**Initialized the local inventory variable to a new DualStateBuilder class and also changed the pastState to use the local inventory variable instead of the new DualStateBuilder class in the loadState method that is located in the GSM class.**

**package** sonar;

**import** java.awt.Graphics;

**import** java.awt.image.BufferedImage;

**import** java.awt.image.DataBufferInt;

**import** sonar.gamestates.states.Inventory;

**import** sonar.gamestates.states.MenuState;

**import** sonar.gamestates.states.PasswordState;

**import** sonar.gamestates.states.StarterStage;

**public** **class** GSM

{

//The GSM class that allows us to switch between different gamestates.

**private** **static** Game *game*;

**private** **static** GameState *currentState*, *pastState*;

**private** **static** BufferedImage *image*;

**private** **static** Screen *screen*;

**private** GSM(){} //Prevents multiple gsms from being created

**final** **static** GSM createGSM(**final** Game cGame)

{

*game* = cGame;

*screen* = Screen.*createScreen*(Game.*getWindowWidth*(), Game.*getWindowHeight*());

*image* = **new** BufferedImage(Screen.*getWidth*(), Screen.*getHeight*(), BufferedImage.***TYPE\_INT\_RGB***);

GSM gsm = **new** GSM();

//GSM is readily available here

GameState.*setGSM*(gsm);

*setState*(StateHolder.***menuState***);

**return** gsm;

}

**public** **final** **static** **void** switchStates(**final** GameState active, **final** GameState passive)

{

**if**(*pastState* != **null**)

{

*currentState* = active;

*pastState* = passive;

}

}

//A gsm is in charge of loading and setting states

**private** **final** **static** **void** loadState(**final** **int** state)

{

*currentState* = **null**;

//Menu and build states are the only ones that stay

//Path only exists for single states

//Regardless of state gsm is set

**if**(state == StateHolder.***menuState***) *currentState* = **new** MenuState(**new** SingleStateBuilder("Menu", "/textures/states/Menu.png"));

**if**(state == StateHolder.***passwordState***) *currentState* = **new** PasswordState(**new** SingleStateBuilder("Password", "/textures/states/Password.png"));

**if**(state == StateHolder.***starterStage***) *currentState* = **new** StarterStage(**new** DualStateBuilder("Starter"));

**if**(*currentState*.getBuildState().stateType().equals("Dual")) //BuildState can't be made static

{

//Might be backwards

DualStateBuilder inventory = **new** DualStateBuilder("Inventory");

*pastState* = **new** Inventory(inventory);

DualStateBuilder inv = (DualStateBuilder) *pastState*.getBuildState();

inv.setPath("/textures/states/Inventory.png");

*pastState*.setBuildState(inv);

}

}

**public** **final** **static** **void** setState(**final** **int** state)

{

*loadState*(state);

}

**public** **void** update()

{

*currentState*.update();

}

**public** **void** render(Graphics g)

{

**int** xScroll = 0;

**int** yScroll = 0;

*screen*.clear();

*currentState*.render(xScroll, yScroll, *screen*, g);

**int**[] pixels = ((DataBufferInt) *image*.getRaster().getDataBuffer()).getData();

**for**(**int** i = 0; i < pixels.length; i++)

{

pixels[i] = Screen.*getPixels*()[i];

}

g.drawImage(*image*, 0, 0, *game*.getWidth(), *game*.getHeight(), **null**);

}

**public** **final** **static** Game getGame(){**return** *game*;}

**public** **final** **static** GameState getCurrentState(){**return** *currentState*;}

**public** **final** **static** GameState getPastState(){**return** *pastState*;}

}