**Added the return statement to the createGSM method and also removed the old constructor method since it is no longer needed 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*;

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

{

*game* = cGame;

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

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

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

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

**return** gsm;

}

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

{

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

{

*currentState* = active;

*pastState* = passive;

}

}

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

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

{

*currentState* = **null**;

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

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

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

**if**(*currentState*.getBuildState().stateType().equals("Dual")){*pastState* = **new** Inventory(**new** DualStateBuilder(), "/textures/states/Inventory.png", "Inventory", **this**);}

}

**public** **void** setState(**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** Game getGame(){**return** *game*;}

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

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

}