**Removed the getTile and drawLevel methods from the Level class.**

**package** sonar.gamestates.states.levels;

**import** sonar.gamestates.Screen;

**import** sonar.gamestates.states.levels.stages.entities.animations.tiles.Tile;

**import** sonar.gamestates.states.levels.stages.entities.animations.tiles.TileManager;

**public** **class** Level

{

//Level retrieves all the tiles that build our level.

**private** LM lm;

**private** LevelBuilder buildLevel;

**protected** Level(LevelBuilder buildLevel, LM lm)

{

**this**.buildLevel = buildLevel;

**this**.lm = lm;

}

**void** update()

{

}

**void** render(**int** xScroll, **int** yScroll, Screen screen)

{

screen.setOffset(xScroll, yScroll);

**if**(lm.getStage().getTmanage() != **null**)

{

**int** x0 = xScroll / lm.getStage().getTmanage().voidTile.getWidth(); //divided by 16

**int** x1 = (xScroll + screen.getWidth() + lm.getStage().getTmanage().voidTile.getWidth()) / lm.getStage().getTmanage().voidTile.getWidth();

**int** y0 = yScroll / lm.getStage().getTmanage().voidTile.getHeight();

**int** y1 = (yScroll + screen.getHeight() + lm.getStage().getTmanage().voidTile.getHeight()) / lm.getStage().getTmanage().voidTile.getHeight();

drawLevel(x0, x1, y0, y1, screen);

}

}

**private** **int** tileColour(**int** x, **int** y)

{

**return** buildLevel.getTiles()[x + y \* buildLevel.getWidth()];

}

LM getLm(){**return** lm;}

LevelBuilder getBuildLevel(){**return** buildLevel;}

}