```java
import java.util.ArrayList;

/*  This class encapsulates player position and direction
 */
public class Player extends Moveable {
	private boolean readyToStart = false;
	private int presses = 0, energy = 200;
	private int roadblockPut = 0;
	private ArrayList<Trap> traps = new ArrayList<Trap>();
	private ArrayList<Roadblock> roadblocks = new ArrayList<Roadblock>();
	private boolean canPutTrap = true, canPutBlock = true;

	public Player(Grid g, int row, int col) throws Exception {
		super(g);
		currentCell = grid.getCell(row, col);
		currentDirection = ' ';
	}

	public Cell move(int presses) {
		boolean canMove = false;
		if ((presses == 1 && energy >= 2) || (presses == 2 && energy >= 6) || 
(presses == 3 && energy >= 14)) {
			canMove = true;
		}
		if (getTrap().size() > 0)
			for (int i = 0; i < getTrap().size(); ++i) {
				getTrap().get(i).decTime();
				if (getTrap().get(i).getTime() == 0) {
					getTrap().remove(i);
				}
			}
		if (canMove) {
			currentCell = grid.getCell(currentCell, currentDirection, 
presses);
			if (currentCell.gotGold) {
				energy += 6;
				currentCell.gotGold = false;
			}
			if (presses == 1) {
				energy -= 2;
			} else if (presses == 2) {
				energy -= 6;
			} else if (presses == 3) {
				energy -= 14;
			}
		}
		if (presses != 0) {
			allowPut();
		}

		clearPress();
		return currentCell;
	}

	// keypress count for moving multiple cells
	public int getPresses() {
		return presses;
	}

	public void setEnergy(int energy) {
		this.energy = energy;
	}

	public int getEnergy() {
		return energy;
	}

	public ArrayList<Roadblock> getBlock() {
        return roadblocks;
    }

    public ArrayList<Trap> getTrap() {
        return traps;
    }

    // eating nougats
    public void eat() {
        energy += 6;
    }

    public void addPress() {
        if (presses < 3) {
            presses += 1;
        }
    }

    public void clearPress() {
        presses = 0;
    }

    public void putTrap() {
        if (canPutTrap && energy >= 50) {
            traps.add(new Trap(grid, currentCell));
            energy -= 50;
            canPutTrap = false;
        }
    }

    public void putBlock() {
        if (canPutBlock && roadblockPut < 3) {
            roadblocks.add(new Roadblock(grid, currentCell));
            roadblockPut += 1;
            canPutBlock = false;
        }
    }

    public void allowPut() {
        canPutTrap = true;
        canPutBlock = true;
    }

    public void setReady(boolean val) {
        readyToStart = val;
    }

    public boolean isReady() {
        return readyToStart;
    }
}
```