**Felipe Brahm**
*lebrahm@uc.cl*

*Pontificia Universidad Católica de Chile*
*Escuela de Ingeniería*
*Departamento de Ciencia de la Computación*

# SOLUCIÓN AYUDANTÍA 8:
## *Ordenamiento y Búsqueda*

*IIC1102 – Introducción a la Programación – Sección 4*

### PROBLEMAS

**1.** *Escriba un método que ordene un arreglo de enteros por* **selección** *y otro método que lo haga por* **inserción.** *Luego analice qué método es más eficiente si recibimos el siguiente arreglo: {7, 8, 3, 1}*

### SOLUCIÓN:

#### POR SELECCIÓN:

```
public static void ordenarPorSeleccion(int[] arreglo) {

      int N = arreglo.length;
      int posMayor, aux;

      while (N > 1) {

            posMayor = 0;
            for (int i = 1 ; i < N ; i++) {
                  if (arreglo[i] > arreglo[posMayor])
                        posMayor = i;
            }
            aux = arreglo[N - 1];
            arreglo[N - 1] = arreglo[posMayor];
            arreglo[posMayor] = aux;
            N--;

      }
}
```

#### POR INSERCIÓN:

```
public static void ordenarPorInsercion(int[] arreglo) {

      int i, k, aux;

      for(i = 1; i < arreglo.length; ++i) {

            aux = arreglo[i];
            k = i-1;

            while(k >= 0 && arreglo[k] > aux) {
                  arreglo[k+1] = arreglo[k];
                  k = k-1;
            }

            arreglo[k+1] = aux;

      }

}
```

**2.** *I4 - Primer Semestre 2005 - Pregunta 1*

*Las patentes (de automóviles), por ejemplo PL7812, están compuestas por un string de dos caracteres, en el ejemplo "PL", y por un número entero de cuatro dígitos, en el ejemplo 7812. Suponga que la clase* **Patente** *tiene la siguiente forma:*

```
public class Patente {
      private String letras;
      private int numero;
```

```
      public Patente(…) {…}
      public String obtLetras() { return letras; }
      public int obtNumero() { return numero; }
}
```

*Suponga también que la clase* **TablaDePatentes** *tiene la siguiente forma:*

```
public class TablaDePatentes {
      private String[] tabla;
      public TablaDePatentes() { tabla = new String[9999]; }

      public boolean buscar(Patente patente) {…}

      … otros métodos …
}
```

*La idea es que* **TablaDePatentes** *almacena en el atributo tabla todas las patentes autorizadas a estacionarse en el campus San Joaquín. En particular, si la patente PL7812 está autorizada, entonces tabla[7812] = "PL", y si la patente JK2345 está autorizada, entonces tabla[2345] = "JK". Además, si dos o más patentes autorizadas tienen el mismo número, entonces sus pares de letras aparecen consecutivamente en el string correspondiente de tabla. Por ejemplo, si las patentes PL7812 y MB7812 están ambas autorizadas, entonces tabla[7812] = "PLMB"; y si las patentes JK2345, RC2345 y DW2345 están todas autorizadas, entonces tabla[2345] = "JKRCDW".*

*Escriba el método* **buscar** *de la clase* **TablaDePatentes**, *que busca rápidamente la patente patente en el atributo tabla, y devuelve true (verdadero) si la patente está en la tabla, y false (falso) en caso contrario.*

**SOLUCIÓN**:

```
public boolean buscar(Patente patente) {

      int num = patente.obtNumero();
      String letras = patente.obtLetras();
      String validas = tabla[num];
      int largo = validas.length();

      int k = 1;
      while (k < largo) {

            if(letras.equals(validas.substring(k-1,k+1)))
                  return true;

            k += 2;
      }

      return false;
}
```

**3**. *I3 - Primer Semestre 2005 - Pregunta 3*

*Se quiere desarrollar un sitio Web para desplegar avisos económicos de arriendo de casas. Como parte de la funcionalidad se desea permitir buscar una propiedad dada su dirección (calle, número y comuna) y desplegar un listado de las propiedades ordenadas por comuna, monto del arriendo, calle y número.*

*Para la implementación ya se han escrito las clases Propiedad y Avisos que poseen los siguientes atributos y métodos:*

**Clase Propiedad**
*Atributos:*

```
      private String calle;  // calle
      private int numero;    // número
      private String comuna; // comuna
      private int arriendo;  // valor del arriendo
```

*Métodos:*

```
      // Constructor: crea una propiedad
            public Propiedad(String calle, int numero,
                             String comuna, int arriendo);
      // getters: permiten acceder a los atributos de la propiedad
            public String getCalle() { return calle; }
```

```
        public int getNumero() { return numero; }
        public String getComuna() { return comuna; }
        public int getArriendo() { return arriendo; }
```

*Clase Avisos*

*Atributos:*

```
    private Propiedad[] arriendos; // arreglo de propiedades
```

*Métodos:*

```
    // Constructor: crea un listado de 'nArriendos' avisos
        public Avisos(int nArriendos);
    // Agrega una propiedad al listado de avisos
        public void AgregarPropiedad(Propiedad prop);
```

*Se pide:*

a) *Implemente el siguiente método de la clase* **Propiedad**:

```
public boolean EsIgual(Propiedad prop);
```

*Retorna* **true** *si la propiedad es igual a la propiedad* **prop** *pasada como parámetro. Considere los siguientes atributos:* **calle**, **numero** *y* **comuna**.

b) *Implemente el siguiente método de la clase* **Avisos**:

```
public int BusquedaPropiedad(Propiedad al);
```

*Retorna la posición de la propiedad buscada dentro del arreglo* **arriendos** *de la clase* **Avisos**, *utilizando el método* **EsIgual** *de la clase* **Propiedad**. *Si la propiedad no existe, retorna -1.*

c) *Implemente el siguiente método de la clase* **Propiedad**:

```
public boolean EsMenor(Propiedad prop);
```

*Retorna* **true** *si la propiedad es menor a la propiedad* **prop** *pasada como parámetro. Considere los siguientes criterios (en el mismo orden):* **comuna**, **arriendo**, **calle** *y* **numero**. *Para el caso de los atributos de tipo String, considere el orden lexicográfico (como en la guía de teléfonos).*

d) *Complete el método* **Ordenar** *de la clase* **Avisos**, *el cual aún no ha sido terminado. Se quiere que este método ordene el arreglo arriendos de la clase* **Avisos** *utilizando el método* **EsMenor** *de la clase* **Propiedad**.

```
// Ordena las propiedades del listado de avisos
public void Ordenar() {
   int N = this.arriendos.length;
   while (N > 1) {
         int posMayor;

          // *** completar ***: encontrar el mayor

         Propiedad aux = this.arriendos[N - 1];
         this.arriendos[N - 1] = this.arriendos[posMayor];
         this.arriendos[posMayor] = aux;

         N--;
   }
}
```

*SOLUCIÓN:*

a)

```
//true si la propiedad tiene la misma calle, número y comuna
public boolean EsIgual(Propiedad prop) {

      if (  this.calle.equalsIgnoreCase(prop.getCalle()) &&
            this.numero == prop.getNumero() &&
            this.comuna.equalsIgnoreCase(prop.getComuna()))

                return true;

      return false;
}
```

**b)**

```
// Busca una propiedad en el listado de avisos
public int BusquedaPropiedad(Propiedad prop) {
  for (int i = 0; i < this.arriendos.length; i++) {
    if (prop.EsIgual(this.arriendos[i])) {
      return i;
    }
  }
  return -1;
}
```

**c)**

```
// true si es menor considerando comuna, arriendo, calle y número
public boolean EsMenor(Propiedad prop) {
  boolean menor;
  menor = false;
  if (this.comuna.compareToIgnoreCase(prop.getComuna()) < 0) {
    menor = true;
  } else if (this.comuna.equalsIgnoreCase(prop.getComuna())) {
    if (this.arriendo < prop.getArriendo()) {
      menor = true;
    } else if (this.arriendo == prop.getArriendo()) {
      if (this.calle.compareToIgnoreCase(prop.getCalle()) < 0) {
        menor = true;
      } else if (this.calle.equalsIgnoreCase(prop.getCalle())) {
        if (this.numero < prop.getNumero()) {
          menor = true;
        }
      }
    }
  }
  return menor;
}
```

**d)**

```
// Buscar el mayor
posMayor = 0;
for (int i = 1; i < N; i++) {
      if (this.arriendos[posMayor].EsMenor(this.arriendos[i]))
      posMayor = i;
}
```