**BoardState.java (Class)**

**BEFORE:**

public int checkRow(int targetRow){

int result = -1;

try{

result = rows[targetRow - 1];

}catch(ArrayIndexOutOfBoundsException e){

}

return result;

}

**AFTER:**

public int checkRow(int targetRow){

int result = 0;

try{

result = rows[targetRow - 1];

}catch(ArrayIndexOutOfBoundsException e){

result = invalidRowCode;

}

return result;

}

- Updated checkRow(public method) to use invalidRowCode instead of -1 ("magic number")

**BEFORE:** (no such variables before)

**AFTER:**

public static final int invalidRowCode = -1;

public static final int noneSelected = -2;

- Added invalidRowCode class constant (value = -1)

- Added noneSelected class constant (value = -2)

**BEFORE:**

public int[] compareStates(BoardState stateOne, BoardState stateTwo){

int[] resultRows = new int[stateOne.rows.length];

for(int i = 0; i < resultRows.length; i++){

resultRows[i] = stateOne.rows[i] - stateTwo.rows[i];

int[] rowDifferences = new int[stateOne.rows.length];

}

return resultRows;

}

**AFTER:**

public int[] compareStates(BoardState stateOne, BoardState stateTwo){

int[] rowDifferences = new int[stateOne.rows.length];

for(int i = 0; i < rowDifferences.length; i++){

rowDifferences[i] = stateOne.rows[i] - stateTwo.rows[i];

}

return rowDifferences;

}

- Updated compareStates(public method) -

- renamed resultRows (variable) to rowDifferences for readability.

**BEFORE:**

public TurnAction howToReachState(BoardState stateToReach){

TurnAction action = null;\

int[] stateDifference = compareStates(this, stateToReach);

int selectedRow = -1;

boolean movePossible = true;

for(int i = 0; i < stateDifference.length; i++){

if(stateDifference[i] > 0 && selectedRow == -1){

selectedRow = i;

}else if(stateDifference[i] > 0 || stateDifference[i] < 0 || (selectedRow == -1 && i == stateDifference.length - 1)){

movePossible = false;

}

}

if(movePossible){

System.out.println("Move is possible");

action = new TurnAction();

action.setTargetRow(selectedRow + 1);

action.setTokenAmount(stateDifference[selectedRow]);

}

return action;

}

**AFTER:**

public TurnAction tryToReachState(BoardState stateToReach){

TurnAction suggestedAction = null;

int[] stateComparisonResults = compareStates(this, stateToReach);

int selectedRow = noneSelected;

boolean movePossible = true;

for(int i = 0; i < stateComparisonResults.length; i++){

boolean noStateDifference = (selectedRow == noneSelected && i == stateComparisonResults.length - 1);

boolean isFirstStateDifference = (stateComparisonResults[i] > 0 && selectedRow == noneSelected);

if(isFirstStateDifference){

selectedRow = i;

}else if(representsStateDifference(stateComparisonResults[i]) || noStateDifference){

movePossible = false;

}

}

if(movePossible){

suggestedAction = new TurnAction();

suggestedAction.setTargetRow(selectedRow + 1);

suggestedAction.setTokenAmount(stateComparisonResults[selectedRow]);

}

return suggestedAction;

}

public boolean representsStateDifference(int testedValue){

boolean representsDifference = (testedValue != 0);

return representsDifference;

}

- Updated howToReachState(public method)

- renamed method to tryToReachState for readability

- updated to use noneSelected instead of -2 ("magic number")

- renamed variable (stateDifferences) to stateComparisonResults for readability

- added supporting predicate (representsStateDifference) for readability

- added variable isFirstStateDifference(boolean) for readability

**GameEngine.java(class)**

**BEFORE:**

public void startMenu() {

boolean userDone = false;

Game initGame;

Scanner sc = Main.Scan;

int pvpCode = 1;

int pvcCode = 2;

int cvcCode = 3;

while (!userDone) {

System.out.println("Welcome to NIM. Choose game option:");

System.out.println("1 - Player vs. Player \n"

+ "2 - Player vs. Computer \n"

+ "3 - Computer vs. Computer \n" + "Any Other Value - Exit");

try {

int answer = Integer.parseInt(sc.next());

if (answer == 1) {

initGame = new Game(GameType.PvP);

initGame.gameLoop();

} else if (answer == 2) {

initGame = new Game(GameType.PvC);

initGame.gameLoop();

} else if (answer == 3) {

System.out.println("How many games should the computers play?");

int rounds = sc.nextInt();

. . .

**AFTER:**

public void startMenu() {

boolean userDone = false;

Game initGame;

Scanner sc = Main.Scan;

int pvpCode = 1;

int pvcCode = 2;

int cvcCode = 3;

while (!userDone) {

System.out.println("Welcome to NIM. Choose game option:");

System.out.println(pvpCode + " - Player vs. Player \n"

+ pvcCode + " - Player vs. Computer \n"

+ cvcCode + " - Computer vs. Computer \n" + "Any Other Value - Exit");

try {

int answer = Integer.parseInt(sc.next());

if (answer == pvpCode) {

initGame = new Game(GameType.PvP);

initGame.gameLoop();

} else if (answer == pvcCode) {

initGame = new Game(GameType.PvC);

initGame.gameLoop();

} else if (answer == cvcCode) {

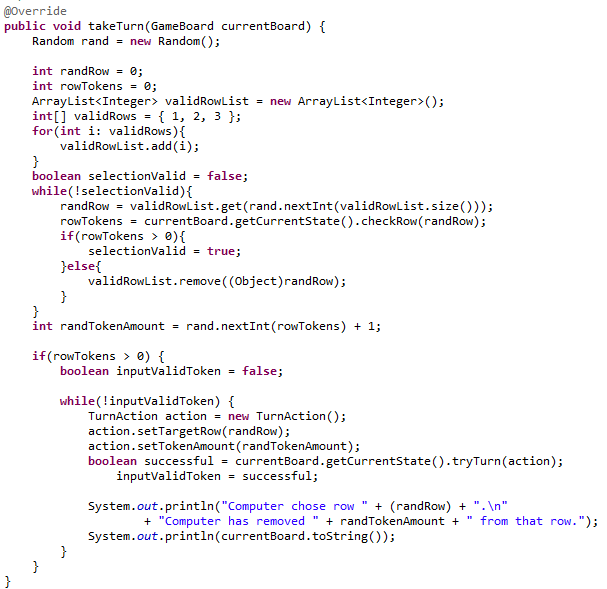
System.out.println("How many games should the computers play?");

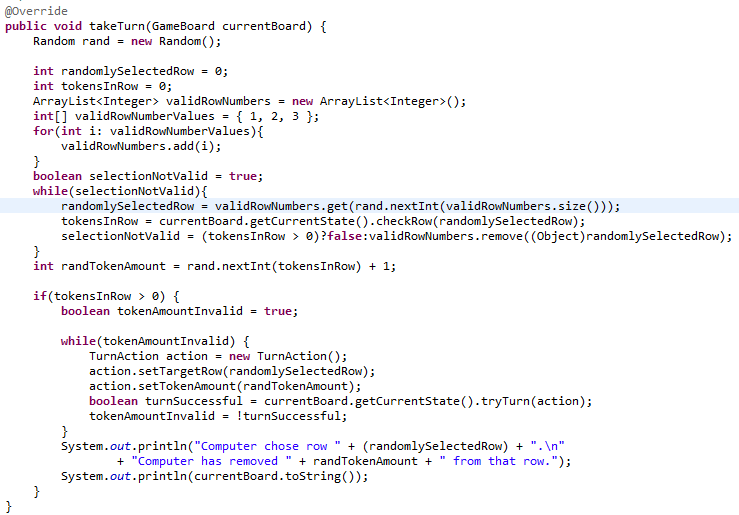
int rounds = sc.nextInt();

. . .

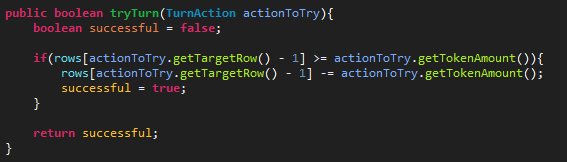
* added code variables to avoid magic numbers and aid readability

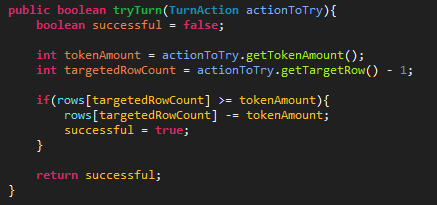
**Computer.java(class)**

 **BEFORE:**

**AFTER:**

**Human.java**

//BEFORE:

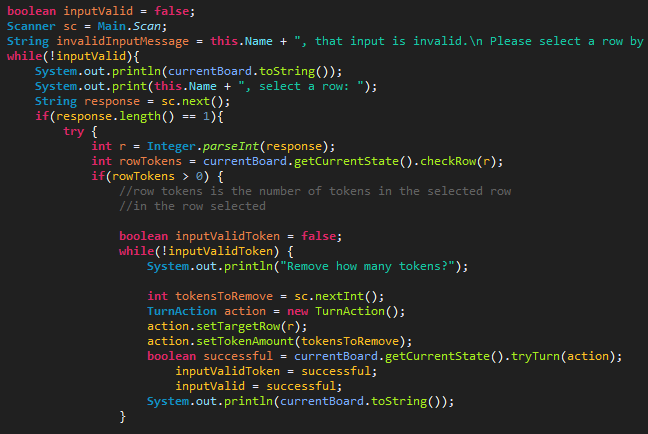
//AFTER:

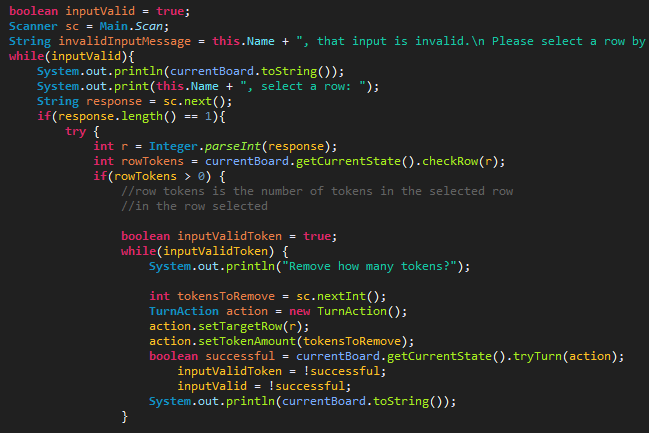
//EXPLANATION:

Rule: Predicate, Chapter 9: Metaphor based rules.

Before: The code was using redundant variables, and the readability for the code was off. I put

the redundant data into variables and made their meaning more obvious.

//BEFORE:

//AFTER:

//EXPLANATION:

Rule: Complements from clauses, Chapter 7: Direct Expression.

Before: I thought that if if-then-else statements should remove the complementary clause, then a while loop should have it removed as well.