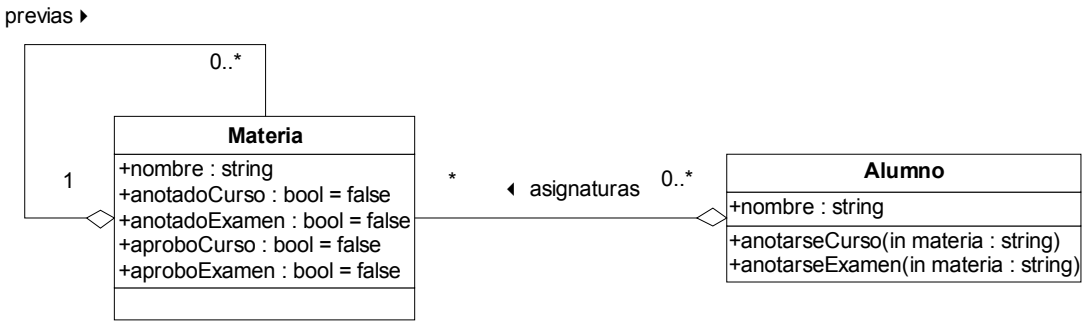
Programación orientada a objetos. Examen febrero 2003.

Lunes 3 de febrero de 2003.

1. Consideren las clases `Alumno` y `Materia` para la inscripción a cursos y exámenes. Cada alumno conoce las materias que debe cursar durante toda la carrera.



La clase `Materia` tiene los siguientes atributos:

- `nombre`: es el nombre de la materia.
- `anotadoCurso`: inicialmente es `false`. Pasa a `true` cuando el alumno se inscribe al curso.
- `anotadoExamen`: inicialmente es `false`. Pasa a `true` cuando el alumno se inscribe al examen.
- `aproboCurso`: inicialmente es `false`. Pasa a `true` cuando el alumno aprueba el curso.
- `aproboExamen`: inicialmente es `false`. Pasa a `true` cuando el alumno aprueba el examen.

La clase `Materia` también tiene métodos `get*()` y `set*()` para acceder y modificar el valor de los atributos.

La clase `Alumno` tiene los siguientes atributos:

- `nombre`: es el nombre del alumno.

La clase `Alumno` también tiene métodos `get*()` y `set*()` para acceder y modificar el valor del atributo, y métodos `anotarseCurso(String nombre)` y `anotarseExamen(String nombre)`. Estos dos últimos métodos buscan en la lista de materias del alumno, la que tengan el nombre pasado como argumento, y asignan `true` al atributo `anotadoCurso` y `anotadoExamen`, respectivamente, de la materia encontrada.

Una parte del código de estas clases aparece a continuación. En `Materia.java`:

```
import java.util.*;

public class Materia {

  private String m_nombre;
  private boolean m_anotadoCurso = false;
  private boolean m_anotadoExamen = false;
  private boolean m_aproboCurso = false;
  private boolean m_aproboExamen = false;
  private Set m_previas = new HashSet();

  public Materia(String nombre) {
    m_nombre = nombre;
  }

  public boolean getAnotadoCurso() {
    return m_anotadoCurso;
  }

  public void setAnotadoCurso(boolean value) {
    m_anotadoCurso = value;
  }
```

```
  public boolean getAnotadoExamen() {
    return m_anotadoExamen;
  }

  public void setAnotadoExamen(boolean value) {
    m_anotadoExamen = value;
  }

  public boolean getAproboCurso() {
    return m_aproboCurso;
  }

  public void setAproboCurso(boolean value) {
    m_aproboCurso = value;
  }

  public boolean getAproboExamen() {
    return m_aproboExamen;
  }

  public void setAproboExamen(boolean value) {
    m_aproboExamen = value;
  }

  public String getNombre() {
    return m_nombre;
  }

  public void setNombre(String value) {
    m_nombre = value;
  }

  public Set getPrevias() {
    return m_previas;
  }
}
```

En `Alumno.java`:

```
import java.util.*;

public class Alumno {

  private Set m_materias = new HashSet();

  public Set getMaterias() {
    return m_materias;
  }

  public void anotarseCurso(String nombre) {
    /* Insert coin here */
  }

  public void anotarseExamen(String nombre) {
    /* Insert coin here */
  }
}
```

Programen los métodos `anotarseCurso` y `anotarseExamen`.

20 puntos

2. Definan en forma breve y precisa **precondición** y **poscondición**. Agreguen a las clases `Materia` y `Alumno` del ejercicio anterior las precondiciones y poscondiciones que consideren necesarias para asegurar que:
   - Si el alumno no está anotado a un curso, no puede tener ni el curso ni el examen aprobado, ni puede estar anotado al examen.
   - Si el alumno no tiene aprobado un curso, no puede estar anotado al examen, ni tener el examen aprobado.
   - Si el alumno no está anotado al examen, no puede tener el examen aprobado.

- El alumno se puede anotar a un curso si tiene aprobados los cursos de las materias previas.
- El alumno se puede anotar a un examen si tiene aprobado el curso y los exámenes de las materias previas.

Agreguen estas afirmaciones al código del ejercicio anterior usando la cláusula **`assert`** de Java. No tienen porqué escribir nuevamente toda la clase, mientras indiquen claramente dónde se inserta cada modificación.

10 puntos

3. Cuando un objeto "cliente" envía un mensaje a un objeto "servidor" ¿de quién es la "culpa" cuándo se viola una precondición o una poscondición? Justifiquen su respuesta.

   Usando las clases de los ejercicios 1y 2 anteriores, escriban un pequeño fragmento de código en Java que muestre la violación de una de las precondiciones o poscondiciones.

   10 puntos

4. Hagan un programa `Prueba` en Java que, usando las clases `Materia` y `Alumno` de los ejercicios 1 y 2 anteriores:

   - Cree un alumno y dos materias, una previa de la otra.
   - Inscriba al alumno a los cursos y exámenes.
   - Apruebe los cursos y exámenes

   Las inscripciones y aprobaciones de cursos y exámenes deberán estar en el orden correcto para que no haya excepciones ni violaciones de precondiciones o poscondiciones.

   Además del código fuente del programa deberán indicar el nombre del archivo que debe contener ese código y la sentencia de línea de comando con la que se debe compilar ese archivo para ejecutar el programa tal como aparece más arriba.

   10 puntos

5. Supongan que existen dos clases diferentes que no tienen métodos con signaturas en común. ¿Es posible tener un objeto capaz de ejecutar todos los métodos de esas clases? ¿Cómo? ¿Hay más de una manera? ¿Cuáles? No pueden escribir los métodos; deben usar los que ya están implementados.

   Además de contestar las preguntas anteriores programen las respuestas en Java usando las siguientes clases:

```
public class Cliente {
  public int getNumeroCliente() {…}
  public void setNumeroCliente(int numeroCliente) {…}
}

public class Alumno {
  public String getEMail() {…}
  public void setEMail(String eMail) {…}
}
```

   10 puntos

6 Expliquen en forma clara y precisa cómo funciona el **encadenamiento estático** y el **encadenamiento dinámico** en esos lenguajes. ¿El concepto es aplicable a todos los lenguajes?

   Muestren las diferencias entre encadenamiento estático y dinámico con un pequeño fragmento de código en Java usando el siguiente ejemplo, suponiendo sólo por un instante que Java soportara encadenamiento estático.

```
public class Persona {
  public void caminar() {}
}


public class Cliente extends Persona {
  public void caminar() {}
}

public class Estudiante extends Persona {
```

```
    public void caminar() {}
}
```

10 puntos

7 Nuevamente supongan sólo por un instante que Java soportara encadenamiento estático pero no así encadenamiento dinámico ¿Qué característica importante de Java se perdería? Muéstrenlo con un pequeño fragmento de código en Java usando las clases del ejercicio 6.

10 puntos

8. ¿Qué es el **principio de sustitución** y para qué sirve? Muéstrenlo con un pequeño fragmento de código en Java usando las clases del ejercicio 6.

10 puntos

9. Un objeto puede tener más de un tipo y varios objetos de diferentes clases pueden tener el mismo tipo. Muestren con un pequeño fragmento de código de Java un ejemplo de un objeto con más de un tipo y otro con varios objetos de diferentes clases y el mismo tipo.

10 puntos