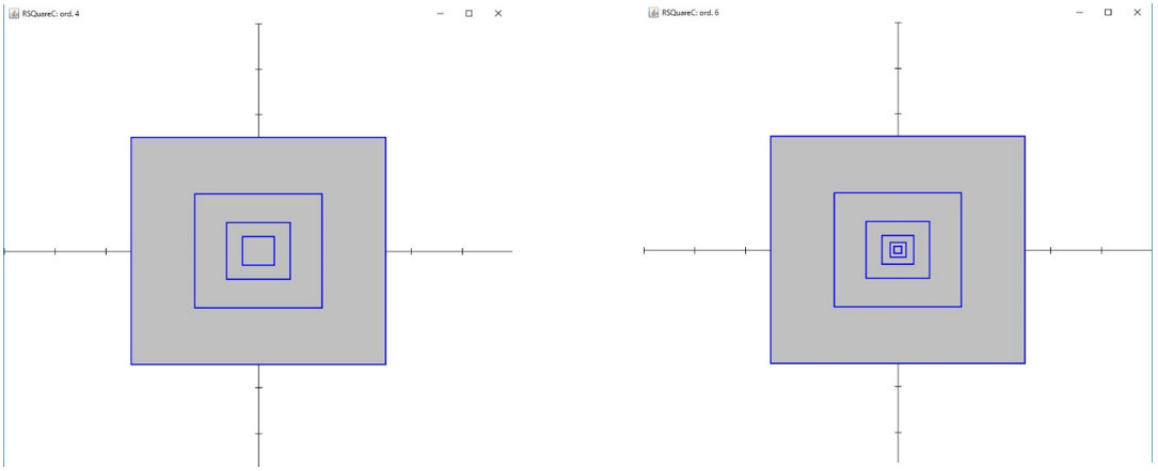
Parcial 1 - PRÁCTICAS - PRG - GIIA. Curso 2017-18
26 de Marzo de 2018. Duración: 1 hora

Nota: El examen se evalúa sobre 10 puntos, pero su peso específico en la nota final de la asignatura es de **0,8 puntos**.

NOMBRE: GRUPO DE PRÁCTICAS:

1. 3 puntos Un ***RSquare-C*** de orden n es una serie de `n` cuadrados tales que: - Todos tienen el mismo centro. - La longitud del lado de cada cuadrado es la mitad de la longitud del cuadrado anterior. - Cada nuevo cuadrado más pequeño se dibuja superpuesto al cuadrado mayor anterior.

   En las siguientes figuras se muestran dos ventanas de dibujo de *RSquare-C*, la de la izquierda de orden `n` = 4, y la de la derecha de orden `n` = 6.



Completar la siguiente implementación de un método recursivo no final que dibuje en la ventana gd un *RSquare-C* de orden n >= 1, con centro en (cX, cY) y cuadrado inicial de lado l.

Código a completar:

```
public static void rSquareC(Graph2D gd, int n, double cX, double cY, double l) {
   if (n > 1) { ... }
   else { ... }
}
```

NOTA: Considerar implementado, y disponible en la misma clase, el método `drawCentSquare` de perfil: `public static void drawCentSquare(Graph2D gd, double cX, double cY, double l);` Este método dibuja en la ventana gd un cuadrado sólido con centro en (cX, cY) y lado l.

Solución:

```
public static void rSquareC(Graph2D gd, int n, double cX, double cY, double l) {
   if (n > 1) {
         drawCentSquare(gd, cX, cY, l);
         rSquareC(gd, n - 1, cX, cY, l / 2);
   }
   else {
         drawCentSquare(gd, cX, cY, l);
   }
}
```

2. [3 puntos] Se considera implementado el método `isPrefix` cuyo perfil es
`public static boolean isPrefix(String a, String b);`
y su funcionalidad consiste en comprobar si el String `a` es un prefijo del String `b`.

Se pide: Completar la siguiente implementación de un método recursivo `howMany` que realice un recorrido del String `b` y devuelva el número de veces que el String no vacío `a` está contenido en el String `b` (i.e., cuántas veces `a` es substring de `b`). Se considera que los métodos `howMany` y `isPrefix` están definidos en la misma clase.

Ejemplos del resultado que devuelve el método `howMany`:
- `howMany(‘‘ab’’, ‘‘bacbac’’)` devuelve 0,
- `howMany(‘‘ab’’, ‘‘babbabb’’)` devuelve 2,
- `howMany(‘‘aaa’’, ‘‘aaaaa’’)` devuelve 3

Código a completar:

```
public static int howMany(String a, String b) {
   if ( ... ) {
      if ( ... ) {  return 1 + howMany(a, ... );   }
      else {  return howMany(a, ... );  }
   }
   else { return 0; }
}
```

NOTA: El método `substring(int)`, aplicado a un String `s`, devuelve la subcadena de `s` formada desde la posición de su parámetro hasta el final de `s`. El método `length()` devuelve la longitud de un String. Ambos métodos pertenecen a la clase `String`.

Solución:

```
public static int howMany(String a, String b) {
   if (a.length() <= b.length()) {
      if (isPrefix(a, b)) {
          return 1 + howMany(a, b.substring(1));
      }
      else {
          return howMany(a, b.substring(1));
      }
   }
   else { return 0; }
}
```

3. [4 puntos] Completar la siguiente implementación de un método para que devuelva el tiempo de ejecución del algoritmo de inserción directa *en su caso peor*, para una talla `t` y un número de repeticiones `numR`.

   Código a completar:

```
public static int measuring(int t, int numR) {
   int[] a = createArray(t);
   int tt = 0;
   for (int r = 0; r < numR; r++) { ... }
   double tPeor =  tt / (double) numR;
   return tPeor;
}
```

   NOTA: Considerar implementado (y disponible *en la misma clase* que el método `measuring`), el método que rellena un array con valores en orden descendente, cuyo perfil es:

```
private static void fillArraySortedInDescendingOrder(int[] a);
```

   Considerar implementado (y disponible en la clase `MeasurableAlgorithms`), el método que ordena un array mediante inserción directa, cuyo perfil es:

```
public static void insertionSort(int[] a);
```

Solución:

```
public static int measuring(int t, int numR) {
   int[] a = createArray(t);
   int tt = 0;
   for (int r = 0; r < numR; r++) {
       fillArraySortedInDescendingOrder(a);
       int ti = System.nanoTime();
       MeasurableAlgorithms.insertionSort(a);
       int tf = System.nanoTime();
       tt += (tf - ti);
   }
   double tPeor =  tt / (double) numR;
   return tPeor;
}
```