الغاية من الجلسة:

برمجة مسألة n_puzzle

آلية العمل :

نحن نعلم أننا بحاجة لتعريف مجموعة من الصفوف لتوصيف شامل للمسألة وبداية مع صف الحالة للمسألة.

```
package examples.nPuzzles;
import core.State;

public class PuzzleBoard implements State{
@Override
public int[][] tiles;
public int boardSize;
public int emptyI;
public int emptyJ;
public PuzzleBoard(int[][] tiles, int boardSize, int emptyI, int
emptyJ){
this.tiles = tiles;
this.boardSize = boardSize;
this.emptyI = emptyI;
this.emptyJ = emptyJ;}

public boolean isGoal() {return false;}
@Override
public String toString(){
String result = "";
for(int i = 0; i< boardSize; i++){
for(int j = 0; j < boardSize; j++){
result += board[i][j] + " ";}
result += "\n";}
return result;
}
@Override
public boolean equals(Object s){ PuzzleBoard state =
(PuzzleBoard) s;
for(int i = 0; i< boardSize; i++){
for(int j = 0; j < boardSize; j++){
```

```
if(tiles[i][j] != state.tiles[i][j]){
return false;}}}
return true;}}}
```

صف الحالة PuzzleBoard يحتاج لمصفوفة ثنائية board [][]int تمثل الرقعة فقط.

صف Action:

وفي هذه المسألة سنمثل الحركات الأربعة الممكنة وهي (Move up,MoveDown ,Move left, Move right)

وبالتالي نحن بحاجة لبناء أربع صفوف.

```
package examples.nPuzzles;
import core.IAction;
import core.State;
import java.util.ArrayList;
import java.util.Collection;
public class MoveUp implements IAction{
@Override
public Collection<State> apply(State s) {
Collection <State> nextStates = new ArrayList<>();
PuzzleBoard state = (PuzzleBoard) s;
int [][] newTiles = new int[state.boardSize][state.boardSize];
for(int i = 0; i < state.boardSize; i++){
for (int j = 0; j < state.boardSize; j++){
newTiles[i][j] = state.tiles[i][j];}}
if(state.emptyI > 0){
newTiles[state.emptyI][state.emptyJ] = newTiles[state.emptyI -
1][state.emptyJ];
newTiles[state.emptyI - 1][state.emptyJ] = 0;
PuzzleBoard nextState = new PuzzleBoard(newTiles,
state.boardSize, state.emptyI - 1, state.emptyJ);
nextStates.add(nextState);}
return nextStates;}
@Override
public String getName() {
return "Move Up";}}
```

بقية الصفوف يمكن التعبير عنها بنفس الطريقة مع تغير طريقة التعديل على index.

بعد الانتهاء من بناء صف الحالة وصفوف الانتقال سيتم بناء صف المسألة الرئيسي والذي يعبر عن فكرة توصيف المسألة الرئيسية والتوابع المتضمنة في الصف هي :

- تابع الحالة البدائية
- تابع الحالة النهائية
- تابع الانتقالات (يرد محموعة من الانتقالات)
- تابع كلفة الانتقال

ملاحظة:
أثناء عملية توصيف أي مسألة من أجل حلها بإحدى خوارزميات البحث الذكية نحن بحاجة الى تعريف مجموعة الصفوف السابقة وتخصيص الحالة والانتقالات وفقا لكل مسألة.

صف المسألة:

```
package examples.nPuzzles;
import core.IAction;
import core.Problem;
import core.State;
import java.util.ArrayList;
import java.util.List;
public class PuzzleProblem  extends
Problem {
private PuzzleBoard initialState;
private PuzzleBoard goalState;
public PuzzleProblem(){
initialState = new PuzzleBoard(new  int[][]{
{0, 1, 2},
{4, 5, 3},
{7, 8, 6}}, 3, 0, 0);
goalState = new PuzzleBoard(new
int[][]{
{1, 2, 3},
{4, 5, 6},
{7, 8, 0}}, 3, 2, 2);}
@Override
public State getInitialState() {
return initialState;}
public State getGoalState() {
```

```
return goalState;}
public void setInitialState(PuzzleBoard initialState) {
this.initialState = initialState;}
public void setGoalState(PuzzleBoard goalState) {
this.goalState = goalState;}
@Override
public boolean isGoal(State state) {
return state.equals(getGoalState());}
@Override
public List<IAction> getActions() {
return new ArrayList<IAction>() {{
add(new MoveUp());
add(new MoveDown());
add(new MoveLeft());
add(new MoveRight());}};}
@Override
public double getActionCost(State s, IAction a, State ss){
return 1;}}
```