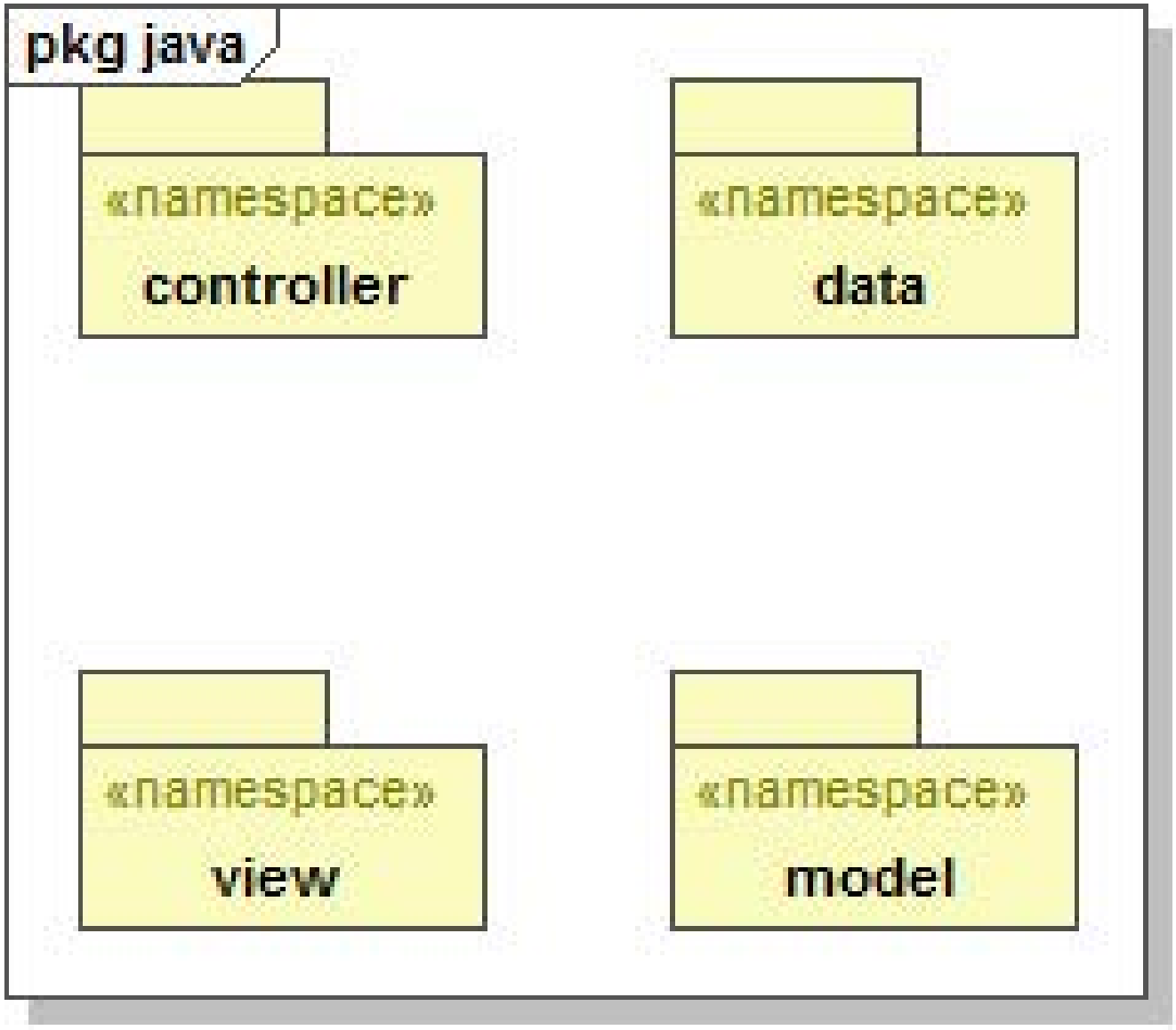
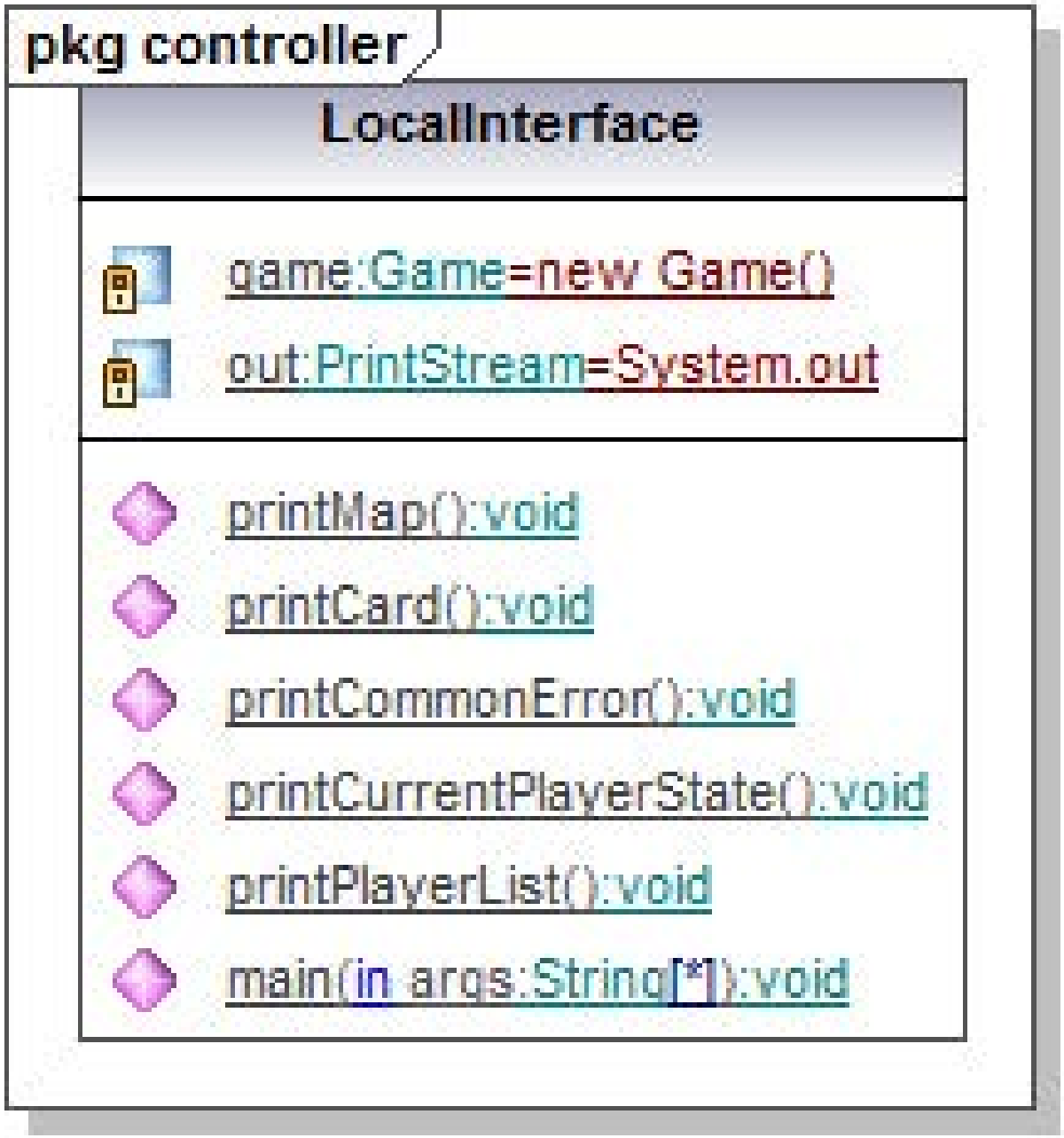

pkg java
«namespace»
controller
«namespace»
data
«namespace»
view
«namespace»
model


pkg controller
LocalInterface
game:Game=new Game()
out:PrintStream=System.out
printMap():void
printCard():void
printCommonError():void
printCurrentPlayerState():void
printPlayerList():void
main(in args:String[*]):void

pkg data

«interface»
Constants

- NORTH:int=0
- SOUTH:int=1
- WEST:int=2
- EAST:int=3
- FILE_POSITION:String="src/main/java/data/carcassonne.txt"
- MAX_PIECE:int=7
- SIDE_NUM:int=4
- LINK_NUM:int=6
- MAX_GIOCATORI:int=5
- MIN_GIOCATORI:int=2
- STREET:int=0
- CITY:int=1
- VOID:int=2
- NS:int=0
- NE:int=1
- NW:int=2
- WE:int=3
- SE:int=4
- SW:int=5
- SIDENORD:int=2
- SIDESUD:int=6
- SIDEOVEST:int=10
- SIDEEST:int=14
- LINKNS:int=19
- LINKNE:int=24
- LINKNW:int=29
- LINKWE:int=34
- LINKSE:int=39
- LINKSW:int=44
- NORDSUD:String="NS"
- NORDEST:String="NE"
- NORDOVEST:String="NW"
- OVESTEST:String="OE"
- SUDEST:String="SE"
- SUDOVEST:String="SW"
- NOTHING:String=""
- MAX_PLAYERS:int=5
- RED:int=0
- BLUE:int=1
- GREEN:int=2
- YELLOW:int=3
- BLACK:int=4
- ZERO_LINE:int=0
- UNO_LINE:int=1
- DUE_LINE:int=2
- TRE_LINE:int=3
- QUATTRO_LINE:int=4
- CINQUE_LINE:int=5
- SEI_LINE:int=6
- POINTS_STREET:int=1
- POINTS_CITY:int=2
- COLOUMN_SIGN_POSITION:int=30
- ROW_SIGN_POSITION:int=36
- COLOUMN_POSITION:int=31
- ROW_POSITION:int=37
- INDEX_1_START:int=2
- INDEX_1_STOP:int=6
- INDEX_2_START:int=9
- INDEX_2_STOP:int=13
- INDEX_3_START:int=16
- INDEX_3_STOP:int=20
- INDEX_4_START:int=23
- INDEX_5_STOP:int=26
- COLUMN:int=4
- ROW:int=5
- LINE_NUMBER:int=6
- ROW_POINTS_PLUS:String=".....+"
- RIGA_CANCELLETTI_PIU:String="##########+"
- RIGA_NIENTE:String=" "
- ROW_VOID_POINT:String=" ."
- ROW_VOID_PLUS:String=" +"
- ROW_PLUS_POINTS_PLUS:String="+.....+"
- ROW_PLUS_VOID_POINTS:String=". ."
- TIMEOUT_START:int=5000
- TIMEOUT_CARD:int=2500

Coordinate

- row:int
- column:int

- «constructor» Coordinate(in column:int, in row:int)
- equals(in coordinate:Object):boolean
- hashCode():int
- getRow():int
- getColumn():int

Methods

- getIndexSideType(in side:char):int
- getNameSydeType(in name:int):char
- getIndexLinkType(in index:String):int
- getLinkName(in index:int):String
- getPlayerGame(in index:int):char
- getPlayerLongName(in index:int):String
- getSideName(in index:int):char
- getOppositeSide(in side:int):int
- linkedSides(in side:int):Integer[*]
- getNeighborsCoordinates(in column:int, in row:int):Coordinate[*]
- getLinkedSide(in link:int, in cardinalPoint:int):int

pkg model

Card

sides:Side[*]=new Side[SIDE_NUM]
links:Boolean[*]=new Boolean[LINK_NUM]

«constructor» Card(in string:String)
setSide(in sideKindIndex:int, in sidePositionIndex:int):void
setLink(in linkKindIndex:char, in linkPositionIndex:int):void
getSide(in cardinalPoint:int):Side
getLink(in linkPosition:int):boolean
isLinked(in posizioneCollegamento:String):boolean
rotateCard():void
getLinksArray():ArrayList<T1->Integer>
getLinkFromSide(in cardinalPoint:int):ArrayList<T1->Integer>
string():String[*]

-currentCard
-deck
-crossedCards

PointsAllocator

crossedCards:ArrayList<T1->Card>=new ArrayList<Card>()
isFinished:Boolean
piecedCards:ArrayList<T1->Side>=new ArrayList<Side>()
playerCounter:ArrayList<T1->Integer>=new ArrayList<Integer>()

«constructor» PointsAllocator()
insertCard(in card:Card):boolean
insertSide(in lato:Side):boolean
getLatoList():ArrayList<T1->Side>
setState(in valore:Boolean):void
getState():boolean
getList():ArrayList<T1->Card>
addCards(in cards:ArrayList<T1->Card>):void
addSides(in side:ArrayList<T1->Side>):void
howManyCards():int
getMax():int
assignPoints(in sideKind:int, in players:ArrayList<T1->Player>):void

Deck

deck:ArrayList<T1->Card>=new ArrayList<Card>()

«constructor» Deck()
getInitialCard():Card
getCard():Card
remanentCards():int

-deck

Game

map:HashMap<T1->Coordinate,T2->Card>=new HashMap<Coordinate,Card>()
players:ArrayList<T1->Player>=new ArrayList<Player>()
deck:Deck=new Deck()
minorColumn:int
majorColumn:int
minorRow:int
majorRow:int
currentPlayer:Player
currentCard:Card
currentColumn:int
currentRow:int

«constructor» Game()
getCurrentPlayer():int
getPlayersNumber():int
getCurrentPlayerObject():Player
getCurrentPlayerScore():int
getCurrentPlayerPiecesNumber():int
insertPlayer():int
setNextPlayer():void
setCurrentCard():void
getRemainingCards():int
setCurrentCard(in tessera:Card):void
setCurrentCoordinate(in column:int, in row:int):void
checkCurrentCardMapInsertability():boolean
checkCurrentCardInsertability():boolean
insertCurrentCard():void
checkInsertabilityPosition(in column:int, in row:int, in card:Card):boolean
isApplicable(in card:Card):boolean
rotateCurrentCard():void
insertCard(in column:int, in row:int, in card:Card):void
checkPiecePresence(in provenienceColumn:int, in provenienceRow:int, in provenienceCardinalPoint:int):boolean
pieceInsertable():Boolean
pieceInsertionSituation():Boolean[*]
insertPiece(in puntoCardinale:int):void
checkFinishObject(in provenienceColumn:int, in provenienceRow:int, in provenienceCardinalPoint:int):PointsAllocator
assignPoints():Boolean
finalOperations(in provenienceColumn:int, in provenienceRow:int, in provenienceCardinalPoint:int, in provenienceCard:Card):PointsAllocator
returnStringedMap():ArrayList<T1->String[]>
riturnStringedCard():String[*]
assignFinalPoints():void

-piecedCardsides
*

Side

sideKind:int
piece:int

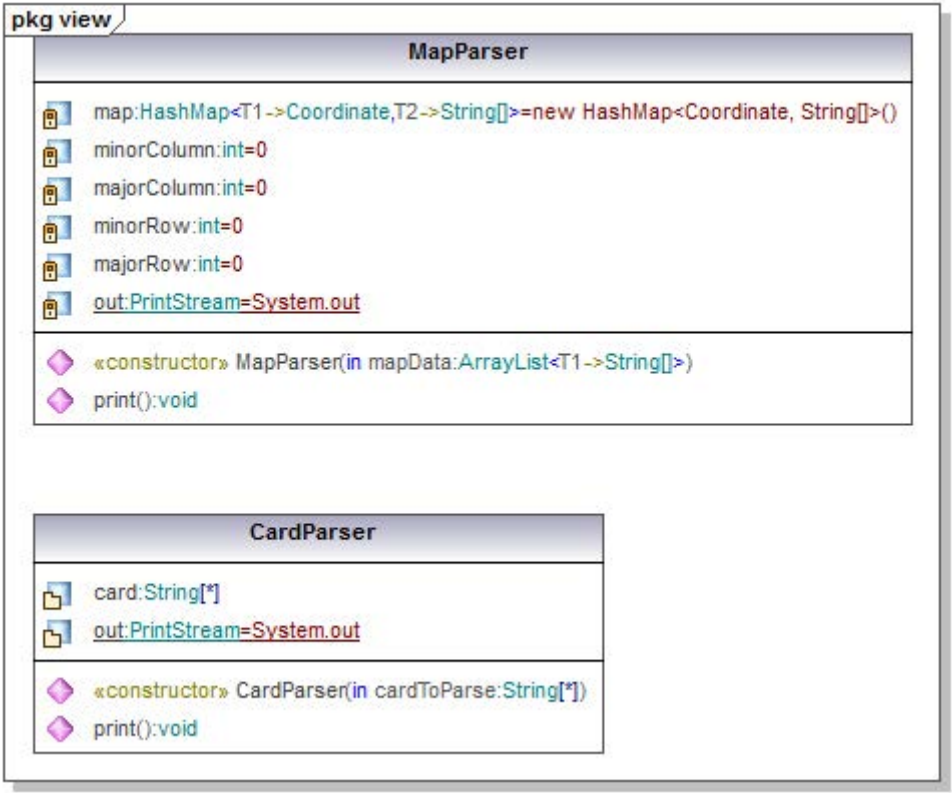
«constructor» Side(in sideKind:int)
getSideKind():int
putPiece(in player:Player):boolean
removePiece(in player:Player):boolean
containsPiece():boolean
getPlayerPiece():int
putPlayerPiece(in color:int):void
containsPlayerPiece(in color:int):boolean
string():String

-players
Player

Player

color:int
score:int
pieces:int

«constructor» Player(in color:int)
getColor():int
getScore():int
getPieces():int
addPiece():void
subPiece():void
addPoints(in points:int):void


pkg view
MapParser
map:HashMap<T1->Coordinate,T2->String[]>=new HashMap<Coordinate, String[]>()
minorColumn:int=0
majorColumn:int=0
minorRow:int=0
majorRow:int=0
out:PrintStream=System.out
«constructor» MapParser(in mapData:ArrayList<T1->String[]>)
print():void
CardParser
card:String[*]
out:PrintStream=System.out
«constructor» CardParser(in cardToParse:String[*])
print():void