**Finalized the last remaining code needed for the GSM to be useful by using StateHolder to define the remaining states left**

**package** sonar.gamestates;

**import** java.awt.Graphics;

**import** sonar.Game;

**public** **class** GSM

{

**private** Game game;

**private** **short**[] gameWindow;

**private** GameState currentState, pastState;

**public** GSM(**short**[] gameWindow, Game currentGame)

{

game = currentGame;

**this**.gameWindow = gameWindow;

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

}

**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(), **this**);

**if**(state == StateHolder.***passwordState***) currentState = **new** PasswordState(**new** SingleStateBuilder(), **this**);

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

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

}

**void** setState(**int** state)

{

loadState(state);

}

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

{

currentState.update();

}

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

{

currentState.render(g);

}

Game getGame(){**return** game;}

**short**[] getGameWindow(){**return** gameWindow;}

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

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

}