada tipo de vehículo en el archivo correspondiente los furgones se representan con la letra "F" y los camiones con la "C".

- En la clase **Flota** implementar el método **mostrar_vehiculos,** que se encarga de realizar la impresión de un archivo debidamente tabulado (**sin usar el carácter '\t'**), que muestre todos los datos de cada vehículo de acuerdo con el tipo de unidad. Para este paso debe utilizar el método polimórfico **mostrar**. Antes del llenado del depósito del vehículo, debe mostrar el mensaje **"No hay pedidos para el cliente",** si es que el mismo NO cuenta con pedidos.
- En la clase **Flota** implementar el metodo **cargar_pedidos,** que se encarga leer el archivo **"Pedidos4.csv"** e insertar los pedidos en cada uno de los vehículos en sus respectivas STL pedidos. Para este paso debe utilizar el metodo polimórfico **insertar**. Considerando no sobrepasar la **carga_maxima** al añadir el pedido, para este control, cada vez que coloca un pedido debe actualizar el atributo **carga_actual**.

Para esta pregunta, por lo menos debe desarrollar los siguientes métodos polimórficos:

- **leer**: para la lectura de los datos de cada uno de los vehículos de acuerdo con su tipo.
- **insertar**: para la inserción de los pedidos en cada uno de los vehículos.
- **mostrar**: para la impresión de los datos de cada uno de los vehículos de acuerdo con su tipo.

Finalmente imprimir un reporte para el siguiente grupo de placas, para probar que su:

"A1Q-612", "D9A-711", "O8J-848", "Q5S-871", "Z4L-514"

Recuerde que está usando una estructura tipo STL-Map, se evaluará el correcto uso de su estructura.
El reporte debe tener el siguiente formato:

```
                    REPORTE DE FLOTA
============================================================
Codigo de Cliente:  52097922
Placa:              A1Q-612
Carga Maxima:       1000
Carga Actual:       20
#Llantas:           6
#Ejes:              2
Lista de Pedidos:
ISU.815   2.5       5
OWN.701   3         2
GBD.525   275       5
UTN.601   1.6       2
XSD.310   330       6
============================================================
Codigo de Cliente:  59407188
Placa:              D9A-711
Carga Maxima:       300
Carga Actual:       6
#Puertas:           3
#Filas:             2
Lista de Pedidos:
BIT.434   60        6
...
```

Consideraciones:
Para el desarrollo de esta pregunta considere el siguiente código:

```
#include "Utils.hpp"
#include "Flota.hpp"
int main() {
   Flota flota;
   flota.cargar_vehiculos();
   flota.cargar_pedidos();
   flota.mostrar_vehiculos();
   return 0;
}
```

NO PUEDE CAMBIAR ESTE CÓDIGO

PARTE03 (8 puntos):
Se solicita que desarrolle un proyecto **"ExamenFinal_PREG03"** dentro de la carpeta correspondiente, DE NO COLOCAR ESTE REQUERIMIENTO SE LE DESCONTARÁ 2 PUNTOS DE LA NOTA FINAL, en la cual se declaren las clases descritas con las relaciones necesarias, que permitan manipularlas empleando herencia:

➢ **Para manejar los Nodos**: La clase se denominará "**Nodo**" y deberá contener lo siguiente: 1) un atributo denominado **unidad** de la clase **Vehiculo**, 2) un atributo denominado **izq**, este atributo es un puntero a la clase **Nodo** (autoreferenciado), 3) un atributo denominado **der**, este atributo es un puntero a la clase **Nodo** (autoreferenciado).

➢ **Para manejar el árbol**: La clase se denominará "**Arbol**" y deberá contener lo siguiente: 1) un atributo denominado **raiz**, este atributo es un puntero de clase **Nodo**. Esta estructura se trata de un ABB ordenado por el dni (ascendente).

➢ **Para manejar la programación de vehículos**: La clase se denominará "**Programacion**" y deberá contener lo siguiente: 1) un atributo denominado **ADespachos**, este atributo es un objeto de la clase **Arbol**, donde se guardarán todos los vehículos de la flota, 2) un atributo denominado **fVehiculos**, este atributo es un objeto de la clase **Flota**.

"**DEBE EMPLEAR OBLIGATORIAMENTE LOS NOMBRES DE LAS CLASES Y SUS ATRIBUTOS**"

Con las clases indicas debe realizar las siguientes operaciones:

- En la clase **Programacion** debe implementar el método **cargavehiculos**, en el mismo se cargará la flota en el atributo **fVehiculos**, empleando su método **cargar_vehiculos**, luego se deben cargar los pedidos empleando su método **cargar_pedidos**. (0.5 puntos)
- En la clase **Programacion** debe implementar el método **cargaprogramacion,** el cual debe recorrer el mapa **vehiculos** de la **Flota fVehiculos** y cargar la información en el árbol ABB denominado **ADespachos**. Puede emplear punteros a **vehiculos** como parámetros en los métodos de inserción para mantener el polimorfismo. (3.0 puntos)
- En la clase **Programacion** implementar el método **reducevehiculos,** que se encargará de eliminar la cantidad de nodos que se recibe como parámetro, los nodos a eliminar del árbol **no deben tener hijos** para evitar aumentar la complejidad computacional del algoritmo. Para un borrado adecuado del nodo debe implementar el destructor de la clase **Nodo.** (4.0 puntos)
- En la clase **Programacion** debe implementar el método **muestraprogramacion**, en el mismo se cargará de mostrar el contenido del árbol **ADespachos en orden**, para esta tarea debe usar el método polimórfico mostrar de la clase **Vehículo.** (0.5 puntos)

Consideraciones:

Para el desarrollo debe considerar el siguiente código:

```
#include "Programacion.h"

using namespace std;
int main(int argc, char** argv) {
   Programacion pro;

   pro.cargavehiculos();
   pro.cargaprogramacion();
   pro.reducevehiculos(10);
   pro.muestraprogramacion();
}
```

NO PUEDE CAMBIAR ESTE CÓDIGO

Para esta pregunta puede emplear el código que ha desarrollado en la pregunta 2 o la biblioteca estática proporcionada. Para facilitar el acceso a la clase Flota desde la clase Programacion, puede definir una relación de amistad entre ellas.

Al finalizar el examen, comprima la carpeta de su proyecto empleando el programa Zip que viene por defecto en el Windows, **no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares**. Luego súbalo a la tarea programa en Paideia para este examen.

Profesor del curso: Rony Cueva
Erasmo Gómez
Miguel Guanira

San Miguel, 5 de DICIEMBRE del 2023.