ELO 326: Segundo Certamen Tiempo 120 minutos Martes 20 de Noviembre de 2001

1.-

a) Indicar dos usos en Java para la palabra reservada ***this***.
Se usa para acceder a los miembros dato que son apantallados por los parámetros de un método.
Se usa para invocar otros constructores.

b) Mencione 3 diferencias entre un Applet y una aplicación.
Un Applet corre dentro de la máquina virtual del navegador
Un Applet tiene restricciones de seguridad como no poder crear archivos locales.
Un applet carece de método estático main, éste es provisto por el framework en el que corre.

c) Explique en qué momento la memoria asociada a los objetos en Java es retornada al sistema.
La memoria asignada del heap es recuperada por el sistema cuando la aplicación o applet no tiene referencia alguna a esa zona de memoria y el sistema efectúa una recolección de basura (garbage collection)

d) Para clase derivada Academico, cómo se efectúa el llamado indicado en ***negrita e itálica***?

```
class Funcionario {
    public void DisplayAtributos() {
            //.... código correspondiente
    }
    //.. otros métodos y atributos
}
class Academico extends Funcionario {
    public void DisplayAtributos () {
            // Aquí se desea llamar a DisplayAtributos de la clase base
            ((Funcionario)this).DisplayAtributos();
            // resto código
    }
    // .. otros métodos y atributos
}
```

2.- Se tiene la siguiente clase:

```
class Circulo
{       public Circulo(Point centro, double radio) {
            c = centro;
            r = radio;
        }
        // ... otros métodos...
        private Point c;
        private double r;
}
```

a) Haga las modificaciones o extensiones necesarias para que esta clase implemente la interfaz Cloneable.

```
class Circulo implements Cloneable
{       public Circulo(Point centro, double radio) {
            c = centro;
            r = radio;
```

```
        }
        public Object  clone()
        {  try{
                Circulo cc = (Circulo)super.clone();
                cc.c = (Point) c.clone();
                return cc;
           } catch (CloneNotSupportedException e) {
                return null;
           }
        }
        // ... otros métodos...
        private Point c;
        private double r;
}
```

b) Implemente el método swap(Circulo A, Circulo B);

```
class Circulo
{
        // ... otros métodos...
        public static void swap(Circulo A, Circulo B) {
                if (A !=null && B != null) {
                        Point p = A.c;
                        double d = A.r
                        A.c=B.c;
                        A.r=B.r;
                        B.c=p;
                        B.r=d;
                }
        }
        private Point c;
        private double r;
}
```

c) Asumiendo que dispone de los métodos de la parte a) y b), implemente el método copiarEn(Circulo destino);

```
class Circulo implements Cloneable
{
        // ... otros métodos...
        public void copiarEn(Circulo destino) {
                destino = clone();
        }
        private Point c;
        private double r;
}
```

3.- La clase del enunciado de la pregunta 2 permite que - por error - se creen instancias de Circulo con radio negativo.

a) A partir de la clase del enunciado en 2.- desarrolle e incorpore los mecanismos necesarios para lanzar la excepción ExcepcionRadioNegativo cuando se detecte esta condición en el constructor dado.

Primero debemos crear nuestra propia clase para la excepción ExcepionRadioNegativo.

```
class ExcepcionRadioNegativo extends Exception
{       public ExcepcionRadioNevativo(){};
        public ExcepcionRadioNegativo(String s) {
                super(s);
        }
}
```

Luego podemos lanzar esta excepción en nuestro código

```
class Circulo
{       public Circulo(Point centro, double radio) throws ExcepcionRadioNegativo {
                c = centro;
                if (r <0) throw (new ExcepcionRadioNegativo("El radio no puede ser negativo!!");
                r = radio;
        }
        // ... otros métodos...
        private Point c;
        private double r;
}
```

b) Adapte el siguiente código a la nueva situación plateada.

```
class Cilindro
{       public Cilindro (Point centro, double radio, double altura) {
                try {
                        base = new Circulo(centro, radio);
                } catch(ExcepcionRadioNegativo e) {}
                h = altura;
        }
        //.... otros métodos
        private Circulo base;
        private double h;
}
```

4.- Haga un programa en C++ que lea la entrada estándar e imprima en pantalla en orden alfabético los palíndromos que encuentre. Un palíndromo es una secuencia de letras delimitada por caracteres no alfabéticos que en orden inverso se lee igual; por ejemplo Anilina, Salas, Ana, level.

```
#include <string>
#include <list>
#include <algorithm>

typedef list<string> Palindromos;

bool isPalindromo(string &s)
{
  for (int i=0; i<= s.length()/2; i++)
    if (s[i]!=s[s.length()-1-i])
      return false;
  return true;
}

// Alternativamente se pudo implementar como sigue
// Cualquier solución es aceptada.
```

```
/*
bool isPalindromo(string &s)
{
  string inv_s=s;
  reverse(inv_s.begin(), inv_s.end());
  return (inv_s == s);
}
*/
int main ()
{
  const char UpperToLower = 'a' - 'A';

  Palindromos palindromos;

  while (cin)
    {
      string w;
      char c;
      do
      {
        cin.get(c);
        if ((c >= 'A') && (c <= 'Z'))
          c += UpperToLower;
        if ((c >= 'a') && (c <= 'z'))
          w += c;
      }
      while ((cin) && ((c >= 'a') && (c <= 'z')));

      if ((w.size() > 0) && isPalindromo(w))
      {
        palindromos.push_back(w);
      }
    }

  palindromos.sort();

  for (Palindromos::iterator j=palindromos.begin(); j!=palindromos.end(); j++)
    cout << (*j) << endl;

  return 0;
}
```