UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Introducción a las Ciencias de la Computación

Práctica 6: Agregación de clases.

Profesora: Amparo López Gaona
Ayudante: Ramsés Antonio López Soto
Ayudantes de Laboratorio:
Adrián Aguilera Moreno
Kevin Jair Torres Valencia

Objetivos

El objetivo de esta práctica es que el alumno consolide sus conocimientos acerca del concepto de agregación al utilizar en la estructura de objetos de una clase en desarrollo, objetos de otras clase.

Introducción

Una vez programada una clase puede considerarse como un nuevo tipo de datos. Como consecuencia, es igualmente sencillo incluir objetos que datos de tipos primitivos en la estructura de una clase. Lo único que debe hacerse es incluir la referencia a ellos. Se debe recordar que se tiene la referencia al objeto, no el objeto en sı. La relación en la que un objeto contiene a otro(s) se denomina relación de agregación.

Es necesario que en los constructores se creen los objetos contenidos, porque en la estructura sólo se declara la referencia a ellos. De esta forma, la creación de un objeto puede llevar a la creación de otros. Una vez creados los objetos contenidos se está en posibilidad de enviarles mensajes.

Para compilar la clase contenedora es necesario que las clases de los objetos contenidos estén en el mismo directorio donde se compile el archivo o bien especificar en la trayectoria de acceso a los archivos el lugar en donde se puede encontrar; no es necesario agregar ningún código extra en el archivo.

En el comentario de inicio de la clase se incluye una lınea que empieza con `@see` para especificar que se incluyen objetos de otra clase cuya documentación puede consultarse.

Desarrollo

1. **Cuadrilátero**
Instrucciones:

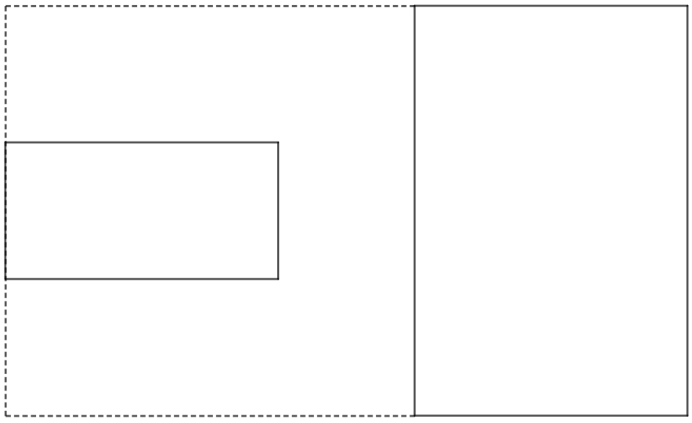
1. En el repositorio `GitHub`, que te asignó, se encuentra un archivo con extensión `.class` llamado `Punto`. Debes moverlo al directorio donde vayas a trabajar esta práctica.

2. Consultar en el material de la práctica la documentación de la clase `Rectangulo` para poder construir la clase de acuerdo con las especificaciones señaladas en la misma.

3. Crear una clase para definir un rectángulo a partir de dos puntos que representan los extremos de una diagonal.

4. La clase `Rectangulo` deberá proporcionar métodos para:

    - Construir un cuadrilátero a partir de dos puntos.
    - Determinar si se trata de un cuadrado o de un rectángulo.
    - Calcular el perımetro del cuadrilátero.
    - Calcular el área del cuadrilátero.
    - Determinar el cuadrilátero mínimo que contiene la unión de dos rectángulo, e.g.



    - Determinar si son iguales dos rectángulos.

2. **Triángulo**
Instrucciones:

1. En el repositorio `GitHub`, que te asignó, se encuentra un archivo con extensión `.class` llamado `Punto`. Debes moverlo al directorio donde vayas a trabajar esta práctica.

2. Consultar en el material de la práctica la documentación de la clase `Triangulo` para poder construir la clase de acuerdo con las especificaciones señaladas en la misma.

3. Crear una clase para definir un triángulo como un conjunto de 3 puntos no alineados. Los atributos de la clase `Triangulo` deberán ser únicamente tres objetos de la clase `Punto`.

4. La clase `Triangulo` deberá proporcionar constructores y métodos para:

   - Un constructor que defina el triángulo cuyos extremos están en los puntos (0,0), (10,0) y (5,10).
   - Un constructor para un triángulo a partir de tres puntos.
   - Un constructor copia.
   - Un método para determinar el perımetro del triángulo.
   - Un método para determinar el área del triángulo con lados de longitud a, b y c utilizando la fórmula de Herón que dice que si el semi-perímetro de un triángulo es $s = \frac{(a+b+c)}{2}$ se tiene que el área es $\sqrt{s(s-a)(s-b)(s-c)}$.
   - Un método para determinar el tipo de triángulo que se trata: equilátero, escaleno o isósceles.
   - Un método para determinar si dos triángulos son iguales.
   - Un método para recuperar una cadena con la información del triángulo de la forma:

$$ab : (x_1, y_1) \to (x_2, y_2); \quad bc : (x_, y_2) \to (x_3, y_3); \quad ac : (x_1, y_1) \to (x_3, y_3)$$

3. Crea una clase `Main.java` que pruebe cada método de los problemas 1 y 2. Procura que se pueda leer de manera rápida y fácil, este será un rubro importante para avaluar este problema.

Formato de Entrega

1. Las prácticas serán entregadas de forma individual.
2. Cada práctica (sus archivos y directorios) deberá estar contenida en un directorio llamado `apellidoPaterno_nombre_pX`, donde X es el número de la práctica.
   Por ejemplo: `aguilera_adrian_p6`
3. NO incluir los archivos .class dentro de la carpeta.
4. Los archivos de código fuente deben estar documentados.
5. Se pueden discutir y resolver dudas entre los integrantes del grupo. Pero cualquier práctica plagiada total o parcialmente será penalizada con cero para los involucrados.
6. La práctica se debe subir al Github Classroom correspondiente.
7. La entrega en classroom debe contener el link HTTPS y SSH de su repositorio y es lo único que se debe entregar.
8. El horario y día de entrega se acordará en la clase de laboratorio y no deberá sobrepasar 2 clases de laboratorio.