

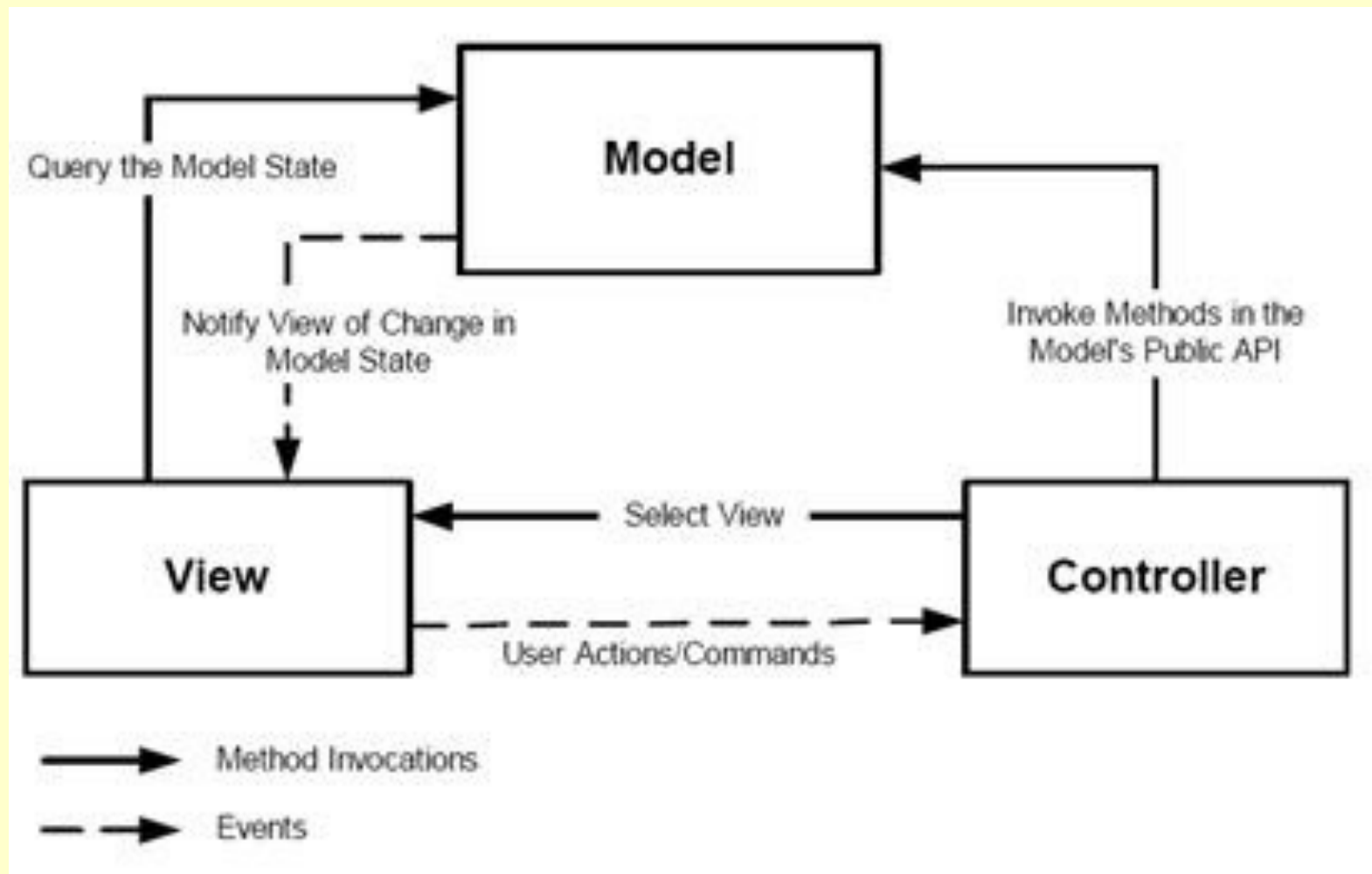
CSC4522 : Ingenierie du developpement logiciel - Patrons de conception et outils

J Paul Gibson, D311

MVC Design Pattern

<.../~gibson/Teaching/CSC4522/CSC4522-DesignPatterns-MVC.pdf>

MVC Design Pattern



<https://fentyoktorina.files.wordpress.com/2011/05/mvc.jpg>

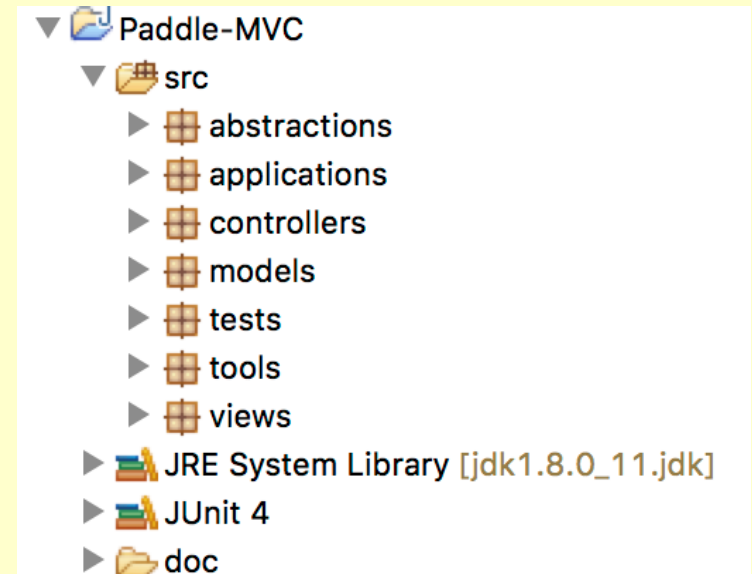
The Model View Design Pattern – PBL Session

Download the code from the module web site:

...~gibson/Teaching/CSC4522/SourceCode/Paddle-MVC.zip

Note the package structure – particularly the MVC components:

- Models
- Views
- Controllers



Let us examine some of this code together

The Model View Design Pattern – PBL Session

PaddleSpecification

I p_abstractions.PaddleSpecification

Specification of simple paddle behaviour for use in a video game.
For teaching MVC design pattern.

- The paddle has one degree of freedom - moving either left or right.
- It has a position which is an integer value bounded by minimum and maximum values

Version:

1.0.0

Author:

J Paul Gibson

The Model View Design Pattern – PBL Session

PaddleSpecification

```
/**
 * The lower bound on the horizontal position of the paddle
 */
final int MINIMUM_position = 0;

/**
 * The upper bound on the horizontal position of the paddle
 */
final int MAXIMUM_position = 31;

/**
 * @return true if the MINIMUM_position value is
 * less than the MAXIMUM_position
 */
boolean INVARIANT_OF_CLASS =
    (MINIMUM_position <= MAXIMUM_position);
```

The Model View Design Pattern – PBL Session

PaddleSpecification

- **int p_abstractions.PaddleSpecification.get_position()**

Returns:

The position of the paddle - must be within the defined limit: MINIMUM_position ... MAXIMUM_position

- **boolean p_abstractions.PaddleSpecification.goingRight()**

Returns:

true if the paddle is moving to the right and false otherwise

The Model View Design Pattern – PBL Session

PaddleSpecification

- **void p_abstractions.PaddleSpecification.updatePosition()**

Update paddle position or direction of movement:

- if moving out of defined limits then change the direction of the paddle movement without changing position
- if moving right inside limits then increment position
- if moving left inside limits then decrement position

The Model View Design Pattern – PBL Session

PaddleSpecification

- **void p_abstractions.PaddleSpecification.changeDirection()**

Changes direction from left to right, or right to left.

- **boolean p_abstractions.PaddleSpecification.equals(Object thing)**

Overrides: [equals\(...\)](#) in [Object](#)

Parameters:

thing is the input object to test for equality

@returns

true if the input parameter is equal to the Paddle object, where 2 paddles are considered equal if they have the same position and the same direction

The Model View Design Pattern – PBL Session

PaddleSpecification

- **String p_abstractions.PaddleSpecification.toString()**

Overrides: [toString\(\)](#) in [Object](#)

@returns

the string representing the state of the Paddle.


The string format follows the template below (illustrated using default constructor values):

```
Paddle: position = 0, moving = right, is in safe state.
```

For an unsafe Paddle, the format simply adds a not to the string, eg:

```
Paddle: position = 100, moving = right, is not in safe state.
```

Paddle Implementation: Paddle which bounces

 **models.PaddleBounce**

Implements [HasInvariant](#), [PaddleSpecification](#)

A simple bouncing paddle model for use in a video game.
For teaching MVC design pattern.

Version:

1.1.0

Author:

J Paul Gibson

The Model View Design Pattern – PBL Session

Paddle Implementation: Paddle

QUESTION: Do the methods that change the state respect the invariant?

```
public void updatePosition() {  
  
    if (directionToRight && position < MAXIMUM_position) position+  
+;  
    else if (!directionToRight && position > MINIMUM_position)  
position--;  
    else changeDirection();  
}  
  
public void changeDirection() {  
directionToRight= ! directionToRight;  
}
```

The Model View Design Pattern – PBL Session

Paddle Implementation: Testing the Paddle Model

p_tests.RandomTest_Paddle

Test class for [Paddle](#) that uses a [Random](#) RNG for simulation purposes.
The RNG can be seeded at the command line, or a default value of 0 can be used.

We use the [DateHeader](#) class to document the date/time of the test execution

Expected Output (using default RNG seed = 0) and `NUMBER_OF_TESTS = 6`:

```
The seed used for the random number generator in the test is 0.  
You can override this value by passing an integer value as a main argument parameter, if you so wish.
```

```
*****  
Execution Date/Time 2011/03/16 11:29:28  
*****  
Creating a random Paddle 6 times:  
  
Paddle: position = 6, moving = right, is in safe state.  
Paddle: position = 7, moving = right, is in safe state.  
Paddle: position = 10, moving = left, is in safe state.  
Paddle: position = 5, moving = left, is in safe state.  
Paddle: position = 6, moving = right, is in safe state.  
Paddle: position = 18, moving = left, is in safe state.
```

Version:

1

Author:

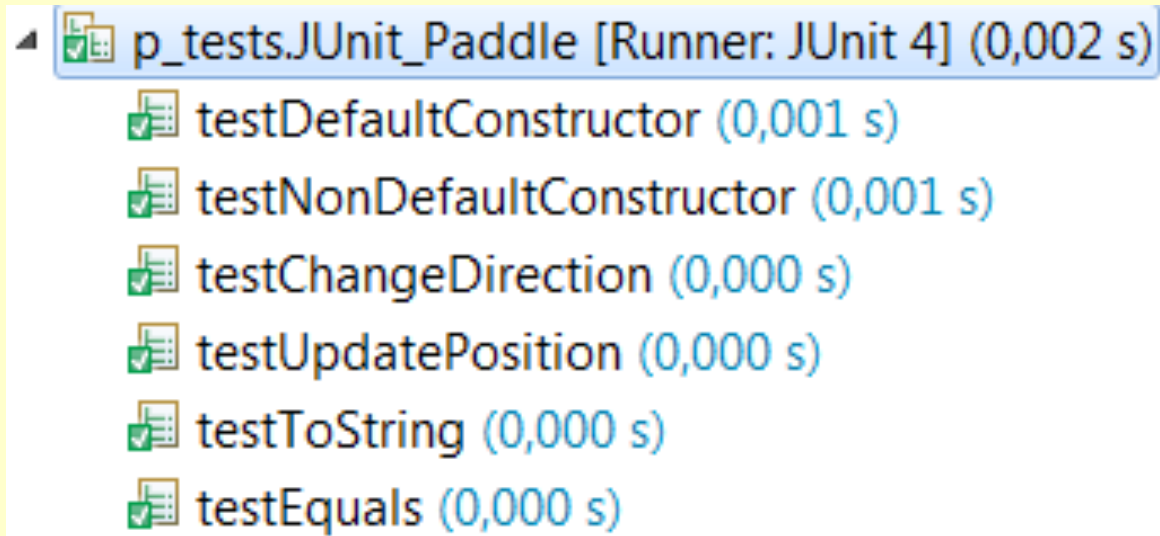
J Paul Gibson

See Also:

[JUnit_Paddle](#)

The Model View Design Pattern – PBL Session

Unit Testing the Paddle Model



TO DO: Check that you understand the unit test code

The Model View Design Pattern – PBL Session

Paddle View Specification

Once we have tested our model (the Paddle) we can develop a (graphical) view:

I **abstractions.PaddleViewSpecification**

Specification of a simple paddle view for use in a video game:

- View is 360*360 square

For teaching MVC design pattern.

Version:

1.0.0

Author:

J Paul Gibson

The Model View Design Pattern – PBL Session

Paddle View Specification

Field Summary

static int	<u>BORDER</u>
static int	<u>VIEW_HEIGHT</u>
static int	<u>VIEW_WIDTH</u>


Method Summary

javax.swing.JFrame	<u>getFrame</u> ()
void	<u>updateView</u> () update the canvas on which the view is painted

QUESTION: Why do we specify these methods?

The Model View Design Pattern – PBL Session

Paddle View – the implementation (as a plain paddle)

 **views.PaddleViewPlain**

Implements [PaddleViewSpecification](#)

The view is a simple rectangle For teaching MVC design pattern.

Version:


1.1.0

Author:

J Paul Gibson

The Model View Design Pattern – PBL Session

Extending the model so that it can be animated inside the view

 **models.RunnableViewablePaddle**

Implements [Runnable](#)

For teaching MVC design pattern.

Version:

1.1.0

Author:

J Paul Gibson

The Model View Design Pattern – PBL Session

Extending the model so that it can be animated inside the view

Method Detail

setView

```
public void setView(p_abstractions.PaddleViewSpecification paddleView2)
```

Parameters:

paddleView2 - the current view which responds to state changes

run

```
public void run()
```

Every 10th of second:

- update the paddle position
- inform the view (if it has been initialised) of the update

Specified by:

run in interface `java.lang.Runnable`

The Model View Design Pattern – PBL Session


Extending the model so that it can be animated inside the view

```
public void run() {  
  
    do{  
        try {  
            Thread.sleep(DELAY);  
        } catch (InterruptedException e) {e.printStackTrace();}  
  
        updatePosition();  
        if (paddleView !=null) paddleView.updateView();  
        }  
        while (true);  
    }
```

QUESTION: do you understand how the delay is implemented?

The Model View Design Pattern – PBL Session

Adding a controller to the system – the specification

 **p_abstractions.PaddleControllerAbstraction**

Implements [KeyListener](#)

Specification of a simple paddle controller for use in a video game.

Typing a character on the keyboard changes direction of movement of the Paddle.

For teaching MVC design pattern.

Method Summary

void	keyPressed (java.awt.event.KeyEvent e) Should not react to key presses
void	keyReleased (java.awt.event.KeyEvent e) Should not react to key releases
abstract void	keyTyped (java.awt.event.KeyEvent e) Should react to key typing (press and release)

The Model View Design Pattern – PBL Session

Adding a controller to the system – the implementation

```
public class PaddleController extends PaddleControllerAbstraction{

    /**
     * The model being controlled by the controller
     */
    PaddleSpecification paddle;

    /**
     * @param rvPaddle is the model to be controlled by the controller
     */
    public PaddleController(PaddleSpecification rvPaddle){

        this.paddle = rvPaddle;
    }

    /**
     * Change direction when a key is typed
     */
    public void keyTyped(KeyEvent e){ paddle.changeDirection();}
}
```

The Model View Design Pattern – PBL Session

Adding a controller to the system – the MVC structure

p_models.PaddleMVC

A first step in building a java game where a paddle constantly moves horizontally at the bottom of a 2-D screen and its direction is changed/controlled by keyboard presses

For teaching MVC design pattern, and introducing Java threads:

- Model is [RunnableViewablePaddle](#)
- View is [PaddleView](#)
- Controller is [PaddleController](#)

NOTE: This is not intended as a good example of UI development in Java, it is intended only as a good introduction to the MVC design pattern

Version:

1.0.0

Author:

J Paul Gibson

The Model View Design Pattern – PBL Session

Adding a controller to the system – the MVC structure

```
public PaddleBounceViewSimple(){  
  
    // Construct model  
    rvPaddle = new RunnableViewablePaddle(new PaddleBounce());  
  
    // Construct view which can see model  
    paddleView = new PaddleViewPlain(rvPaddle);  
  
    //Allow the model to see view in order to make updates when state changes  
    rvPaddle.setView(paddleView);  
  
    //Construct controller  
    PaddleController paddleController = new PaddleController(rvPaddle);  
  
    //The frame which contains the view must allow the controller to react to key presses  
    paddleView.getFrame().addKeyListener(paddleController);  
}
```

The Model View Design Pattern – PBL Session

Adding a controller to the system – the MVC structure

Now we just need a method that starts a thread containing the runnable viewable paddle

```
public void startgame() {  
  
    Thread paddleThread = new Thread((Runnable) rvPaddle);  
    paddleThread.run();  
}
```


The Model View Design Pattern – PBL Session

Adding a controller to the system – the MVC system application

p_applications.PaddleMVC_Application

Instantiates [PaddleMVC](#) and starts its execution
For teaching MVC design pattern.

Version:

1.0.0

Author:

J Paul Gibson

```
/**
 * Instantiates {@link PaddleBounceViewSimple} and starts its execution <br>
 * For teaching MVC design pattern.
 *
 * @version 1.0.0
 * @author J Paul Gibson
 */
public class PaddleMVC_Application {

    public static void main(String[] args){

        PaddleBounceViewSimple application = new PaddleBounceViewSimple();
        application.startgame();

    }
}
```

The Model View Design Pattern – PBL Session

The MVC problem

The use of the MVC design pattern should make it easier to maintain/extend/update the Paddle application.

TO DO:

1. Change the model so that the paddle doesn't bounce it wraps around
2. Change the controller so that you have to hold down a button to move left and hold down a button to move right
3. Change the view so that the direction of the next move is represented graphically

QUESTION: How many different Paddle systems are now possible (based on 2 different models, two different views and 2 different controllers)?

Can you instantiate all of them and test their behaviour?

The Model View Design Pattern – PBL Session

The MVC problem

The use of the MVC design pattern should make it easier to maintain/extend/update the Paddle application.

Lessons?

**The model and view need to be properly separated/
independent/decoupled**

**The model can be connected to only a single view – we need the
observer design pattern to connect it to multiple views**

**The controller code requires better understanding of threads
and event handling....**