**class** Piece

**types**

**public** Player = **<White>** | **<Black>**; -- Define player again??!

**public** belongTo : Player;

**protected** promoted : **bool** := **false**;

**operations**

**protected** Piece: Player ==> Piece

Piece(p) == belongTo:= p;

**protected** direction: () ==> **int**

direction() == **if** belongTo = **<White>** **then** **return** -1 **else** **return** 1;

**public** validMovement: Cell\*Cell ==> **bool**

validMovement(c1,c2) == **is** **subclass** **responsibility**

**pre** c1 = Board`getInstance().getCell(c1.x,c1.y) **and** c1 = Board`getInstance().getCell(c2.x,c2.y);

**public** capture:() ==> ()

capture() == (**if** belongTo = **<White>** **then** belongTo := **<Black>** **else** belongTo:= **<White>**; promoted:= **false**)

**post** promoted = **false**;

**public** promote:() ==> ()

promote() == promoted:= **true**

**pre** **not** **isofclass**(GoldGeneral, **self**) **and** **not** **isofclass**(King,**self**) **and** **not** promoted;

**public** getPromoted:() ==> **bool**

getPromoted() == **return** promoted;

**protected** validGoldGeneralMove: Cell\*Cell ==> **bool**

validGoldGeneralMove(c1,c2) ==

**return** (c1.x = c2.x **and** **abs**(c1.y-c2.y) = 1) **or** (c1.y = c2.y **and** **abs**(c1.x-c2.x) = 1) **or**

(c1.y = c2.y + direction() **and** **abs**(c1.x-c2.x) = 1)

**pre** c1 = Board`getInstance().getCell(c1.x,c1.y) **and** c1 = Board`getInstance().getCell(c2.x,c2.y);

**protected** validKingMove: Cell\*Cell ==> **bool**

validKingMove(c1,c2) == **return** (**abs**(c1.x-c2.x) = 0 **and** **abs**(c1.y-c2.y) = 1 ) **or** (**abs**(c1.x-c2.x) = 1 **and** **abs**(c1.y-c2.y) = 0 ) **or** (**abs**(c1.x-c2.x) = 1 **and** **abs**(c1.y-c2.y) = 1 )

**pre** c1 = Board`getInstance().getCell(c1.x,c1.y) **and** c1 = Board`getInstance().getCell(c2.x,c2.y);

**Functions**

**protected** dist: **nat**\***nat** -> **nat**

dist(n1,n2) == **abs**(n1-n2);

**protected** intermediateVals: **nat1**\***nat1** -> **set** **of** **nat1**

intermediateVals(x1,x2) == {x1 + (x2-x1)/**abs**(x2-x1),..., x2 + (x1-x2)/**abs**(x1-x2)} **union**

{x2 + (x1-x2)/**abs**(x1-x2),..., x1 + (x2-x1)/**abs**(x2-x1)}

**end** Piece