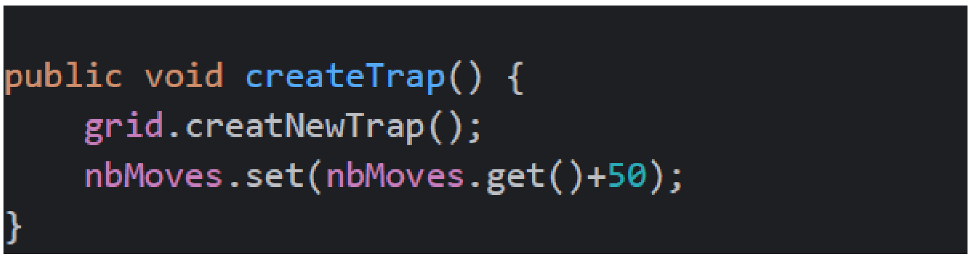
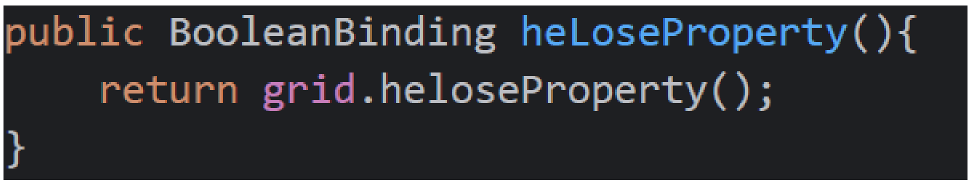
Exemple :

Board4Play :

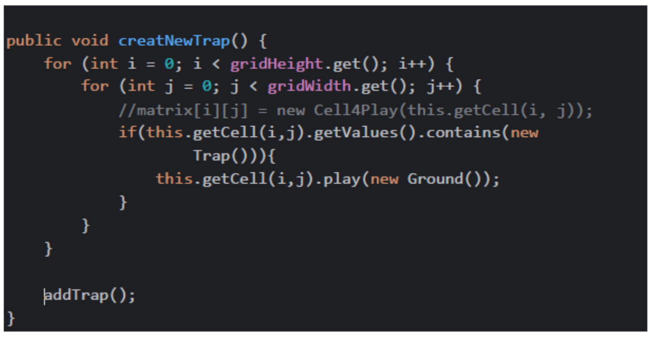


```java
public void createTrap() {
    grid.creatNewTrap();
    nbMoves.set(nbMoves.get()+50);
}
```

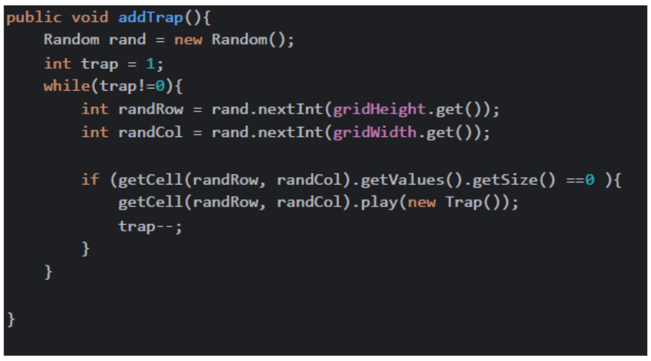


```java
public BooleanBinding heLoseProperty(){
    return grid.heloseProperty();
}
```

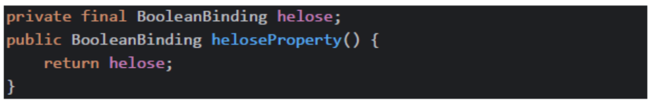
Grid4Play :



```java
public void creatNewTrap() {
    for (int i = 0; i < gridHeight.get(); i++) {
        for (int j = 0; j < gridWidth.get(); j++) {
            //matrix[i][j] = new Cell4Play(this.getCell(i, j));
            if(this.getCell(i,j).getValues().contains(new
                    Trap())){
                this.getCell(i,j).play(new Ground());
            }
        }
    }

    addTrap();
}
```

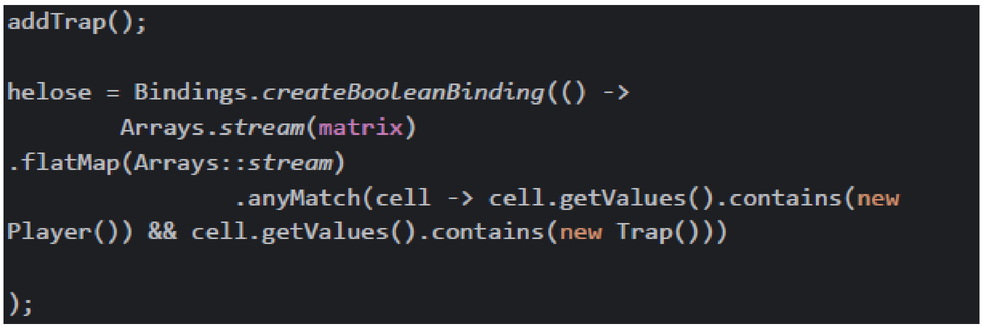


```java
public void addTrap(){
    Random rand = new Random();
    int trap = 1;
    while(trap!=0){
        int randRow = rand.nextInt(gridHeight.get());
        int randCol = rand.nextInt(gridWidth.get());

        if (getCell(randRow, randCol).getValues().getSize() ==0 ){
            getCell(randRow, randCol).play(new Trap());
            trap--;
        }
    }

}
```



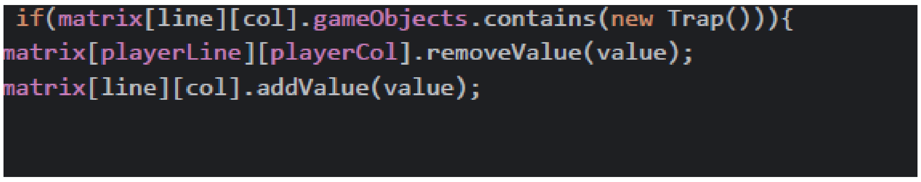
```java
private final BooleanBinding helose;
public BooleanBinding heloseProperty() {
    return helose;
}
```

Dans le constructeur ajoute :

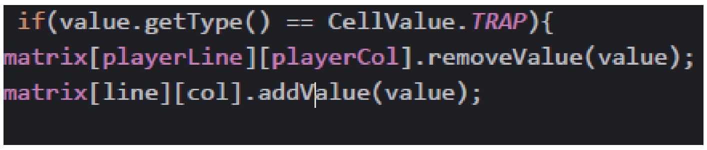


```
addTrap();

helose = Bindings.createBooleanBinding(() ->
        Arrays.stream(matrix)
.flatMap(Arrays::stream)
                .anyMatch(cell -> cell.getValues().contains(new
Player()) && cell.getValues().contains(new Trap()))

);
```

Ajoute une condition

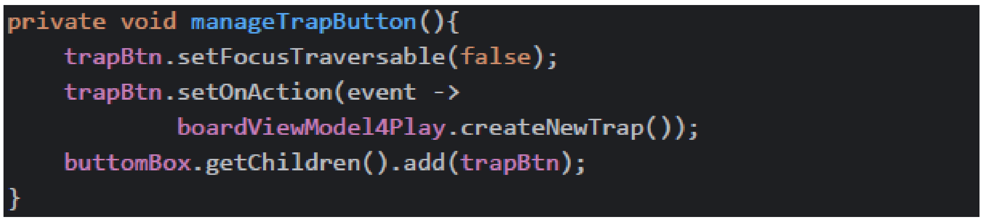


```
 if(matrix[line][col].gameObjects.contains(new Trap())){
matrix[playerLine][playerCol].removeValue(value);
matrix[line][col].addValue(value);
```



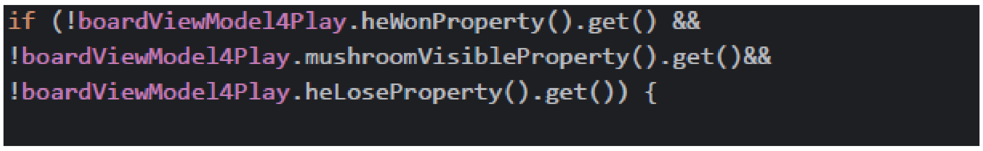
```
 if(value.getType() == CellValue.TRAP){
matrix[playerLine][playerCol].removeValue(value);
matrix[line][col].addValue(value);
```

BoardV4Play :



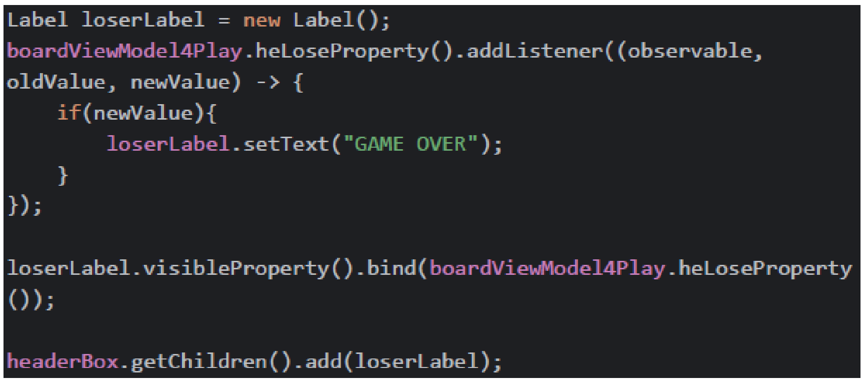
```
private void manageTrapButton(){
    trapBtn.setFocusTraversable(false);
    trapBtn.setOnAction(event ->
            boardViewModel4Play.createNewTrap());
    buttomBox.getChildren().add(trapBtn);
}
```

Condition (méthode)



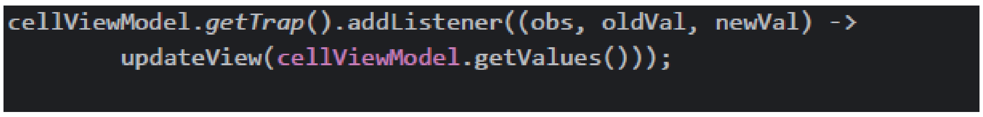
```
if (!boardViewModel4Play.heWonProperty().get() &&
!boardViewModel4Play.mushroomVisibleProperty().get()&&
!boardViewModel4Play.heLoseProperty().get()) {
```

Ajoute dans méthode createheader

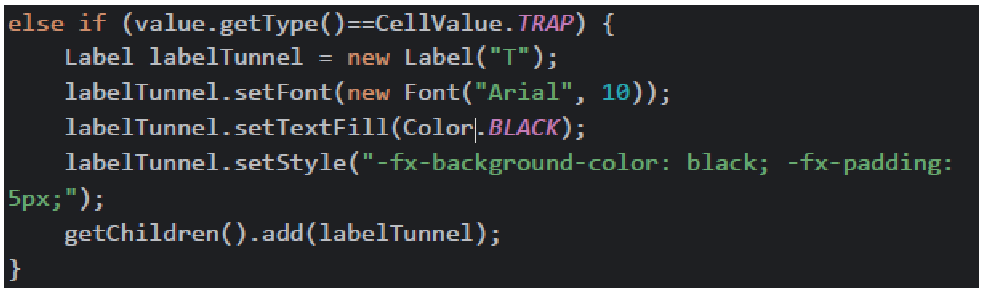


```
Label loserLabel = new Label();
boardViewModel4Play.heLoseProperty().addListener((observable,
oldValue, newValue) -> {
    if(newValue){
        loserLabel.setText("GAME OVER");
    }
});

loserLabel.visibleProperty().bind(boardViewModel4Play.heLoseProperty
());

headerBox.getChildren().add(loserLabel);
```

CellV4Play :

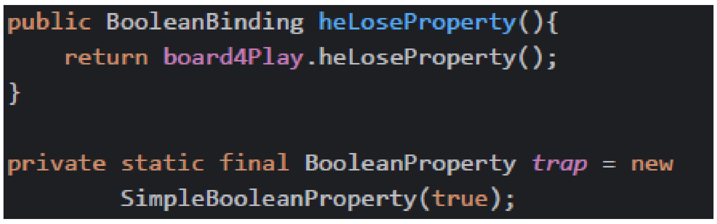


```
cellViewModel.getTrap().addListener((obs, oldVal, newVal) ->
        updateView(cellViewModel.getValues()));
```

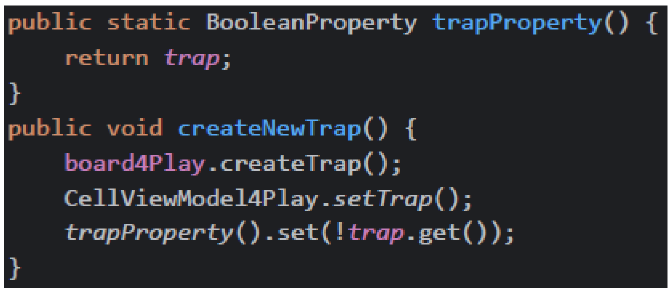


```
else if (value.getType()==CellValue.TRAP) {
    Label labelTunnel = new Label("T");
    labelTunnel.setFont(new Font("Arial", 10));
    labelTunnel.setTextFill(Color.BLACK);
    labelTunnel.setStyle("-fx-background-color: black; -fx-padding:
5px;");
    getChildren().add(labelTunnel);
}
```

BoardVM4Play :

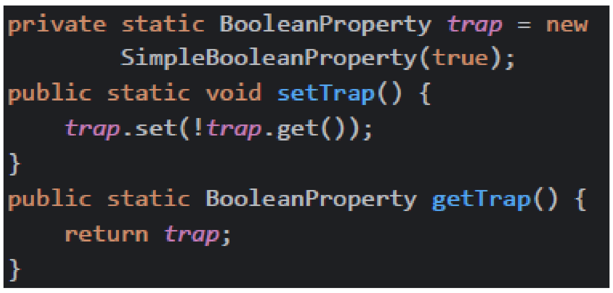


```
public BooleanBinding heLoseProperty(){
    return board4Play.heLoseProperty();
}

private static final BooleanProperty trap = new
        SimpleBooleanProperty(true);
```



```
public static BooleanProperty trapProperty() {
    return trap;
}
public void createNewTrap() {
    board4Play.createTrap();
    CellViewModel4Play.setTrap();
    trapProperty().set(!trap.get());
}
```

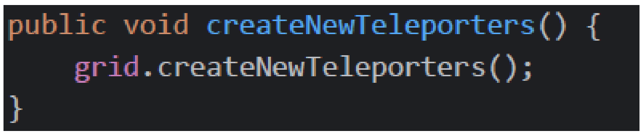
CellVM4Play



```
private static BooleanProperty trap = new
        SimpleBooleanProperty(true);
public static void setTrap() {
    trap.set(!trap.get());
}
public static BooleanProperty getTrap() {
    return trap;
}
```

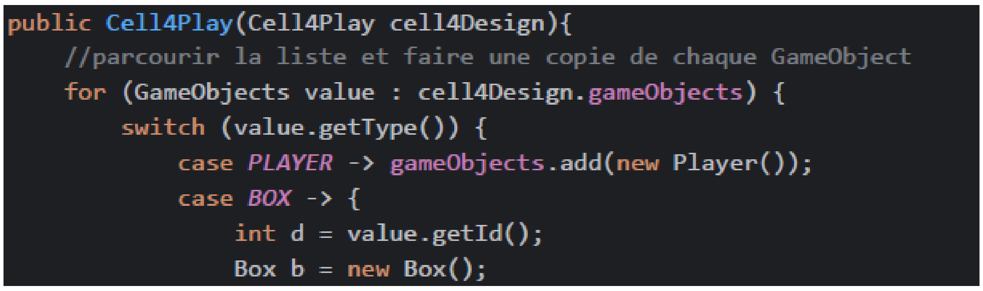
Ex : T

Board4Play

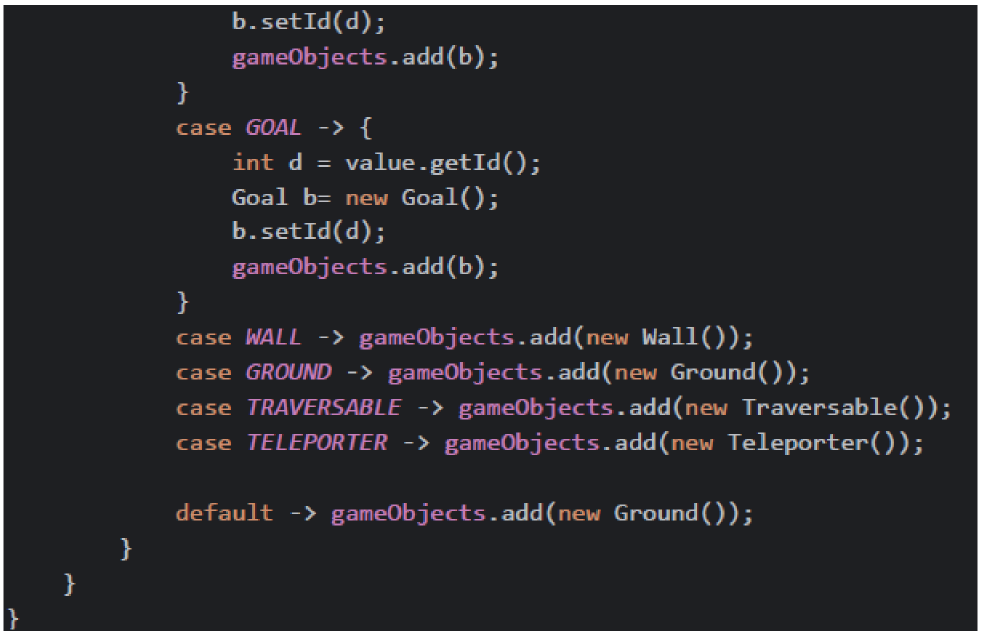


```java
public void createNewTeleporters() {
    grid.createNewTeleporters();
}
```

Cell4Play :

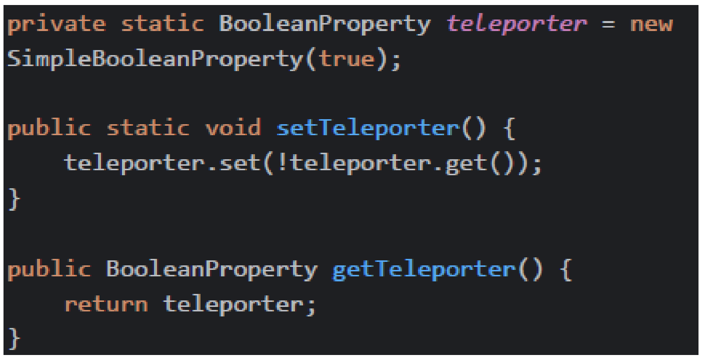


```java
public Cell4Play(Cell4Play cell4Design){
    //parcourir la liste et faire une copie de chaque GameObject
    for (GameObjects value : cell4Design.gameObjects) {
        switch (value.getType()) {
            case PLAYER -> gameObjects.add(new Player());
            case BOX -> {
                int d = value.getId();
                Box b = new Box();
```



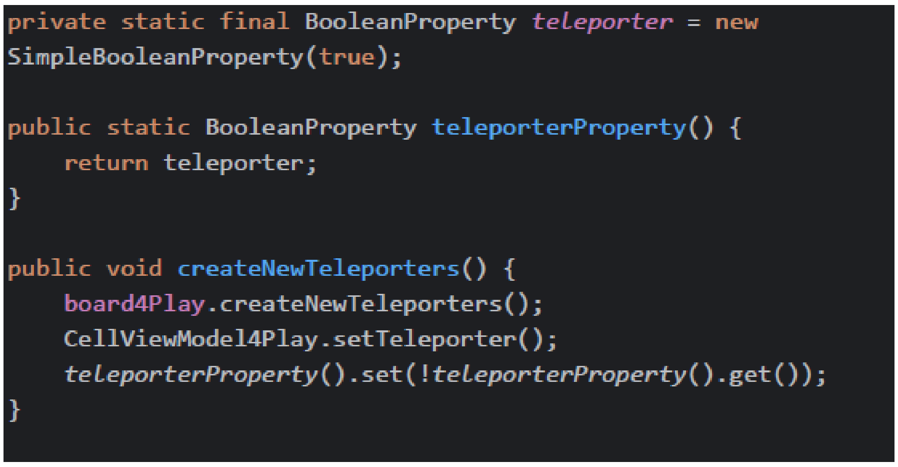
```java
                b.setId(d);
                gameObjects.add(b);
            }
            case GOAL -> {
                int d = value.getId();
                Goal b= new Goal();
                b.setId(d);
                gameObjects.add(b);
            }
            case WALL -> gameObjects.add(new Wall());
            case GROUND -> gameObjects.add(new Ground());
            case TRAVERSABLE -> gameObjects.add(new Traversable());
            case TELEPORTER -> gameObjects.add(new Teleporter());

            default -> gameObjects.add(new Ground());
        }
    }
}
```

CellVM4play :



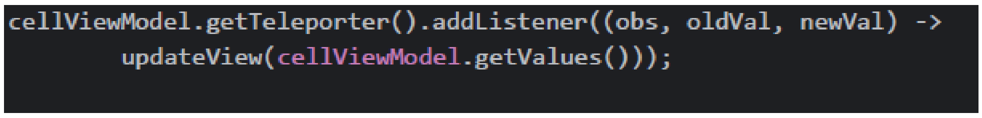
```java
private static BooleanProperty teleporter = new
SimpleBooleanProperty(true);

public static void setTeleporter() {
    teleporter.set(!teleporter.get());
}

public BooleanProperty getTeleporter() {
    return teleporter;
}
```

BVM4P



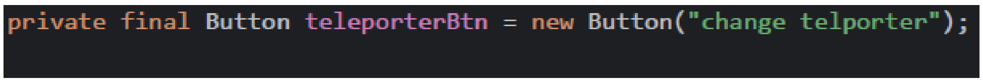
```
private static final BooleanProperty teleporter = new
SimpleBooleanProperty(true);

public static BooleanProperty teleporterProperty() {
    return teleporter;
}

public void createNewTeleporters() {
    board4Play.createNewTeleporters();
    CellViewModel4Play.setTeleporter();
    teleporterProperty().set(!teleporterProperty().get());
}
```

CV4P :

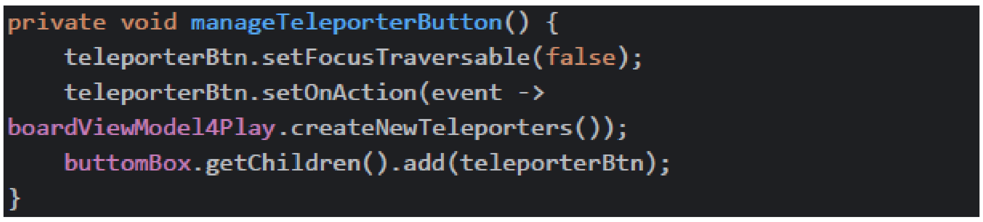


```
cellViewModel.getTeleporter().addListener((obs, oldVal, newVal) ->
        updateView(cellViewModel.getValues()));
```

BV4P :



```
private final Button teleporterBtn = new Button("change telporter");
```



```
private void manageTeleporterButton() {
    teleporterBtn.setFocusTraversable(false);
    teleporterBtn.setOnAction(event ->
boardViewModel4Play.createNewTeleporters());
    buttomBox.getChildren().add(teleporterBtn);
}
```

G4P :

```
private void addTeleporter() {
    Random rand = new Random();
    int tcount = 2;
    int firstcol = 0;
    int firstrow = 0;
    while(tcount!=1){
        int randRow = rand.nextInt(gridHeight.get());
        int randCol = rand.nextInt(gridWidth.get());

        if (getCell(randRow, randCol).getValues().isEmpty()){
            getCell(randRow, randCol).play(new Teleporter());
            firstcol = randCol;
            firstrow = randRow;
            tcount--;
        }
    }
    while(tcount!=0){
        int randRow = rand.nextInt(gridHeight.get());
        int randCol = rand.nextInt(gridWidth.get());

        if (getCell(randRow, randCol).getValues().isEmpty()){

           Teleporter tp = new Teleporter();
```

```
            tp.setPairedCol(firstcol);
            tp.setPairedRow(firstrow);

            getCell(randRow, randCol).play(tp);

            getCell(firstrow,
firstcol).getValues().get(0).setPairedCol(randCol);
            getCell(firstrow,
firstcol).getValues().get(0).setPairedRow(randRow);
            tcount--;
        }
    }
}
```

```
public void createNewTeleporters() {
    for (int i = 0; i < gridHeight.get(); i++) {
        for (int j = 0; j < gridWidth.get(); j++) {

            //matrix[i][j] = new Cell4Play(this.getCell(i, j));
            if(this.getCell(i,j).getValues().contains(new
Teleporter())){
                this.getCell(i,j).play(new Ground());
            }
        }
    }
    addTeleporter();
}
```