Introducción a la Programación Orientada a Objetos
DCIC – UNS Primer Cuatrimestre de 2013

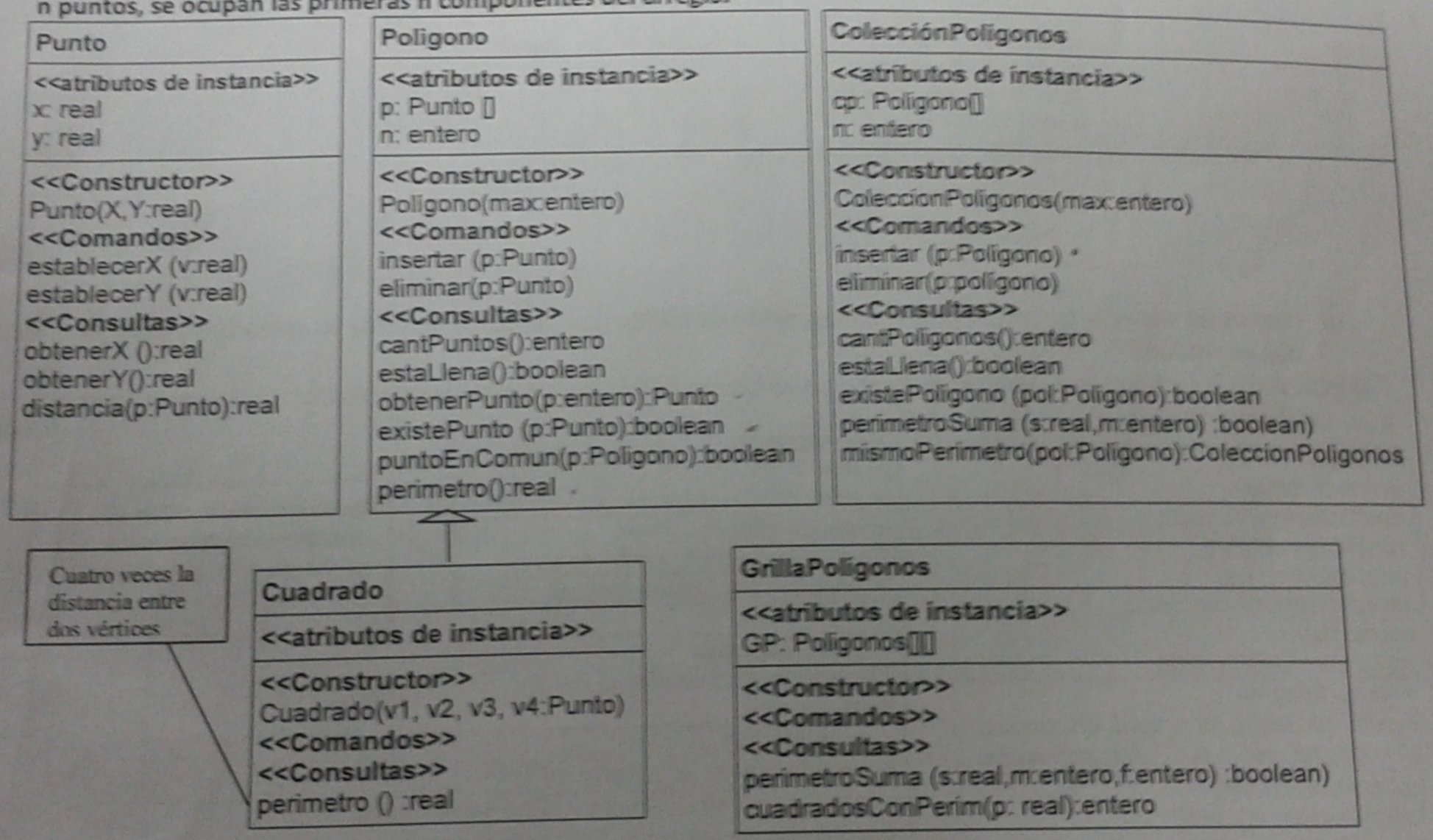
RECUPERATORIO PRIMERO , SEGUNDO Y TERCER PARCIAL – 28/06/2013

APELLIDO Y NOMBRE: ..... LU: .....
CANTIDAD DE HOJAS ENTREGADAS (SIN ENUNCIADO): ...4.....
Importante: La interpretación del enunciado es parte del examen.
SE TOMARÁ EN CUENTA A LA HORA DE CORREGIR LA **CORRECTITUD, LEGIBILIDAD Y EFICIENCIA** DE LAS SOLUCIONES PROPUESTAS.

EJERCICIO 1

La clase ColeccionPoligonos modelada en el diagrama permite almacenar la representación de una **colección** de polígonos. Cada vez que se inserta un polígono en la colección se incrementa el valor de n. Si se almacenan n polígonos, se ocupan las primeras n componentes del arreglo.

Cada polígono de n lados se representa como una **secuencia** de n puntos. El servicio obtenerPunto retorna el punto que ocupa la posición p en la secuencia, comenzando desde la posición 1. Cada lado del polígono queda determinado por los elementos que ocupan las posiciones i e i+1 en la secuencia, excepto uno de los lados que se forma con los puntos que ocupan la posición 1 y n. Cada vez que se inserta un punto en un polígono, se incrementa el valor de n. **Si se almacenan n puntos, se ocupan las primeras n componentes del arreglo.**



IMPLEMENTE LOS SIGUIENTES MÉTODOS

- **puntoEnComun(p:Poligono):** retorna verdadero si el polígono que recibe el mensaje tiene algún punto en común con el polígono pasado por parámetro
- **perimetroSuma(s:real,m:entero,f:entero):** retorna verdadero si la grilla contiene en la fila **f**, **m** polígonos almacenados en forma consecutiva, cuyos perímetros suman **s**.
- **mismoPerimetro(pol:Poligono):ColeccionPoligonos:** retorna todos los polígonos cuyo perímetro sea igual al del pasado por parámetro.
- **cuadradosConPerim(p: real):**entero retorna la cantidad de cuadrados con perímetro p que hay en la grilla

EJERCICIO 2: Una secuencia de números S = a1 a2 ..akbk .. b2 b1 de longitud par (2k) se dice "Antireflex" si no existe ningún par ai=bi, con i≤K. Ejemplos:

* S = 1 20 3 5 23 12 (longitud=6) ES "Antireflex" .
* S = 6 2 4 5 3 41 9 (longitud=8) NO ES "Antireflex" (ya que a3 = b3= 4)
* Si S es vacía, NO ES "Antireflex".

a) Escriba un planteo recursivo para determinar si una secuencia S de longitud par es "Antireflex".
b) Escriba un método recursivo que lea una secuencia S de longitud N (par) ingresada por teclado e implemente el planteo recursivo para decidir si la secuencia es o no Antireflex.

EJERCICIO 3: Dadas las siguientes definiciones de clases:

```
class Uno {
  public void alfa (int x)
    {System.out.println("alfa en Uno");}
  public void beta (int x)
    {System.out.println("beta en Uno");}
  public void delta (int x)
    {System.out.println("delta en Uno");}}
```

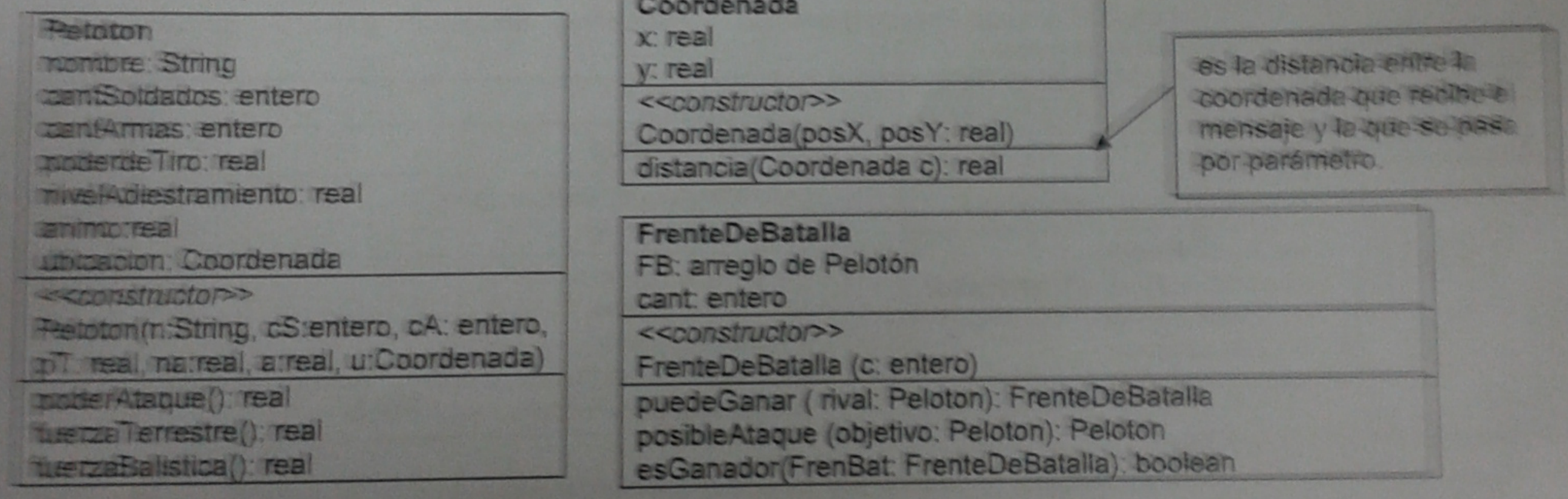
```
class Dos extends Uno {
  public void beta (int x)
    {System.out.println("beta en Dos");}
  public void delta (String s)
    {System.out.println("delta en Dos");}}
class Tres extends Dos {
  public void beta (String s)
    {System.out.println("beta en Tres");}
  public void alfa (int x)
    {System.out.println("alfa en Tres");}
  public void delta (int x)
    {System.out.println("delta en Tres");}}
```

a) Indique que instrucciones provocarán errores de compilación, justificando apropiadamente su respuesta, y muestre la salida para las instrucciones que no tienen errores

```
Uno u = new Dos();
u.beta(1);
u.delta(1);
u = new Tres();
u.alfa(1);
u.beta(1);
u.delta(1);
u.delta("hola");
Dos d = new Dos();
d.delta(1);
d.delta("hola");
u = d;
u.alfa(1);
u.beta(1);
u.delta(1);
u.delta("hola");
```

b) Usando las clases anteriores muestre un segmento de código que produzca un error en ejecución y explique por qué se produce.

EJERCICIO 3: En un batallón se va a llevar a cabo un simulacro bélico. Para eso se decide diseñar un diagrama de clases que modele el problema.



- Implemente los siguientes métodos de la clase Peloton
  - **fuerzaTerrestre(): real** Se calcula como la cantidad de armas por el poder de tiro
  - **fuerzaBalistica(): real** Se calcula como la cantidad de soldados por el nivel de adiestramiento
  - **poderAtaque() : real** Es la suma de fuerza terrestre y la fuerza balística, multiplicado por el ánimo del pelotón

Implemente los siguientes métodos de la clase FrenteDeBatalla

- **puedeGanar(rival: Peloton): FrenteDeBatalla**
  Devuelve un frente de batalla conteniendo los pelotones que pueden ganarle al pelotón rival. Se tiene en cuenta que un pelotón gana a otro si tiene mayor poder de ataque.
- **posibleAtaque(objetivo:Pelotón): Pelotón**
  Devuelve el pelotón del frente de batalla que esté a menor distancia del pelotón objetivo
- **esGanador(FrenBat: FrenteDeBatalla): boolean**
  Devuelve verdadero si todos los pelotones del frente de batalla que recibe el mensaje le ganan a los respectivos pelotones del frente de batalla pasado por parámetro. La comparación se realiza uno a uno entre los pelotones de ambos frentes de batalla. Se asume que ambos Frentes tienen la misma cantidad de pelotones