Programación 4

SOLUCIÓN EXAMEN FEBRERO 2011

Problema 1 (35 puntos)

Usted es el responsable del relevamiento para un sistema de Atención al Cliente con el fin de brindar registro y seguimiento de incidentes en los productos de una empresa de software.

Los clientes actuales son empresas o particulares, pudiendo incorporarse nuevos tipos como ser organismos o instituciones. En particular, de las empresas interesa registrar su RUT. Cada cliente se identifica con un código y se le ofrece soporte únicamente sobre los productos que ha adquirido, debiendo indicar su licencia con el número de serie correspondiente. Se asume que un cliente no puede adquirir más de una licencia de un mismo producto.

De cada incidente se registra la fecha, una descripción y estado actual, debiendo permitir identificar el producto y cliente correspondientes. Además se le asigna un nivel de complejidad. Los niveles se identifican con una letra que pondera la complejidad correspondiente y poseen una descripción (por ejemplo: básico, medio, avanzado). Para la resolución de un incidente se asigna un grupo de técnicos certificados tanto en el producto como en el nivel de complejidad. Un grupo está compuesto por exactamente 5 técnicos con uno de ellos como responsable. Para cada técnico, se conocen las distintas certificaciones que posee.

Se pide:

i. Modelar la realidad planteada mediante un Diagrama de Modelo de Dominio UML.

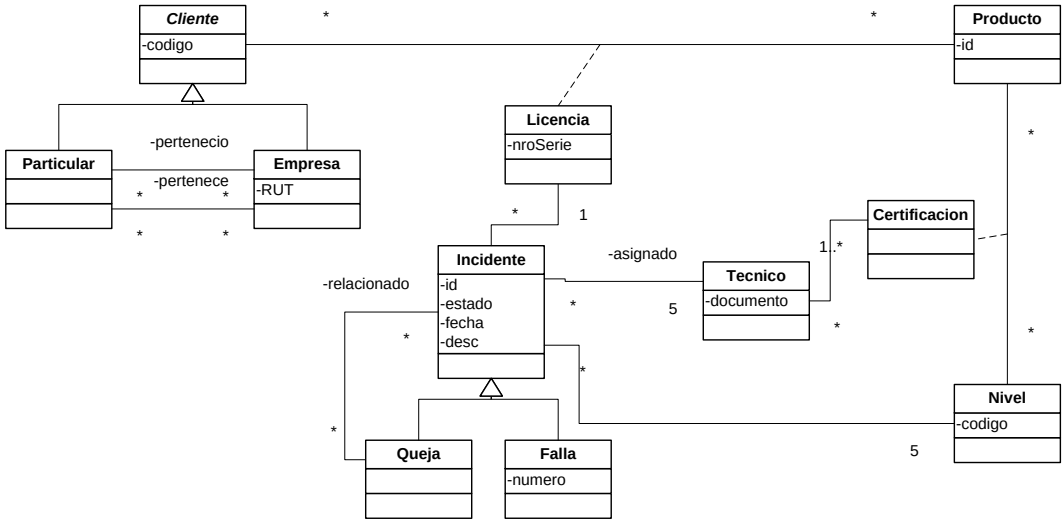
ii. Expresar todas las restricciones del modelo en **lenguaje natural**.

iii. Por motivos comerciales es de interés saber si los clientes particulares pertenecen a alguna empresa. ¿Qué modificaciones haría al modelo de la parte i?

iv. Un nuevo requerimiento establece la necesidad de discriminar algunos tipos de incidentes particulares: quejas y fallas, debiendo registrar el número de error para estos últimos. También se desea poder llevar registro de los incidentes relacionados a una queja. ¿Qué modificaciones haría al modelo de la parte i?

Solución:

Partes i, iii y iv.



Parte ii.

- Unicidad de Clientes, Técnicos, Productos, Niveles e Incidentes.
- Todo técnico asignado a un incidente posee una certificación para el producto y nivel del incidente.

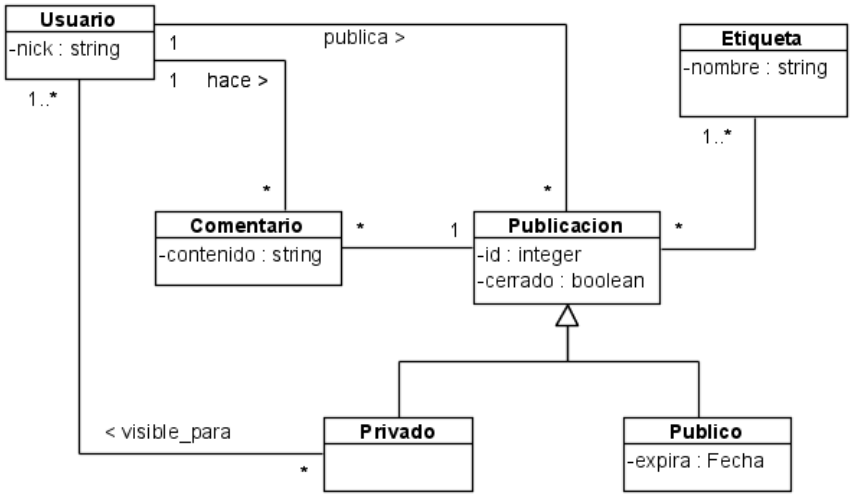
Problema 2 (35 puntos)

Usted ha sido asignado para realizar el diseño de un caso de uso en el marco del desarrollo de un sistema de gestión de un sitio web de foros de noticias. El sitio permite a los usuarios ver y publicar contenidos clasificados de acuerdo a determinados temas (denominados etiquetas) y hacer comentarios de las publicaciones.

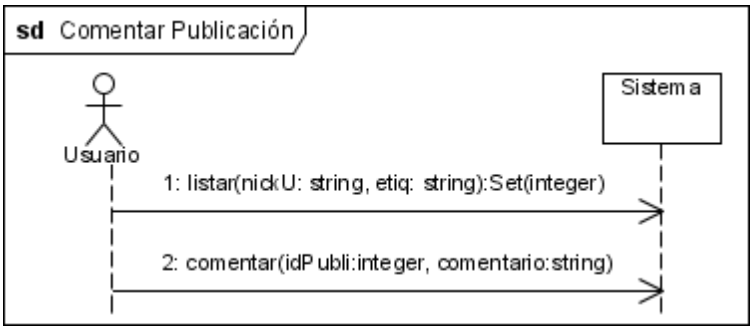
El sistema mantiene un conjunto de las publicaciones hechas por los usuarios. Cada publicación se identifica por su id y se conoce el usuario que la publicó. Hay dos tipos de publicaciones: las privadas y las públicas. Las publicaciones privadas sólo pueden ser vistas por un conjunto reducido de usuarios, por ejemplo el grupo de amigos, conocidos del usuario, etc., para controlar su visibilidad en el sitio web. Las publicaciones públicas pueden ser vistas por todos los usuarios. Sin embargo, no pueden ser vistas luego de su fecha de expiración, por una política del sitio web. Las publicaciones, independientemente de su tipo, pueden ser cerradas en cualquier momento y pasan a no ser visibles por ningún usuario.

Un usuario solamente puede ver y hacer comentarios en aquellas publicaciones que son visibles para él.

Como resultado de la etapa de análisis se llegó al siguiente modelo de dominio.



Además se dispone del Diagrama de Secuencia de Sistema (DSS) para el caso de uso *Publicar Comentario* junto al contrato de sus operaciones.



Operación	`listar(nickU: string, nomEtiq: string):Set(integer)`
Descripción	Devuelve la lista de identificadores correspondientes a las publicaciones visibles

	para un usuario que tengan cierta etiqueta.
Pre y postcondiciones	
def: Sea **fechaActual** la fecha actual del sistema. def: Sea **colRes** la colección de instancias de Publicacion **p** asociadas a la Etiqueta con nombre = nomEtiq tales que: **a)** Si **p** es una instancia de Publico entonces p.expira es posterior a **fechaActual.** b) Si **p** es una instancia de Privado, entonces Usuario con Nick = nickU está asociado a través de visible_para con **p.** c) **p.**cerrado = False. pre: Existe una instancia de Usuario con nick = nickU. pre: Existe una instancia de Etiqueta con nombre = nomEtiq. post: Se retorna una colección de p.id para cada **p** en **colRes.** post: El sistema recuerda la instancia de Usuario con nick = nickU en la memoria del sistema.	

Operación	comentar(idPubli: integer, comentario: string)
Descripción	Agrega un comentario a la publicación.
Pre y postcondiciones	
pre: Existe una Publicacion con id = idPubli. pre: Existe un Usuario recordado. post: Se crean: una instancia **c** de Comentario con atributo contenido = comentario, un link entre **c** y la instancia de Usuario recordada por el sistema y otro link entre **c** y la Publicacion con id = idPubli.	

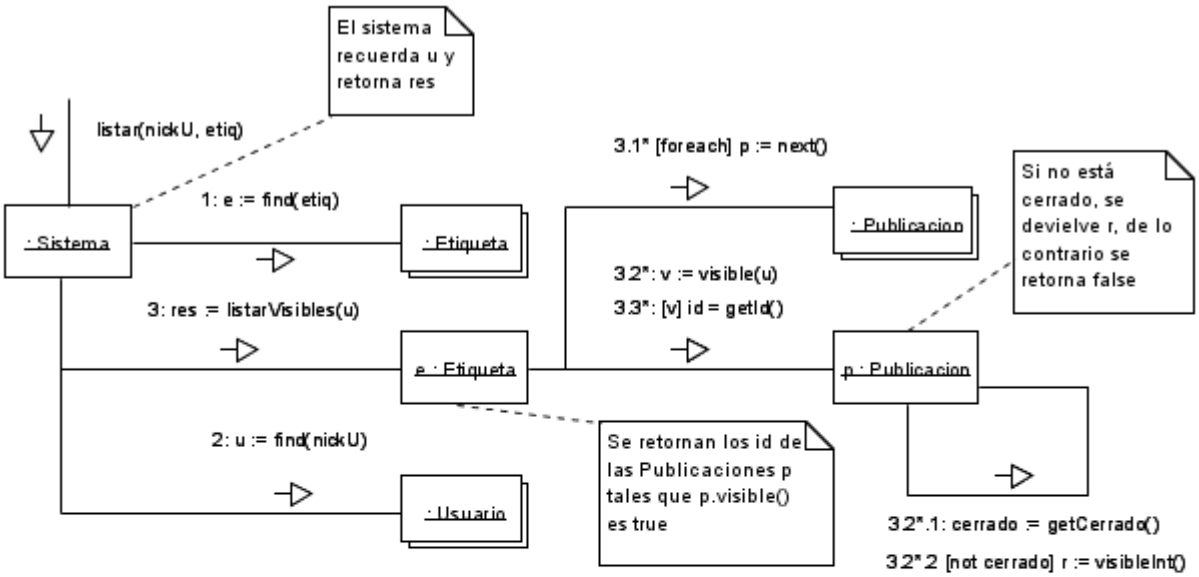
Se pide:

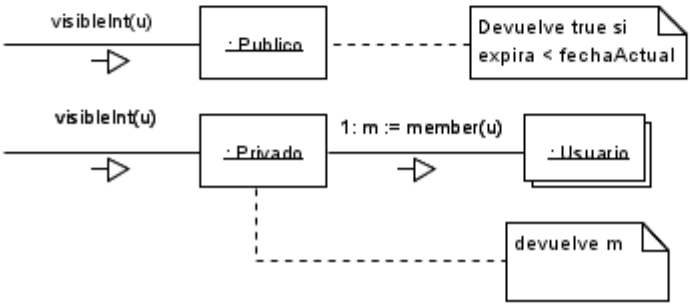
i. Realice los Diagramas de Comunicación para las operaciones descritas anteriormente. No es necesario indicar las visibilidades.

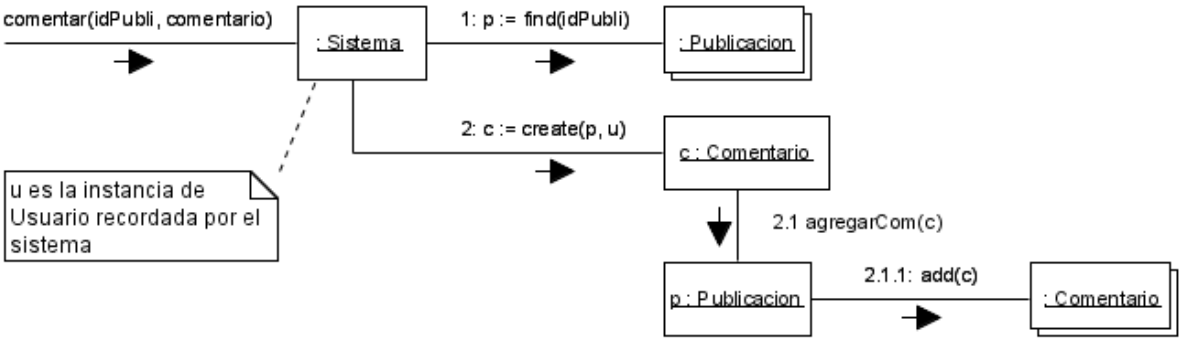
ii. Realice el Diagrama de Clases de Diseño (DCD) correspondiente.

Solución:

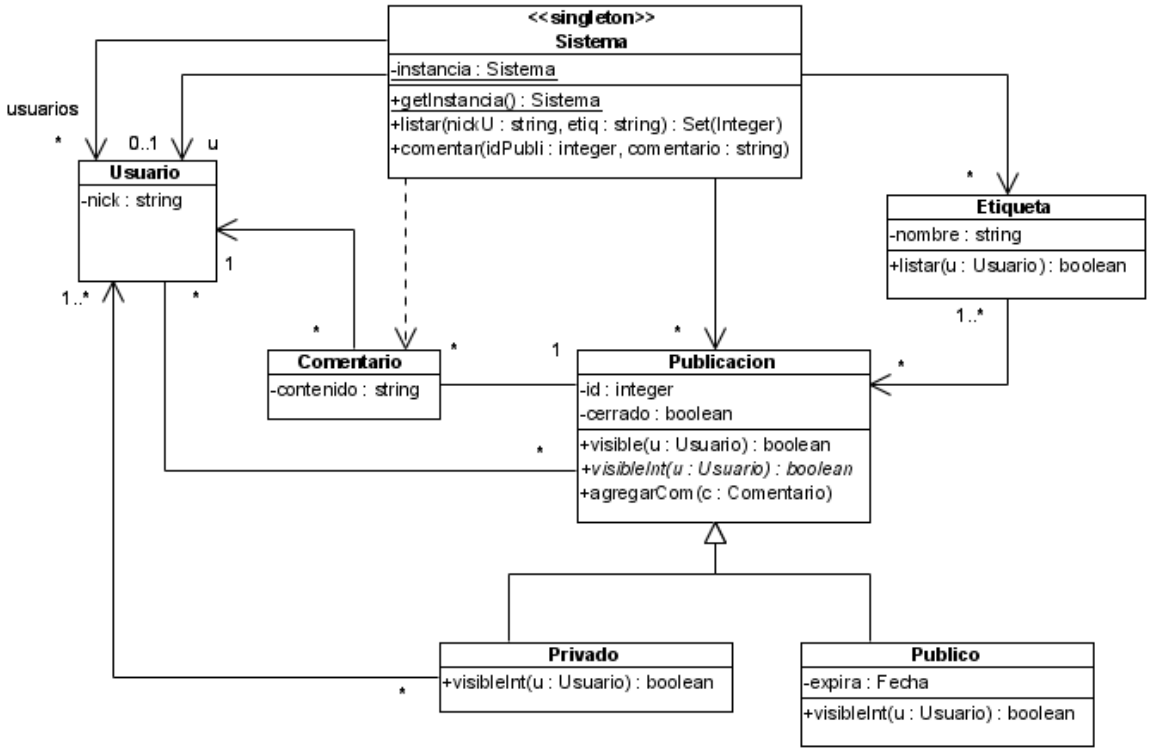
Parte i:







Parte ii:



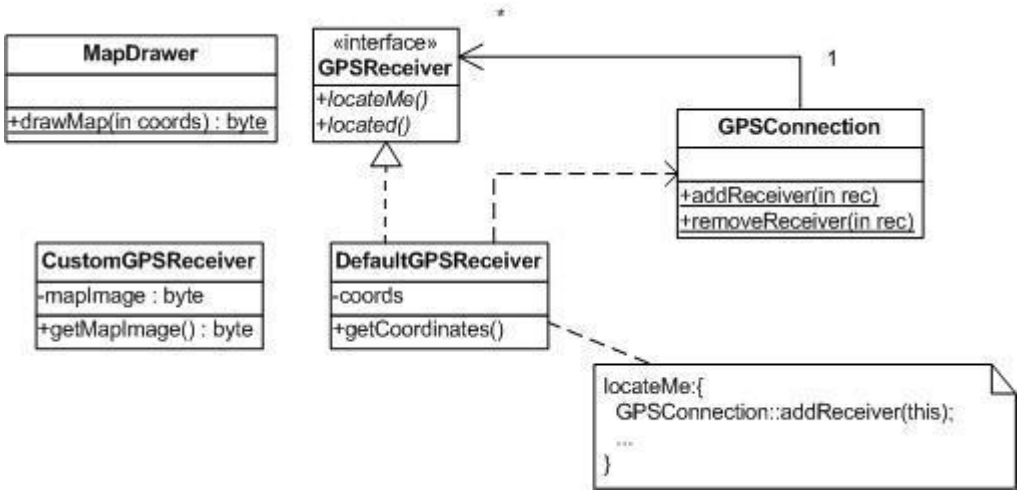
Problema 2 (30 puntos)

La empresa Microtech quiere desarrollar una aplicación para dispositivos móviles que permite geolocalizar sitios relevantes en un mapa utilizando información provista por satélites GPS. Dado que no tiene experiencia, se plantea en primer lugar prototipar la geolocalización del propio dispositivo móvil como un punto en un mapa para en el futuro agregar los sitios relevantes que se encuentren en su entorno.

Para ello la empresa adquirió 3 clases utilitarias ya implementadas: una primer clase que facilita el acceso a la comunicación con los satélites (llamada `GPSConnection`), una segunda clase que permite obtener la imagen de un mapa a partir de un conjunto de coordenadas (llamada `MapDrawer`) y por último, una tercer clase que permite dibujar geometrías en imágenes (llamada `GeomDrawer`).

Para resolver el diseño de este prototipo, se dividió en 2 etapas. En la primer etapa, el objetivo es aprender a generar la imagen de un mapa centrado en las coordenadas de la ubicación del dispositivo móvil. En la segunda etapa, el objetivo es dibujar el punto de la ubicación del dispositivo sobre el mapa ya generado.

Aquí debajo se muestra un DCD que contiene las clases utilitarias y el diseño parcial de la solución para la primer etapa.



Para que un objeto representando al dispositivo móvil sea geolocalizado se debe registrar en la clase utilitaria `GPSConnection` (mediante `addReceiver`) quien solicita al sistema de satélites que se lo ubique geográficamente en algún sistema de coordenadas. Una vez ubicado, `GPSConnection` notifica de ello invocando desde su implementación interna a la operación `located`.

La interfaz `GPSReceiver` permite realizar las solicitudes al sistema de satelites y que éste notifique. Cuenta con dos operaciones. `locateMe` tiene como responsabilidad solicitarle al sistema de satélites la ubicación (en algún sistema de coordenadas). Por su parte, `located` es la operación invocada desde la clase `GPSConnection` (luego que el objeto se registra en ella) con el fin de ser notificado cuando es localizado geográficamente.

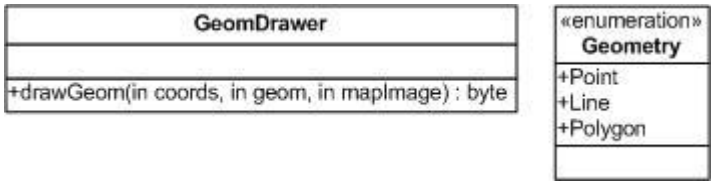
La clase `DefaultGPSReceiver` es una implementación por defecto de la interfaz descripta. Implementa `locateMe` registrándose en la clase GPSConnection `eimplementa located` que en su método captura las coordenadas (mediante la operación `getCoordinates)` que la clase `GPSConnection` detectó (a través del sistema de satélites) al geolocalizar el último objeto registrado, almacenándolas luego en su atributo. Cabe destacar que GPSConnection puede geolocalizar sólo un objeto a la vez.

Con estos recursos se diseña parcialmente la clase `CustomGPSReceiver` cuyo comportamiento le debe permitir registrarse en la clase `GPSConnection` como lo hace la clase `DefaultGPSReceiver` y que, al ser localizado geográficamente, pueda generar la imagen de un mapa centrado en las coordenadas de su ubicación utilizando la clase `MapDrawer` y posteriormente almacenarlo en el atributo correspondiente. Para simplificar, las imágenes se representan mediante valores del tipo `byte`.

Se pide:

i. Completar el diseño de la clase `CustomGPSReceiver` considerando la realidad planteada. Indicar, en caso que aplique, los patrones de diseño que se utilizaron.

Se continúa con la segunda etapa del diseño con el fin agregar a la clase `CustomGPSReceiver` la capacidad de dibujar el punto de la ubicación actual del dispositivo móvil sobre el mapa generado. Por lo que se utiliza la clase `GeomDrawer` que posee un método que recibe una coordenada, el valor de un enumerado (`Geometry`), la imagen de un mapa y retorna la imagen del mapa con la figura representada en el enumerado en las coordenadas indicadas.



Se quiere generar la imagen del mapa cuando el objeto es localizado, dejándola almacenada en el mismo atributo de la clase `CustomGPSReceiver` (como ocurría en la primer parte del diseño).

Sólo se deben considerar los aspectos mencionados en esta realidad al momento de diseñar la solución. Considerar que la solución debe ser reusable así como evitar repetir tareas o código.

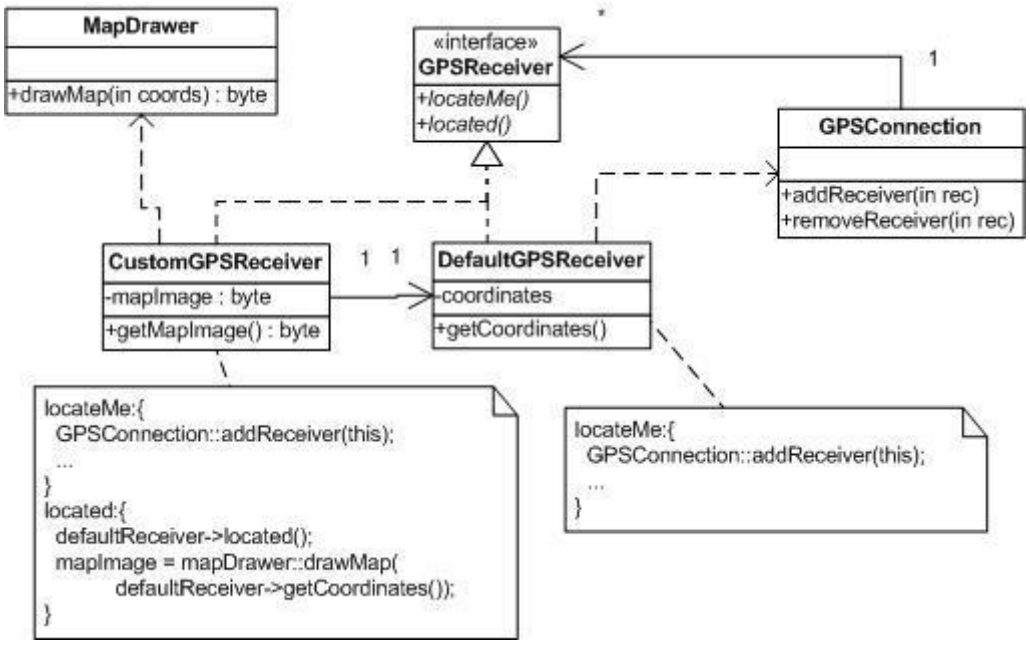
Se pide:

ii. Modificar el diseño de la parte anterior para soportar el nuevo requerimiento. Indicar, en caso de que aplique, los patrones de diseño que se utilizaron.

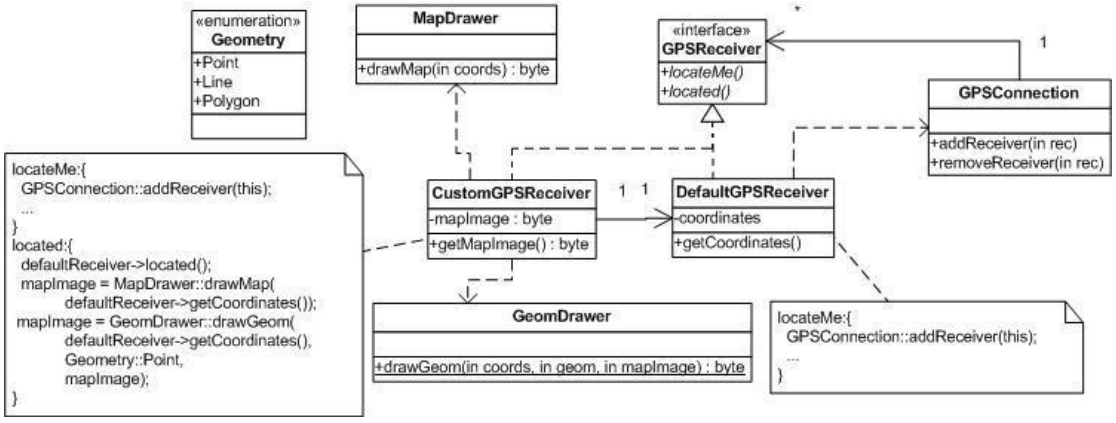
iii. Implementar en C++ la clase `CustomGPSReceiver`. Considerar, en caso de ser necesario, que existen implementaciones disponibles de Listas e Iteradores con las interfaces dadas en el curso.

Solución:

Parte i.



Parte ii.



Patrones utilizados: Observer y Proxy,

Parte iii.

```
class CustomGPSReceiver : public GPSReceiver{
      private:
            char *mapImage;
            DefaultGPSReceiver *defaultReceiver;
      public:
            CustomGPSReceiver(DefaultGPSReceiver *defaultReceiver);
            locateMe();
            char *located();
            char *getMapImage();
            virtual ~CustomGPSReceiver();
};

CustomGPSReceiver::CustomGPSReceiver(DefaultGPSReceiver
*defaultReceiver){
      this->defaultReceiver = defaultReceiver;
      this->mapImage = 0;
}
CustomGPSReceiver::locateMe(){
      GPSConnection::addReceiver(this);
}
char *CustomGPSReceiver::located(){
      defaultReceiver->located();
      mapImage = MapDrawer::drawMap(
                  defaultReceiver->getCoordinates());
      mapImage = GeomDrawer::drawGeom(
                  defaultReceiver->getCoordinates(),
                  Geometry::Point,
                  mapImage);
}
char *CustomGPSReceiver::getMapImage(){
      return mapImage;
}
CustomGPSReceiver::~CustomGPSReceiver(){
      if(mapImage != 0)
            delete mapImage;
      delete defaultReceiver;
}
```