Parcial 4

```
Public int resolver(Grafo<Ciudad> ciudades, String origen, String
destino, int max Controles) {
  if( ciudades.esVacio()) return 0;
  Vertice<Ciudad> inicio = obtenerVertice(origen, ciudades);
  Vertice<Ciudad> fin = obtenerVertice(destino, ciudades);
  if(inicio == null || fin == null) return 0;
  boolean[] marca = new boolean[ciudades.listaDeVertices().tamanio()];
  for(boolean e : marca) e = false;
return resolver(ciudades, inicio, fin, marca, max Controles);
}

Private int resolver(Grafo<Ciudades> grafo, Vertice<Ciudades> actual,
  Vertice<Ciudades> destino, boolean[] marca, int max Controles) {
  int resultado = 0; int aux += actual.tiempo()
  marca[actual.Posicion()] = true;
  if(actual.equals(destino))
    resultado = aux;
  else {
    ListaGenerica<Arista<Ciudades>> ady = grafo.listaDeAdyacentes
                                          (actual);
    Arista<Ciudades> arista;
    ady.comenzar();
    while(!ady.fin()) {
      arista = ady.Proximo();
      if(!marca[arista.verticeDestino().Posicion()] && (arista.Peso <=
        max Controles)) {
        aux += resolver(grafo, arista.verticeDestino(), destino, marca,
                  max Controles);
        if(aux > resultado) resultado = aux;
      }
    }
  }
marca[actual.Posicion()] = false;
return resultado;
}
```

```
Private Vertice<Ciudad> obtenerVertice(String dato, Grafo<Ciudad> grafo){
  Lista Generica<Vertice<Ciudad>> vertices;
  vertices = grafo.listaDeVertices();
  vertices.comenzar(); Vertice<Ciudad> aux;
  while(!vertices.fin()){
    aux = vertices.Proximo();
    if(aux.dato().nombre().equals(dato))
      return aux;
  }
  return null;
}
```

```
Public class Ciudad {
  Private String nombre;
  Private int tiempo;
  Public String nombre() {
    return this.nombre();
  }
  Public int tiempo() {
    return this.tiempo();
  }
}
```