Introducción a la Programación Orientada a Objetos
DCIC – UNS
14 de diciembre de 2022

Examen de promoción
LA EVALUACIÓN CONSIDERA LA **CORRECTITUD, LEGIBILIDAD Y EFICIENCIA** DE LAS SOLUCIONES.

La interpretación del enunciado es parte del examen.

1. Dado el siguiente segmento de código:

```
Robot r = new RobotAlfa(123);
boolean t = r.tieneEnergia();
```

   - Explicá el significado de cada instrucción.
   - Enumerá las causas por las cuales cada una de las instrucciones anteriores podría dar error.
2. Explicá la diferencia entre copy superficial y en profundidad y proponé un ejemplo simple y un diagrama de objetos para ilustrar esta diferencia.
3. Definí el concepto de polimorfismo. Explicá por qué en Java el chequeo de tipos limita el nivel de polimorfismo.
4. Dado el siguiente diagrama de clases:

MatrizGenerica
<< Atributos de instancia >> mat: [ ][ ] Elemento
<< constructor >> MatrizGenerica(n, m: entero) << Comandos >> establecer(fila: entero, col: entero, e: Elemento) << Consultas >> obtener(fila: entero, col: entero): Elemento cantFilas(): entero cantCol(): entero segundoNulo(): VectorGenerico cantInversosMitad (): entero

VectorGenerico
<< Atributos de instancia >> vec: [ ] Elemento
<< constructor >> VectorGenerico(m: entero) << Comandos >> establecer(i: entero, e: Elemento) << Consultas >> obtener(i: entero): Elemento cantElementos():entero

Punto
<< Atributos de instancia>> x: real y: real
<< constructor >> Punto(xa, ya: real) << Consultas >> obtenerX(): real obtenerY(): real unoNulo():boolean esInverso(e: Elemento): boolean difX(p:Punto):real

*Elemento
*unoNulo():boolean *esInverso(e: Elemento): boolean

a) Implementá las clases VectorGenerico y MatrizGenerica considerando las siguientes funcionalidades y responsabilidades:

- **VectorGenerico(m: entero)** crea una matriz de `n` filas por `m` columnas y la liga a `mat`. Requiere `n` y `m` if mmayores a 0.
- **MatrizGenerica(n, m: entero)** crea una matriz de `n` filas por `m` columnas y la liga a `mat`. Requiere `n` y `m` if mmayores a 0.
- **segundoNulo(): VectorGenerico** genera un vector de `cantFilas()` componentes. Recorre cada fila `i` de `mat` y cuando encuentra el primer par de elementos en posiciones consecutivas tales que ambos verifican la propiedad unoNulo, asigna la referencia del segundo de estos elementos a la posición `i` del vector. Si no encuentra dos elementos en posiciones consecutivas que verifican la condición mencionada, la posición `i` del vector queda en nulo.

- **cantInversosMitad (): entero** consideremos que una matriz de nFil x nCol se divide verticalmente por la mitad, esto es, la mitad izquierda ocupa las columnas entre 0 y (nCol/2)-1 y la mitad derecha corresponde a las columnas entre nCol/2 y nCol-1. Es decir, la posición 0 en la mitad derecha corresponde a la columna nCol/2. La consulta retorna la cantidad de filas que verifican la siguiente propiedad: cada uno de los elementos de la mitad izquierda es el inverso del elemento que ocupa la misma posición en la mitad derecha. Requiere nCol par y todas las componentes ligadas.

b) Implementá la clase Elemento con los servicios que usa la clase MatrizGenerica.

c) Implementá la clase Punto que extiende a Elemento considerando que:

- **unoNulo():boolean** retorna true si **x** es 0 o bien **x** es distinto de 0 e **y** es 0
- **esInverso(e: Elemento): boolean** retorna true si **x** es igual a la ordenada de **e** y además **y** es igual a la abscisa de **e**. Requiere que **e** esté ligado y sea de clase Punto.
- **difX(p:Punto):real** retorna la diferencia en valor absoluto entre la abscisa del punto que recibe el mensaje y la abscisa del punto **p**.

d) Implementá una clase MatrizPuntos que extienda a MatrizGenerica e implemente el método:

- **diferenciaCreciente():boolean** retorne true si la matriz contiene al menos una fila tal que, todas las posiciones están ligadas y además las diferencias absolutas entre las abscisas de cada par de puntos consecutivos, forman una secuencia creciente.

Por ejemplo, en la siguiente fila las diferencias absolutas entre los valores de las abscisas de puntos consecutivos forman una secuencia creciente:

(6.2,0.0)	**(6.4,-1.0)**	**(6.8,1.0)**	**(5.8,0.0)**	**(7.8,-1.0)**	**(4.0,-1.1)**

En cambio, en la siguiente fila las diferencias absolutas no son crecientes:

(6.4,1.0)	**(6.8,0.0)**	**(7.2,-2.0)**	**(7.8,1.0)**	**(8.0,0.0)**	**(7.8,-1.0)**

e) Implementá un tester que permita verificar el servicio **diferenciaCreciente():boolean**

AUNQUE NO FORMA PARTE DEL EXAMEN SI TERMINA ANTES DEL HORARIO DE ENTREGA LE SUGERIMOS QUE TAMBIÉN IMPLEMENTE EL TESTER DE LOS DEMÁS RECORRIDOS

f) Comprimí el proyecto completo usando WinRar y enviélo a ipoo.dcic@gmail.com.