Convenio Representación JAVA: Problem.java

```
package aima.search.framework;
/**
 • Tiene cuatro componentes:
 • 1) Estado inicial.
 • 2) Función sucesor.
 • 3) Test de Objetivo.
 • 4) Coste del camino.
 • 5) Valor de la heurística.
*/
public class Problem {
   protected Object initialState;
   protected SuccessorFunction successorFunction;
   protected GoalTest goalTest;
   protected StepCostFunction stepCostFunction;
   protected HeuristicFunction heuristicFunction;
   protected Problem() {
                    }
```


Coste de los operadores

- (4) Hay que definir la función que define el coste de cada operador
  - Si no se define todo operador tiene coste 1 por defecto
  - Creando una clase que implemente la interfaz aima.search.framework.StepCostFunction
  - Ha de implementar la funcion
    - Double calculateStepCost(Object fromState, Object toState, String action);
    - Las características de la función dependen del problema.
- EJ1: 8-puzzle
  - No es necesario definirla
    - coste = 1 para todos los operadores

Búsqueda heurística

- (5) Hay que definir la función que calcula la heurística h'
  - Creando una clase que implemente la interfaz aima.search.framework.HeuristicFunction
  - Ha de implementar la funcion
    - public double getHeuristicValue(Object n)
  - Las características de la función dependen del problema.
- EJ1: 8-puzzle (2 posibles heurísticas)
  - Distancia de Manhattan
    - ManhattanHeuristicFunction.java
      - `public class ManhattanHeuristicFunction implements HeuristicFunction`
  - Piezas descolocadas
    - MisplacedTileHeuristicFunction.java
      - `public class MisplacedTileHeuristicFunction implements HeuristicFunction`

Ejecutar una Demo de Búsqueda: uso de la clase Problem

EJ1: \ejemplo\ej8p-BusqInformada\src\problemas\demos\EightPuzzleDemo.java

El main(…) llama a este método:

```
private static void eightPuzzleAStarManhattanDemo()
{
   try {
       Problem problem = new Problem(random1,
                       new EightPuzzleSuccessorFunction(),
                       new EightPuzzleGoalTest(),
                       new ManhattanHeuristicFunction());
              Search search = new AStarSearch(new GraphSearch());
              SearchAgent agent = new SearchAgent(problem, search);
                       printActions(agent.getActions());
              printInstrumentation(agent.getInstrumentation());
       } catch (Exception e) { e.printStackTrace();}
   }
```

- También se podría hacer con MisplacedTileHeuristicFunction

Ejecutar una Demo de Búsqueda: uso de la clase Problem

EJ1: \ejemplo\ej8p-BusqInformada\src\problemas\demos\EightPuzzleDemo.java

El main(…) llama a este método:

```
private static void eightPuzzleGreedyBestFirstDemo()
{
   try {
       Problem problem = new Problem(random1,
                    new EightPuzzleSuccessorFunction(),
                    new EightPuzzleGoalTest(),
                    new MisplacedTileHeuristicFunction());
       Search search = new GreedyBestFirstSearch(new GraphSearch());
       SearchAgent agent = new SearchAgent(problem, search);
       printActions(agent.getActions());
       printInstrumentation(agent.getInstrumentation());
       } catch (Exception e) { e.printStackTrace(); }
}
```

- También se podría hacer con ManhattanHeuristicFunction

Búsqueda heurística

- Qué hay en el material en el Campus Virtual:
  - Ejemplo 8-puzzle: ej8p-BusqInformada.rar
    - \ej8p-BusqInformada
      - Carpeta con lo necesario para ejecutar y crear ejemplos con AIMA
    - Cómo se instala? => proyecto eclipse