dit UPM	ETSIT-UPM. PROGRAMACIÓN. CURSO 2013-2014. EXAMEN 2. APELLIDOS: NOMBRE: EMAIL:	

1. (2.5 puntos) Escriba un método que tiene como parámetros dos *String*, la primera es un texto y la segunda una palabra. El método devuelve el índice en el que la palabra aparece por primera vez en el texto, o *-1* si no está (o alguno de los parámetros es una *String* nula o vacía). No lanza excepciones. Recuerde que puede usar los métodos *length()*, *charAt(int)* de *String*, y los bucles que considere convenientes; no puede usar los métodos *indexOf* o *subString* que ya vienen hechos con la clase *String*.

```
public int esta (String t, String p) {
  if ((t == null)  || (t.length() == 0) || (p == null) || (p.length() == 0)) {
    return -1;
  }
  if (t.length() < p.length() ) return -1;
  for (int i = 0; i < t.length(); i++) {
    boolean esta = true;
    for (int j = 0; j < p.length(); j++) {
      if (i+j >= t.length()) {
        esta = false;
        break;
      }
      if (t.charAt(i+j) != p.charAt(j)) {
        esta = false;
        break;
      }
    }
    if (esta)
      return i;
  }
  return -1;
}
```

2. (2.5 puntos). Escriba el cuerpo del método *double[][] traspuesta (double[][] m1) throws Exception* que devuelve la matriz traspuesta de la que se le pasa como parámetro. Lanza excepción si el parámetro es null, o si no es rectangular.

```
double[][] traspuesta(double[][] m1) throws Exception {
  if (m1 == null)
     throw new Exception();
  for (int i = 0; i < m1.length; i++)
    if (m1[0].length != m1[i].length)
     throw new Exception();
  double[][] resultado = new double[m1[0].length][ m1.length];
  for (int i = 0; i < resultado.length; i++) {
    for (int j = 0; j < resultado[0].length; j++) {
      resultado[i][j] = m1[j][i];
    }
  }
 return resultado;
}
```

dit UPM	ETSIT-UPM. PROGRAMACIÓN. CURSO 2013-2014. EXAMEN 2. APELLIDOS: NOMBRE: EMAIL:	

3. (2.5 puntos). Dada la siguiente clase, añada los atributos y métodos que considere convenientes y un método constructor con tres parámetros de forma que se pueda saber en todo momento el número de objetos de la clase que se han creado. El constructor lanza una excepción si el nombre es una *String* nula o vacía. Escriba también el método *equals*.

```
class Coordenada {
  private String nombre;
  private double x;
  private double y;
  // métodos accesores
  // métodos modificadores
}
```

```
private static int nCoordenadas = 0;
public static int nCoordenadas () {
  return nCoordenadas;
}
public Coordenada (String nombre, double x, double y) throws Exception {
  if ((nombre == null) || (nombre.length() == 0))
     throw new Exception();
  this.nombre = nombre;
  this.x = x;
  this.y = y;
  nCoordenadas++;
}
public boolean equals (Coordenada otra) {
  return this.nombre.equals (otra.nombre) && (this.x == otra.x) && (this.y == otra.y);
}
```

4. (2.5 puntos) Escriba un método que toma como parámetro una *String* que contiene un número en hexadecimal (los dígitos van del 0 al 9 y después de 'A' a 'F' hasta completar los 16 valores de la base), y devuelve su valor en decimal (base 10). Si la *String* es nula o vacía, o algún dígito hexadecimal no está en los conjuntos ('0'..'9', 'A'..'F') devuelve -1.
Recuerde que puede usar el método *charAt(int)* y *length()* de *String*; y que para calcular el valor decimal que representa un carácter de dígito '0'..'9' se fuerza su tipo a entero y se le resta el código entero del dígito *'0'*. Para un carácter 'A'..'F' se fuerza el tipo a entero, se sumar 10 y se resta el código entero de 'A'.

```
int hex2dec (String hex) {
  if ((hex == null) || (hex.length() == 0))
    return -1;
  int valor = 0;
  for (int i = 0; i < hex.length(); i++) {
     char c = hex.charAt(i);
     if ((c >= '0') && (c<='9'))
       valor = (valor * 16) + ((int) c) - ((int) '0');
     else if ((c >= 'A') && (c<='F'))
       valor = (valor * 16) + 10 + ((int) c) - ((int) 'A'));
     else
       return -1;
  }
  return valor;
}
```