**Program:**

|  |  |
| --- | --- |
| import java.util.HashSet;  import java.util.LinkedList;  class WJ {  HashSet<State> uniqueStates;  void letsRoll() {  int jug1 = 5;  int jug2 = 4;  int amtNeeded = 2;  State initialState = new State(0, 0);  State finalState = new State(amtNeeded, 0);  State finalPath = null;  uniqueStates = new HashSet<>();  LinkedList<State> queue = new LinkedList<>();  queue.add(initialState);  while (!queue.isEmpty()) {  State currState = queue.poll();  if (currState.equals(finalState)) {  finalPath = currState;  break;  }  if (currState.x == 0) {  State nextState = new State(jug1, currState.y, currState);  checkUniqueStates(uniqueStates, nextState, queue);  }    if (currState.y == 0) {  State nextState = new State(currState.x, jug2, currState);  checkUniqueStates(uniqueStates, nextState, queue);  }    if (currState.x > 0) {  State nextState = new State(0, currState.y, currState);  checkUniqueStates(uniqueStates, nextState, queue);  }  if (currState.y > 0) {  State nextState = new State(currState.x, 0, currState);  checkUniqueStates(uniqueStates, nextState, queue);  }    if (currState.x > 0 && currState.y < jug2) {  int amtToTransfer = Math.min(currState.x, jug2 - currState.y);  State nextState = new State(currState.x - amtToTransfer, currState.y  + amtToTransfer,  currState);  checkUniqueStates(uniqueStates, nextState, queue);  } | if (currState.y > 0 && currState.x < jug1) {  int amtToTransfer = Math.min(currState.y, jug1 - currState.x);  State nextState = new State(currState.x + amtToTransfer, currState.y  - amtToTransfer,  currState);  checkUniqueStates(uniqueStates, nextState, queue);  }  }  if (finalPath != null) {  System.out.println("J1 J2");  System.out.println(finalPath);  }  else{  System.out.println("Not Possible");  }  }  void checkUniqueStates(HashSet<State> uniqueStates, State toCheck,  LinkedList<State> queue) {  if (!uniqueStates.contains(toCheck)) {  uniqueStates.add(toCheck);  queue.add(toCheck);  }  }  public static void main(String[] args) {  new WJ().letsRoll();  }  }  class State {  int x;  int y;  State pre;  public State(int x, int y) {  super();  this.x = x;  this.y = y;  }  public State(int x, int y, State pre) {  super();  this.x = x;  this.y = y;  this.pre = pre;  }  @Override  public int hashCode() {  final int prime = 31;  int result = 1;  result = prime \* result + x;  result = prime \* result + y;  return result;  } |

|  |  |
| --- | --- |
| public boolean equals(Object obj) {  if (this == obj) {  return true;  }  if (obj == null) {  return false;  }  if (getClass() != obj.getClass()) {  return false;  }  State other = (State) obj;  if (x != other.x) {  return false; | if (y != other.y) {  return false;  }  return true;  }  @Override  public String toString() {  StringBuilder builder = new StringBuilder();  if (pre != null) {  builder.append(pre);  }  builder.append(x);  builder.append(" ").append(y).append("\n");  return builder.toString();  }  } |

**Output:**

