**Added the GameState.createGameState method call to the Inventory constructor and also removed the super constructor from the Inventory constructor in the Inventory class.**

**package** sonar.gamestates.states;

**import** sonar.GSM;

**import** sonar.GameState;

**import** sonar.Screen;

**import** sonar.StateBuilder;

**import** sonar.gamestates.states.levels.stages.entities.SpriteManager;

**import** sonar.gamestates.states.levels.stages.entities.animations.energies.EnergyManager;

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

**import** sonar.gamestates.states.levels.stages.entities.animations.weapons.WeaponManager;

**public** **class** Inventory **extends** GameState

{

//A class that will hold all energy for the player.

**private** WeaponManager wmanage;

**private** EnergyManager emanage;

**public** Inventory(StateBuilder buildState) //Extends the GameState class

{

GameState.*createGameState*(buildState);

}

**protected** **void** update()

{

**if**(getKey() == **null**)

{

initKey();

setSmanage(**new** SpriteManager(getIdentity()));

setTmanage(**new** TileManager(getSmanage()));

wmanage = **new** WeaponManager(getSmanage());

emanage = **new** EnergyManager(getSmanage());

}

getKey().update();

**if**(getKey().a)

{

resetKeyboard();

resetSmanage();

resetTmanage();

wmanage = **null**;

emanage = **null**;

GSM.*switchStates*(GSM.*getPastState*(), GSM.*getCurrentState*());

}

}

**public** **void** renderEnergies(Screen screen)

{

**if**(emanage != **null**)

{

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

{

**if**(emanage.getEnergies()[i] != **null**) emanage.getEnergies()[i].render(screen);

}

}

}

**public** **void** renderWeapons(Screen screen)

{

**if**(wmanage != **null**)

{

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

{

**if**(wmanage.getWeapons()[i] != **null**) wmanage.getWeapons()[i].render(screen);

}

}

}

}