PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1
2do.Examen
(Primer Semestre 2024)

Indicaciones Generales:

- Duración: 3 horas.

NO SE PERMITE EL USO DE APUNTES DE CLASE, FOTOCOPIAS NI MATERIAL IMPRESO

- 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 **"EX_FINAL_CO_PA_PN_2024_1"** 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 (7 puntos)

Se solicita que desarrolle un proyecto **"OrdenaPedidos"** 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**: La clase se denominará "**Pedido**" y deberá contener lo siguiente: 1) un atributo denominado **fecha** (`int`) que representa la fecha en que se realizó el pedido con el formato aaaammdd, 2) un atributo denominado **dni** (`int`) que representa el documento de identidad del cliente que realizó el pedido, 3) un atributo denominado **monto** (`double`) que representa el valor el pedido.
- **Para manejar los pedidos tipo envio**: La clase se denominará "**Envio**" y deberá contener lo siguiente: 1) un atributo denominado **direccion** (`char*`) definido por una cadena dinámica de caracteres que representa la dirección del cliente, 2) un atributo denominado **fecha** (`int`) que representa la fecha en que se realizará la entrega del pedido con el formato aaaammdd, 3) un atributo denominado **costo** (`double`) que representa el costo del envío. Esta clase posee datos heredados de la clase **Pedido**.
- **Para manejar los pedidos tipo recojo**: La clase se denominará "**Recojo**" y deberá contener lo siguiente: 1) un atributo denominado **tienda** (`char*`) definido por una cadena dinámica de caracteres que representa tienda donde se encuentra el pedido a recoger, 2) un atributo denominado **dias** (`int`) que representa los días que el pedido estará en la tienda. Esta clase posee datos heredados de la clase **Pedido**.
- **Para manejar las ordenes de pedidos**: La clase se denominará "**Orden**" y deberá contener lo siguiente: 1) un atributo denominado **pped,** este es un puntero de la clase **Pedido**, 2) un atributo denominado **sig**, este atributo es un puntero a la clase **Orden** (autoreferenciado).
- **Para manejar la Pila de ordenes**: La clase se denominará "**Pila**" y deberá contener lo siguiente: 1) un atributo denominado **nelementos** (`int`) que representa la cantidad ordenes que tiene almacenada la pila, 2) un atributo denominado **pcima**, este atributo es un puntero de clase **Orden**.
- **Para manejar el proceso**: La clase se denominará "**Procesa**" y deberá contener lo siguiente: 1) un atributo denominado **ppedidos** dado por un objeto de la clase **Pila**.

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

Consideraciones:

Para el desarrollo de la pregunta debe considerar el siguiente código:

```
#include "Procesa.h"

using namespace std;

int main(int argc, char** argv) {
   Procesa pro;

   pro.carga();
   pro.ordena();
   pro.muestra();

   return 0;
}
```

NO PUEDE CAMBIAR ESTE CÓDIGO

Con las clases indicas debe realizar las siguientes operaciones:

- **(3 puntos)** En la clase **Procesa** implementar el método **carga,** que se encarga de la lectura del archivo "**Pedidos.csv**" y colocará el pedido leído en el objeto **Orden**, para la lectura de los datos correspondiente de cada pedido debe utilizar el método polimórfico **lee**, ya que los mismos pueden ser **Envio** o **Recojo**. Para diferenciar cada tipo de pedido en el archivo correspondiente los envíos se representan con la letra "E" y los recojos con la "R". Con el objeto **Orden** cargado debe colocar el elemento en la Pila **ppedidos** incrementando el atributo **nelementos**. **Para esta pregunta debe implementar los métodos necesarios que utiliza una pila, como son: apilar, desapilar y pilavacia. Los métodos no deben recibir o devolver punteros, excepto las que manejan cadenas de caracteres.**

- **(3 puntos)** En la clase **Procesa** implementar el método **ordena**, que se encargará de **ordenar** las ordenes de la pila **ppedidos**, este método debe ordenar la pila de acuerdo con la fecha pedido de forma ascendente visto desde la cima.
- **(1 punto)** En la clase **Procesa** implementar el método **muestra**, que se encarga de realizar la impresión de un archivo debidamente tabulado (**sin usar el carácter '\t'**), que muestre los datos de cada **Orden**. Recuerde que es una pila por tal motivo no se puede recorrer en ningún momento. Para la impresión de los pedidos contenidos en cada **Orden** debe usar el método polimórfico **imprime**. Esta pregunta es válida solo se realizó el método carga de forma correcta. El reporte debe tener el siguiente formato:

```
          Reporte de Ordenes
          ==========================================
          Direccion:  Villa María del Triunfo
          Fecha Entrega: 2024/05/05  Costo: 5.00
          Fecha Pedido : 2024/05/01  DNI: 90367684
          Monto del Pedido: 1102.03

          Direccion:  Magdalena del Mar
          Fecha Entrega: 2024/05/05  Costo: 1.00
          Fecha Pedido : 2024/05/01  DNI: 63640178
          Monto del Pedido: 5610.00

          Tienda de Recojo: Mall Santa Anita
          Fecha Pedido: 2024/05/01    DNI: 77440060
          Monto del Pedido: 5866.26
...
```

Se recomienda revisar los archivos que servirán para la lectura de datos, los cuales se describen a continuación:

Pedidos.csv

```
E,20240501,90367684,1102.03,20240505, Villa María del Triunfo,5.00
R,20240504,59561864,4505.00,Jockey Plaza,5
E,20240511,45914393,1020.00,20240515, San Borja,1.50
…
```

Tipo, Fecha del pedido, DNI, Monto, Fecha Entrega/Local de recojo, Costo entrega/Dias de recojo

Recuerde que si no usa polimorfismo la respuesta no será válida, así mismo siempre debe mantener el encapsulamiento por lo cual debe realizar cada operación donde le corresponde, por ejemplo, si va a leer o imprimir un pedido esta operación la debe realizar en la clase que le corresponde. La cláusula *friend* solo se puede emplear para acceder desde la TAD al nodo (Orden).

Pregunta 2 (7 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 los conocimientos vistos en clas:

- **Para manejar los juegos**: La clase se denominará "**Juego**" y deberá contener lo siguiente: 1) un atributo denominado **codigo** (**string**) definido por objeto de la clase string que representa el código del producto, 2) un atributo denominado **titulo** (**string**) que representa el título del Juego, 3) un atributo denominado **desarrollador** (**string**) que representa el código del desarrollador del Juego, 4) **genero**(string) que representa el género del Juego, 5) **tema**(string) que representa el tema del Juego.
- **Para manejar los desarrolladores**: La clase se denominará "**Desarrollador**" y deberá contener lo siguiente: 1) un atributo denominado **codigo** (**string**) definido por objeto de la clase string que representa el código del desarrollador, 2) un atributo denominado **nombre** (**string**) que representa el nombre del desarrollador, 3) un atributo denominado **juegos** este es un STL-List de la clase **Juego**, donde se guardarán los juegos de la compañia desarrolladora.
- **Para manejar los usuarios**: La clase se denominará "**Usuario**" y deberá contener lo siguiente: 1) un atributo denominado **codigo** (**string**) definido por objeto de la clase string que representa el código del usuario, 2) un atributo denominado **nombre** (**string**) que representa el nombre del desarrollador, 3) un atributo denominado **juegos_comprados** este es un STL-Vector de la clase Juego, 4) un atributo denominado **cantidad_juegos_comprados**(int), 5) un atributo denominado **recomendaciones** este es un STL-Map, donde la llave es un objeto de la clase Juego y el valor es un STL-Vector de la clase Juego.
- **Para manejar la tienda de juegos**: La clase se denominará "**GameStore**" y deberá contener lo siguiente: 1) un atributo denominado **desarrolladores** este será definido por STL-Map, donde la llave

es un String que representa al código del desarrollador y el valor es un objeto de la clase Desarrollador, 2) un atributo denominado **juegos** este será definido por STL-Map, donde la llave es un String que representa al código del juego y el valor es un objeto de la clase Juego, 3) un atributo denominado **usuarios** que será definido por un STL-Vector de la clase Usuario.

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

Con las clases indicas debe realizar las siguientes operaciones:

- En la clase **GameStore,** implementar el método **cargar_juegos,** que se encarga de la lectura del archivo "**Juegos.csv**" y cargar la información en el STL-Vector denominado **juegos**.
- En la clase **GameStore,** implementar el método **cargar_desarrolladores,** que se encarga de la lectura del archivo "**Desarrolladores.csv**" y cargar la información en el STL-Map denominado **desarrolladores**. Recuerde completar la STL-List **juegos** con la ayuda del vector juegos.
- En la clase **GameStore**, implementar el método **cargar_usuarios,** que se encarga de la lectura del archivo "**Usuarios.csv**" y carga la información del STL-Vector denominado **usuarios**. Este método deberá encargarse también de calcular las recomendaciones de cada usuario y llenar la estructura **recomendaciones**. Siguiendo las siguientes consideraciones
    - Un juego es recomendado si comparte el mismo Desarrollador.
    - Un juego es recomendado si son del mismo Género.
    - Un juego es recomendado si tratan el mismo Tema.
    - Recuerde que NO debe recomendar el mismo juego.
- En la clase GameStore, implementar un método que permita mostrar el siguiente reporte:

```
                    REPORTE DE RECOMENDACIONES
============================================================
Codigo de Usuario:  ABC1234  Nombre: Erasmo Gomez
Juego Comprado 1) Pokemon Legend: Arceus
Juegos Recomendados: Pokemon Scarlet
                     Legend of Zelda: Breath of the Wild
Juego Comprado 2) Pokemon Scarlet
Juegos Recomendados: Pokemon Legend: Arceus
Juego Comprado 3) GTA: San Andreas
Juegos Recomendados: Legend of Zelda: Breath of the Wild
Juego Comprado 4) Legend of Zelda: Breath of the Wild
Juegos Recomendados: Pokemon Legend: Arceus
                     GTA: San Andreas
============================================================
...
```

Consideraciones:

Para el desarrollo de esta pregunta considere el siguiente código:

```
#include "GameStore.hpp"

int main() {
   GameStore gamestore;
   gamestore.cargar_juegos("Juegos.csv");
   gamestore.cargar_desarrolladores("Desarrolladores.csv");
   gamestore.cargar_usuarios("Usuarios.csv");
   gamestore.mostrar_recomendaciones("Reporte.txt");
   return 0;
}
```

NO PUEDE CAMBIAR ESTE CÓDIGO

PREGUNTA 3 (6 puntos)

Crea una carpeta denominada "Pregunta03_Programa_En_Java", dentro de la carpeta anteriormente descrita, y en ella desarrollará el programa que dé solución al problema planteado. DE NO COLOCARSE 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.

Un restaurante maneja un archivo como el que se muestra a continuación:

```
R TER-1004  Terraza        2   María-Gómez
R SAL-1004  Salón          6   Ana-Martínez
P SP1-1001  Salón-Elite  10  Laura-Sánchez Alejandro-Díaz  3  Televisor Bar Audio
…
FIN
R  98765432  Sergio-Díaz       T  2   21.5  N
C  21098765  Hugo-Sánchez  T  2  19.5   A   2  Nueces Trigo
P  54101098  Hugo-Pérez         10          G  Televisor Bar Audio  3
…
```

En la primera parte del archivo vienen los datos de las mesas del restaurante que están disponibles en una fecha determinada. En cada línea se ha colocado el tipo de mesa (R: Regular, P: Premium), el código de la mesa, la ubicación de la mesa (Terraza, Salon, Salon-Elite, Salón-Ejecutivo, Salon-VIP), el nombre del camarero(s) asignado(s) a la mesa y la capacidad (cantidad de personas) de la mesa. Para las mesas Premium también se ha colocado la cantidad y los nombres de las amenidades proporcionadas. La información de las mesas termina con la palabra FIN.

En la segunda parte se coloca la lista de solicitudes de reserva realizadas. En cada línea se ha colocado el tipo de solicitud (R: Regular, C: Con Restricciones, P: Premium), el DNI y el nombre de la persona que realizó la reserva, la ubicación deseada (T: Terraza, S: Salón), la cantidad de personas, la hora de la reserva y la ocasión (N: No Aplica, C: Cumpleaños, A: Aniversario, G: Graduación). Hay que tener en cuenta que el restaurante solo cuenta con dos turnos, el primer turno inicia a las 19:30 pm. y el segundo turno a las 21:30 pm., la hora de las solicitudes solo podrán tener los valores 19.5 o 21.5. En el caso de las solicitudes de reservas Premium no se coloca la ubicación deseada ni la hora ya que estas solicitudes serán asignadas a los salones privados (Salon-Elite, Salon-Ejecutivo o Salon-VIP) según disponibilidad y estas mesas son reservadas desde la apertura hasta el cierre del restaurante (único turno), la hora de estas solicitudes siempre será 19:30 pm., también se han colocado el nombre y la cantidad de las amenidades solicitadas.

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

- **Clase Restaurante**: la clase debe contener los siguientes atributos: 1) **mesas** (`ArrayList`) que contendrá las mesas disponibles en el restaurante 2) **solicitudesReservas** (`ArrayList`) que contendrá las solicitudes de reserva del restaurante.
- **Clase Mesa**: clase abstracta que debe contener los siguientes atributos: 1) **codigo** (`String`), 2) **ubicacion** (`String`), 3) **capacidad** (`int`), 4) **ocupadaPrimerTurno** (`boolean`) que especificará si la mesa ya se encuentra ocupada para el primer turno.
- **Clase MesaRegular**: que herede de la clase Mesa, debe contener los siguientes atributos: 1) **camarero** (`String`) que contendrá el nombre del camarero asignado a la mesa, 2) **ocupadaSegundoTurno** (`boolean`).
- **Clase MesaPremium**: que herede de la clase Mesa, debe contener los siguientes atributos: 1) **camareros** (`ArrayList`) que contendrá los nombres de los camareros asignados a la mesa, 2) **amenidades** (`ArrayList`) que contendrá la lista de amenidades que ofrece la mesa, 3) **numeroDeAmenidades** (`int`).
- **Clase SolicitudReserva**: clase abstracta que debe contener los siguientes atributos: 1) **dni** (`int`) que contendrá el DNI de la persona que realizó la reserva, 2) **nombre** (`String`) que contendrá el nombre de la persona que realizó la reserva, 3) **ubicacionDeseada** (`char`) contendrá la ubicación en el restaurante que prefiere el cliente (T: Terraza, S: Salon, E: Para solicitudes Premium), 4) **cantidadPersonas** (`int`) contendrá la cantidad de personas en la reserva, 5) **hora** (`double`) que contendrá la hora de la reserva, 6) **ocasion** (`char`) (N: No Aplica, C: Cumpleaños, A: Aniversario, G: Graduación), 7) **mesaAsignada** (`String`) que contendrá el código de la mesa que se asignó a la reserva, 8) **estado** (`char`) (R: Rechazada, A: Aprobada).
- **Clase SolicitudReservaRegular**: que herede de la clase SolicitudReserva, esta clase no cuenta con atributos adicionales, pero tener en cuenta que sobre escribe e implementa métodos polimórficos.

- **Clase SolicitudReservaConRestricciones**: que herede de la clase SolicitudReserva, debe contener los siguientes atributos: 1) **alergias** (`ArrayList`) que contendrá las restricciones alimentarias para tener en cuenta en la reserva, 2) **numeroDeAlergias** (`int`) número de restricciones alimentarias.
- **Clase SolicitudReservaPremium**: que herede de la clase SolicitudReserva, debe contener los siguientes atributos: 1) **amenidades** (`ArrayList`) que contendrá de lista de amenidades que se requieren en la reserva, 2) **numeroDeAmenidades** (`int`). Las reservas Premium no cuentan con ubicación deseada dado que son asignadas a una mesa en un salón privado en este caso se usará E: Elite, así como para la hora de la reserva, una reserva Premium no tiene hora de llega ya que la mesa está disponible desde la hora de apertura hasta la hora de cierre del restaurante, se debe usar un valor de 19.5 (19:30 pm.).

"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 Restaurante, las características de cada mesa, así como las características de cada solicitud de reserva. Se deberá asignar a cada reserva una mesa de acuerdo a las características de la reserva, una reserva regular deberá ser asignada a una mesa que se encuentre disponible en Terraza o Salón, teniendo en cuenta la capacidad de la mesa la cual no se puede exceder, en caso ya no queden mesas disponibles en la ubicación deseada en la solicitud de reserva se asignará una mesa en otra ubicación (terraza o salón).

La asignación de mesas debe tener en cuenta la hora de la solicitud de reserva ya que para el caso de mesas regulares se manejan dos turnos (una mesa puede ser asignadas a dos solicitudes si estas se encuentran en diferentes horarios). En el caso de una reserva Premium se asignará una mesa ubicada en un salón privado teniendo en cuenta la capacidad de la mesa y la que cumpla con las amenidades solicitadas.

En caso no se pueda encontrar una mesa para una reserva (porque ya no hay mesas disponibles) la mesa asignada en la solicitud de reserva tendrá un valor de null y el estado de la solicitud se colocará como R: Rechazado. Finalmente la aplicación deberá emitir un reporte también a través de la clase Restaurante donde se muestre muy claramente los datos de las solicitudes de reserva incluyendo la mesa que fue asignada, el camarero o camareros asignados, y las características de la reserva: dni, nombre, hora (en formato HH:MM pm.), ocasión (cumpleaños, aniversario, graduación), cantidad de personas, y para el caso de reservas con restricciones a lista de las restricciones y para el caso de reservas Premium la lista de amenidades que fueron solicitadas.

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 PERTENEZCAN A OTRA CLASE.**

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, Eric Huiza, Heider Sánchez

San Miguel, 9 de julio del 2024.