Facultad: Ingeniería
Escuela: Ingeniería en Computación
Asignatura: Programación Orientada a Objetos G03L - CICLO II 219

Guía de ejercicios práctica

Fecha de entrega: Sábado 09 Noviembre 11.50 (AulaVirtual)

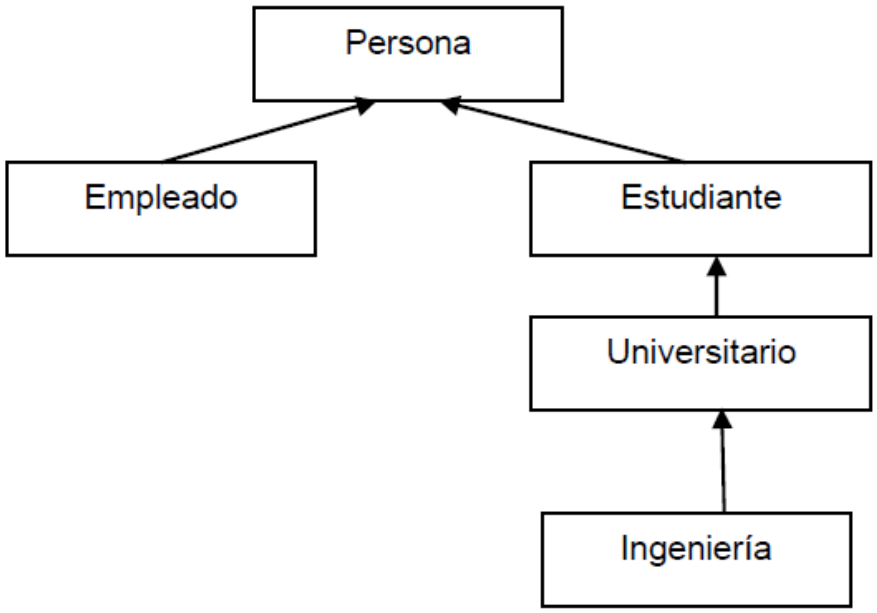
Indicaciones:

- Usando modo grafico
- Guía en pareja
- Utilizar la programación Orientada a Objetos. (POO)
- Códigos similares o extraídos del internet, anula guía.
- El sistema será compartido por aula Virtual.
- Utilizar base de datos queda opcional, si pueden utilizar variables locales para guardar información.
- Se debe de enviar un archivo de texto con los nombres y el porcentaje de la guia(100% , 80% , 70% , etc)

Desarrollo:

Ejercicio 1:

Construya una solución para la jerarquía de clases mostrada en la siguiente figura:



EN ENTORNO GRÁFICO desarrolle:

Para la clase Estudiante, considere los atributos: número de carnet, nivel de estudios.
Para la clase Universitario considerar los atributos: nombre de la universidad, carrera, materias inscritas, notas, CUM.

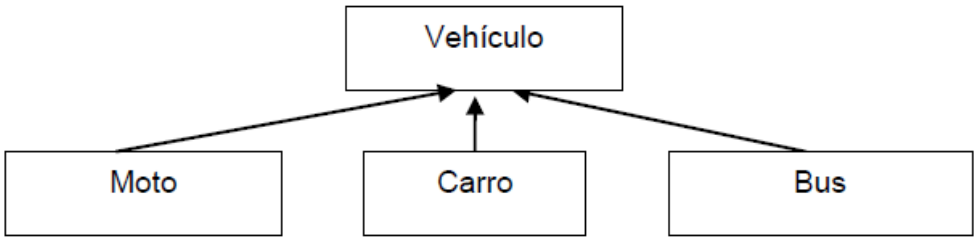
Para la clase Ingeniería considerar los atributos: nombre del proyecto, total de horas (duración de la pasantía), número de horas completadas.

La solución se debe manejar a través de un menú (botones, opciones, combobox u otros) que permita realizar las siguientes acciones:

a) Crear los objetos de tipo Ingeniería, solicitando los datos al usuario.
b) Verificar que universidad proporciona la mayoría de estudiantes de ingeniería
c) Visualizar el promedio de notas de los estudiantes
d) Salir de la aplicación.

Ejercicio 2:

Considere la siguiente jerarquía de herencias:



Definir las clases. Considerar todas las propiedades y funciones necesarias para una implementación completa haciendo uso de funciones virtuales y polimorfismo. Decidir qué atributos y métodos incluir en cada clase de tal manera que su programa realizar como mínimo las siguientes acciones:

a) Crear objetos de cualquier tipo, a excepción de la clase base que NO debe tener objetos (asegúrese de ello), solicitando los datos al usuario.

b) Visualizar un objeto en particular, con todos sus atributos.

Ser creativos con la solución y el programa debe estar debidamente comentado.

Ejercicio 3:

Nos piden hacer un programa orientado a objetos sobre un cine (solo de una sala) tiene un conjunto de asientos (8 filas por 9 columnas, por ejemplo).

- Del cine nos interesa conocer la **película** que se está reproduciendo y el **precio** de la entrada en el cine.

- De las películas nos interesa saber el **título**, **duración**, **edad mínima** y **director**.

- Del espectador, nos interesa saber su **nombre**, **edad** y **el dinero que tiene.**

- Los asientos son etiquetados por una letra (columna) y un número (fila), la fila 1 empieza al final de la matriz como se muestra en la tabla. También deberemos saber si está ocupado o no el asiento.

8 A 8 B 8 C 8 D 8 E 8 F 8 G 8 H 8 I
7 A 7 B 7 C 7 D 7 E 7 F 7 G 7 H 7 I
6 A 6 B 6 C 6 D 6 E 6 F 6 G 6 H 6 I
5 A 5 B 5 C 5 D 5 E 5 F 5 G 5 H 5 I
4 A 4 B 4 C 4 D 4 E 4 F 4 G 4 H 4 I
3 A 3 B 3 C 3 D 3 E 3 F 3 G 3 H 3 I
2 A 2 B 2 C 2 D 2 E 2 F 2 G 2 H 2 I
1 A 1 B 1 C 1 D 1 E 1 F 1 G 1 H 1 I

Realizaremos una pequeña simulación, en el que ingresamos muchos espectadores y los sentaremos aleatoriamente (no podemos donde ya este ocupado).

En esta versión sentaremos a los espectadores de uno en uno.

Solo se podrá sentar si tienen el suficiente **dinero**, hay espacio libre y tiene edad para ver la película, en caso de que el asiento este ocupado le buscamos uno libre.

Ejercicio 4:

realizar una clase llamada Raices, donde representaremos los valores de una ecuación de 2º grado.

Tendremos los 3 coeficientes como atributos, llamémosles a, b y c.
Hay que insertar estos 3 valores para construir el objeto.
Las operaciones que se podrán hacer son las siguientes:

- obtenerRaices(): imprime las 2 posibles soluciones
- obtenerRaiz(): imprime única raíz, que será cuando solo tenga una solución posible.
- getDiscriminante(): devuelve el valor del discriminante (double), el discriminante tiene la siguiente formula, (b^2)-4*a*c
- tieneRaices(): devuelve un booleano indicando si tiene dos soluciones, para que esto ocurra, el discriminante debe ser mayor o igual que 0.
- tieneRaiz(): devuelve un booleano indicando si tiene una única solución, para que esto ocurra, el discriminante debe ser igual que 0.
- calcular(): mostrara las posibles soluciones que tiene nuestra ecuación, en caso de no existir solución, mostrarlo también.
- Formula ecuación 2º grado: (-b±√((b^2)-(4*a*c)))/(2*a)

Solo varia el signo delante de -b