TC2016 – Programación Orientada a Objetos
Proyecto # 3

Forma de Trabajo : Individual
Ponderación : 15 puntos del puntaje final de proyectos

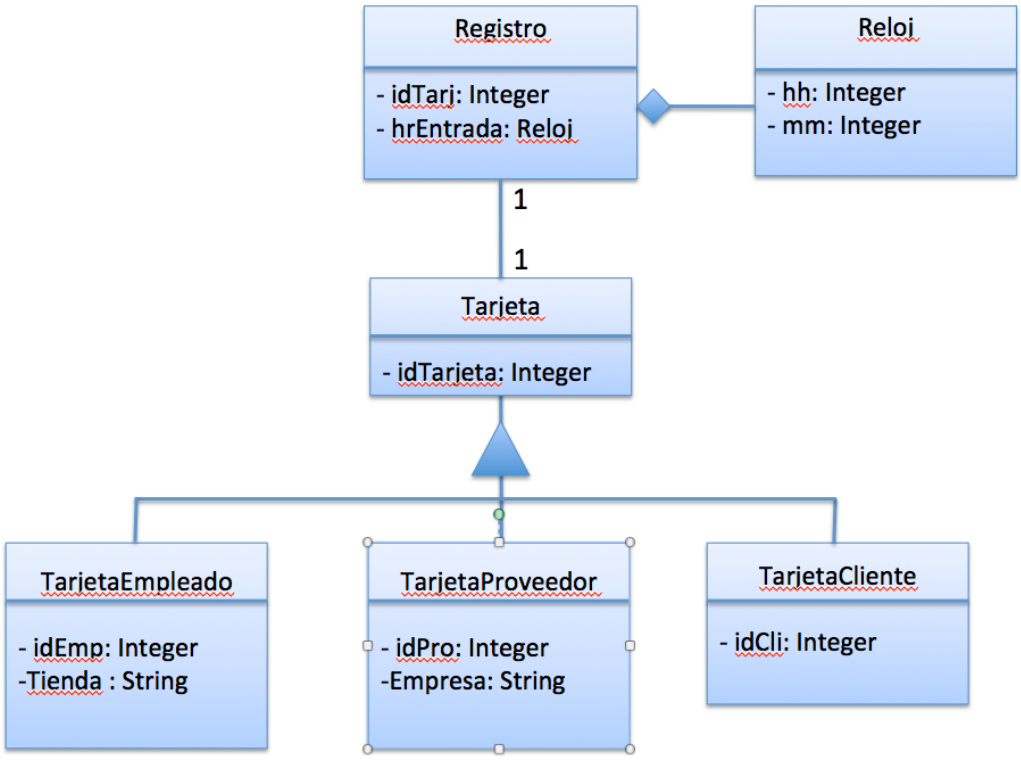
Entrega final	**Fecha de Entrega:** Domingo 5 de Mayo hasta las 11:59 PM **Forma de Entrega:** Subir a Blackboard en la sección Assignments en la liga Proyecto 3 los archivos .cpp y .h del proyecto.

Descripción del programa a realizar:

Se trata de un estacionamiento privado en el que se pueden estacionar los empleados, proveedores y clientes de un negocio. El horario del estacionamiento es de 8:00 a 22:00 hrs y no se pueden quedar carros en la noche.

Descripción de las clases:

Escribe las clases **Registro**, **Reloj**, **Tarjeta**, **TarjetaEmpleado**, **TarjetaProveedor** y **TarjetaCliente** de acuerdo con el siguiente diagrama:



NOTAS:

- Las clases **Registro** y **Reloj** deben tener un constructor default y un constructor con todos los parámetros; así como métodos de acceso y modificación para cada uno de sus atributos.
- La clase **Reloj** debe utilizar sobrecarga de operadores al menos para los operadores:
    - **>=** y/o **<=** que se usará para verificar que la hora de entrada al estacionamiento no sea posterior a la hora de salida.
    - **–** que servirá para calcular la diferencia entre la hora de salida y la hora de entrada.
    - **>>** y **<<** para utilizar objetos de tipo Reloj con cin y cout.
- La clase **Tarjeta** es una clase abstracta que tendrá los métodos:
    - **calculaPago** que recibe como parámetro la hora de entrada, la hora de salida del estacionamiento y la tarifa por hora. Este método debe ser **abstracto**.
    - **calculaHoras** que recibe como parámetro la hora de entrada y la hora de salida del estacionamiento y regresa como valor de retorno las horas que se deben cobrar al auto. Debe regresar un número de entero. Se cobrará la hora a partir de 15 minutos, por ejemplo si estuviste 1 hr con 15 min se cobrarán 2 hrs y si estuviste 1 hr con 10 min se cobrará 1 hr.
    - **muestraDatos** que no recibe parámetros ni tiene valor de retorno y muestra todos los datos del objeto. Este método debe ser **abstracto**.
- Las clases **TarjetaEmpleado, TarjetaProveedor y TarjetaCliente** que son clases derivadas de la clase **Tarjeta**.
  Estas clases deben tener constructor default y constructor con todos los parámetros, métodos de acceso y modificación para todos sus atributos y deben implementar los métodos abstractos de la clase padre.
- La clase **TarjetaEmpleado** debe implementar **calculaPago** considerando que los empleados tienen estacionamiento gratuito.
- La clase **TarjetaProveedor** debe implementar **calculaPago** considerando que los proveedores tienen 1 hr de cortesía cada vez que usan el estacionamiento.
- La clase **TarjetaCliente** debe implementar **calculaPago** de manera tradicional, es decir horasACobrar * tarifa.

Descripción del Proyecto:

1. Escribe un programa que tenga un arreglo de objetos de tipo **Registro**, que será el registro de los autos que están en el estacionamiento con capacidad máxima de 20 y un arreglo de **apuntadores a objeto de tipo Tarjeta** que tendrá la lista de todas las tarjetas autorizadas para usar el estacionamiento con una capacidad máxima de 40.

2. Carga los datos de las tarjetas de un archivo llamado "**tarjetas.txt**" que tiene el siguiente formato, considera que la cantidad de renglones del archivo puede variar:

```
E   101   123   Sanborns
C   102   456
P   103   169   Office Depot
C   104   321
P   105   987   Sabritas
```

   El primer dato de tipo caracter es el **tipo de Tarjeta**, el segundo dato que es un entero es el **id de la tarjeta**, el tercer dato que es un entero es el **id específico** (de empleado, de proveedor o de cliente) y si es un empleado o un proveedor enseguida viene un string de una o más palabras que contiene la **tienda** o **empresa** correspondiente.

3. Tu programa tiene un menú con las siguientes opciones:
   a. **Entrada al estacionamiento**
      En esta opción pide al usuario:
      - el id de la tarjeta del auto que entra, verifica que sea un id de tarjeta existente
      - la hora a la que entra al estacionamiento, verifica que la hora sea válida

      Esta información se debe agregar al arreglo de objetos de tipo Registro.

   b. **Salida del estacionamiento**
      En esta opción pide al usuario:
      - el id de la tarjeta del auto que sale, verifica que sea un id de tarjeta que se encuentre en el estacionamiento; es decir, no puede salir un auto que no esté en el estacionamiento.
      - la hora a la que sale el auto, verifica que la hora sea válida y además que la hora de salida sea posterior a la hora de entrada del auto.

      En esta opción debes mostrar un mensaje que indique a qué hora entró el auto, a qué hora está saliendo, cuantas horas se cobrarán y el precio que debe pagar.

   c. **Consulta del Estacionamiento**
      Esta opción muestra en la pantalla todos los autos que se encuentran en el estacionamiento, debe mostrar un mensaje que indique si se trata de un empleado, un proveedor o un cliente y que muestre todos los datos de la tarjeta correspondiente.
      Por ejemplo:
      CLIENTE 456
      Tarjeta 102

      PROVEEDOR 987
      Tarjeta 105

Empresa: Sabritas

EMPLEADO 123
Tarjeta 101
Tienda Sanborns

d. Consulta de Tarjetas
Muestra en la pantalla todas las tarjetas que se encuentran en el arreglo de Tarjetas con todos sus datos.

e. Salir