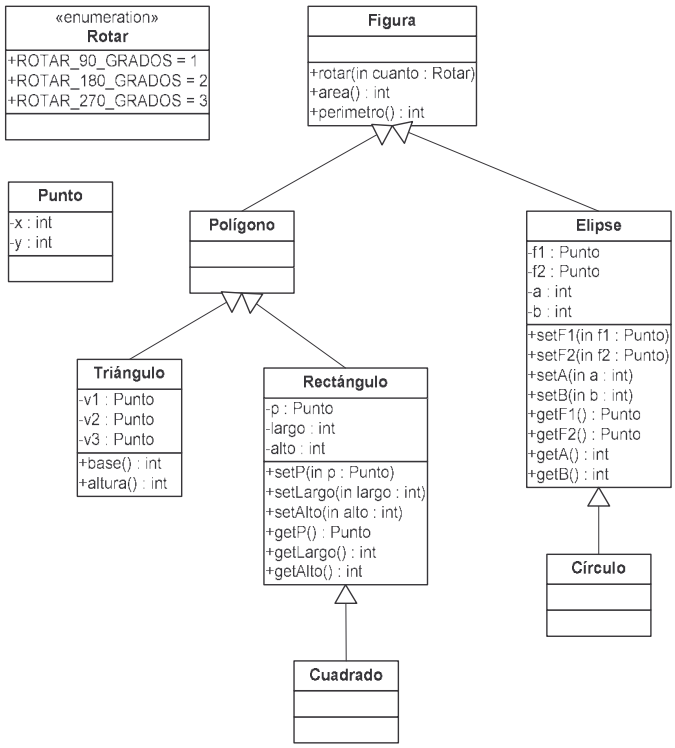
Programación orientada a objetos. Segundo parcial 2003.

Viernes 20 de junio de 2003.

1. Sean las clases que aparecen a continuación.



   Todos los métodos de la clase `Figura` son abstractos. Indiquen en que clases es necesario **implementar** y en que clases es conveniente **sobrescribir** estos métodos, para que una variable declarada de la clase `Figura` pueda contener instancias de `Triángulo`, `Rectángulo`, `Cuadrado`, `Elipse` y `Círculo`, aplicando el **principio de sustitución**. Justifiquen la respuesta.

   Cuándo es necesario **implementar** un método en una subclase es obvio: el método de la superclase es abstracto o la semántica de la subclase lo requiere. Entendemos que es conveniente **sobrescribir** un método de una superclase cuando el método en la subclase puede ser implementado nuevamente de forma más sencilla o con mejor desempeño.

   Implementen la respuesta en Java. Para no pasar vergüenza con cuestiones de geometría elemental, pongan un comentario describiendo como funcionarían los métodos.

   13 puntos

2. Indiquen las **precondiciones**, **poscondiciones**, e **invariantes** que es necesario agregar en las clases `Cuadrado` y `Círculo` del ejercicio anterior. Indiquen también en que métodos es necesario agregarlas. Justifiquen la respuesta.

   Programen la respuesta en Java. Pueden escribir solamente las diferencias con el código que ya han escrito, mientras quede claro dónde introducen las modificaciones.

   13 puntos

3. Cuando un emisor envía un mensaje a un receptor:

   a. ¿De quién es la "culpa" cuando se viola una **precondición**?

   b. ¿De quién es la "culpa" cuando se viola una **poscondición**?

   c. ¿De quién es la "culpa" cuando se viola una **invariante de clase**?

   Justifiquen sus respuestas.

Hagan un programa en Java en donde se viole una precondición y otro en donde se viole una poscondición.

13 puntos

4. Completen la siguiente tabla en referencia a **precondiciones** y **poscondiciones**:

	Obligaciones	Beneficios
Emisor		
Receptor		

13 puntos

5. Java usa siempre **encadenamiento dinámico** para los métodos de instancia y **encadenamiento estático** para los métodos de clase. C++ y Object Pascal permiten al programador seleccionar el mecanismo de encadenamiento método por método; casualmente ambos lenguajes emplean la palabra clave `virtual` para seleccionar encadenamiento dinámico; en ausencia de esa palabra clave, el encadenamiento es estático.

   ¿Cómo influye esto en el **polimorfismo**?

   a. No influye absolutamente para nada.
   b. Influye negativamente.
   c. Influye positivamente.
   d. Influye negativamente y positivamente.

   Justifiquen su respuesta.

12 puntos

6. ¿Puede una subclase redefinir una **invariante de clase** de forma tal que sea más estricta que en la superclase? Es decir, la nueva invariante se cumple en un objeto de la subclase pero no en uno de la superclase.

   ¿Qué relación tiene esto con el **principio de sustitución**?

   a. No tiene ninguna relación.
   b. Están estrechamente relacionados.

   Justifiquen su respuesta.

12 puntos

7. ¿Puede una **precondición**, **poscondición** o **invariante de clase** hacer referencia a **atributos privados** de la clase?

   ¿Puede una **invariante** o una **poscondición** hacer referencia a los **argumentos de un método**?

   Justifiquen su respuesta.

   En caso que alguna respuesta sea afirmativa, programen un pequeño ejemplo en Java para esa o esas respuestas.

12 puntos

8. Java no tiene soporte nativo para **precondiciones**, **poscondiciones** e **invariantes de clase**; es necesario simularlas usando la palabra clase **`assert`**. ¿Qué problemas tiene esto?

12 puntos