**Called gsm.update() from the update method and the gsm.render(g) from the render method so that we can update and render objects from the GameStateManager.**

**package** sonar;

**import** java.awt.Graphics;

**import** java.awt.image.BufferStrategy;

**import** sonar.gamestates.GSM;

**class** Mobile

{

**private** **short**[] gameKeeper;

**private** BufferStrategy bs;

**private** Game game;

**private** String title;

**private** **double** renderTime;

**private** **double** previousGameWorldTime = 0;

**private** **double** oneSecond = 1000;

**private** **boolean** renderFrame;

**private** **boolean** coolDown = **false**;

**private** **boolean** skipFrame = **false**;

**private** GSM gsm;

Mobile(Game game, String title)

{

**this**.game = game;

**this**.title = title;

gameKeeper = **new** **short**[2];

gsm = **new** GSM();

}

**void** loop()

{

**if**(!coolDown)

{

**double** timeInGameWorld = convertNanosToMillis(System.*nanoTime*());

getGameWorld();

**double** timeOutOfGameWorld = convertNanosToMillis(System.*nanoTime*());

**double** totalGameWorldTime = timeOutOfGameWorld - timeInGameWorld;

renderTime += totalGameWorldTime;

sleepIfNeeded(totalGameWorldTime);

displayTitleAndUpdateFrameRate();

**if**(bs == **null**) game.createBufferStrategy(2);

bs = game.getBufferStrategy();

}

**else**

{

coolDown = **false**;

**try** {Thread.*sleep*(5000);}

**catch** (InterruptedException e){e.printStackTrace();}

}

}

**private** **double** convertNanosToMillis(**long** nanoTime){**return** ((**double**) nanoTime / 1000000.0);}

**private** **void** getGameWorld()

{

**if**(gameKeeper[0] < 60)

{

update();

gameKeeper[0]++;

}

**boolean** render = (renderFrame || gameKeeper[0] == 60);

**if**(render)

{

render(bs);

gameKeeper[1]++;

}

**else**

{

skipFrame = **true**;

}

}

**private** **void** displayTitleAndUpdateFrameRate()

{

**if**(renderTime > oneSecond)

{

game.getFrame().setTitle(title + " | ups: " + gameKeeper[0] + "fps: " + gameKeeper[1]);

gameKeeper[0] = 0;

gameKeeper[1] = 0;

oneSecond += 1000;

**if**(oneSecond >= 300000.0)

{

coolDown = **true**;

renderTime = 0;

oneSecond = 1000;

game.getFrame().setTitle(title + " Game has overheated");

}

}

}

**private** **void** sleepIfNeeded(**double** gameWorldTime)

{

renderFrame = (gameWorldTime < previousGameWorldTime || gameWorldTime == previousGameWorldTime);

**if**(renderFrame && gameKeeper[0] < 60 && !skipFrame)

{

**double** sleepTime = (previousGameWorldTime - gameWorldTime);

**int** milliSleepTime = (**int**) sleepTime;

**int** nanoSleepTime = 0;

**if**(milliSleepTime > 0){nanoSleepTime = (**int**) ((sleepTime - milliSleepTime) \* 1000000);}

**else**{nanoSleepTime = (**int**) (sleepTime \* 1000000);}

**if**(sleepTime > 0)

{

**try** {Thread.*sleep*(milliSleepTime, nanoSleepTime);}

**catch** (InterruptedException e){e.printStackTrace();}

}

}

previousGameWorldTime = gameWorldTime;

skipFrame = **false**;

}

**private** **void** update()

{

gsm.update();

}

**private** **void** render(BufferStrategy bs)

{

**if**(bs == **null**) **return**;

Graphics g = bs.getDrawGraphics();

gsm.render(g);

g.dispose();

bs.show();

}

}