/\*\*

The Game Play Engine has an active MVC (model, view, controller) architecture.

The purpose of this file is to create the model, view and controller components and their interfaces.

\*/

package gamePlayEngine;

import gamePlayEngine.controller.GamePlayEngineController;

import gamePlayEngine.controller.MessageType;

import gamePlayEngine.model.game.GameBoundary;

import gamePlayEngine.view.GameView;

/\*\*

\* Core GamePlayEngine class - Creates Model, View and Controller and sets up interaction amongst them.

\*/

public class GamePlayEngine {

/\*\* Main entry point of the GamePlayEngine - Sets up the game and starts it up.

\* @param args

\* @throws Exception

\*/

public static void main(String[] args) throws Exception {

// Create the model

// calls need to be very consistent wrt naming convention – make it easier to understand

gameXxx vs. gamePlayEngineXxx

Controller, Model, View – not used consistently

Boundary not used consistently

GameBoundary gameBoundary = new GameBoundary(); Model??

gameBoundary.load("GameXML\\TestGame1.xml");

pass in as a parameter in main - remove hardcoding in the call here

// Create the View

GameView view = new GameView();

// Register the view for observing any updates to the model

gameBoundary.addObserver(view);

// Create the Controller

GamePlayEngineController gamePlayEngineController = new GamePlayEngineController(

gameBoundary);

// Tell the view the controller that will respond to user input.

view.setController(gamePlayEngineController);

// play the game -- start the game -- need to be able to see: start, play, end in a very obvious way in the code - rename?

gamePlayEngineController.play(MessageType.Internal, null);

}

}