

CSC 7322 : Object Oriented Development

J Paul Gibson, A207

`paul.gibson@int-edu.eu`

<http://www-public.it-sudparis.eu/~gibson/Teaching/CSC7322/>

Game Paddle – MVC Design Pattern

[.../~gibson/Teaching/CSC7322/PaddleForGameMVC.pdf](http://www-public.it-sudparis.eu/~gibson/Teaching/CSC7322/PaddleForGameMVC.pdf)

The Model View Design Pattern – PBL Session

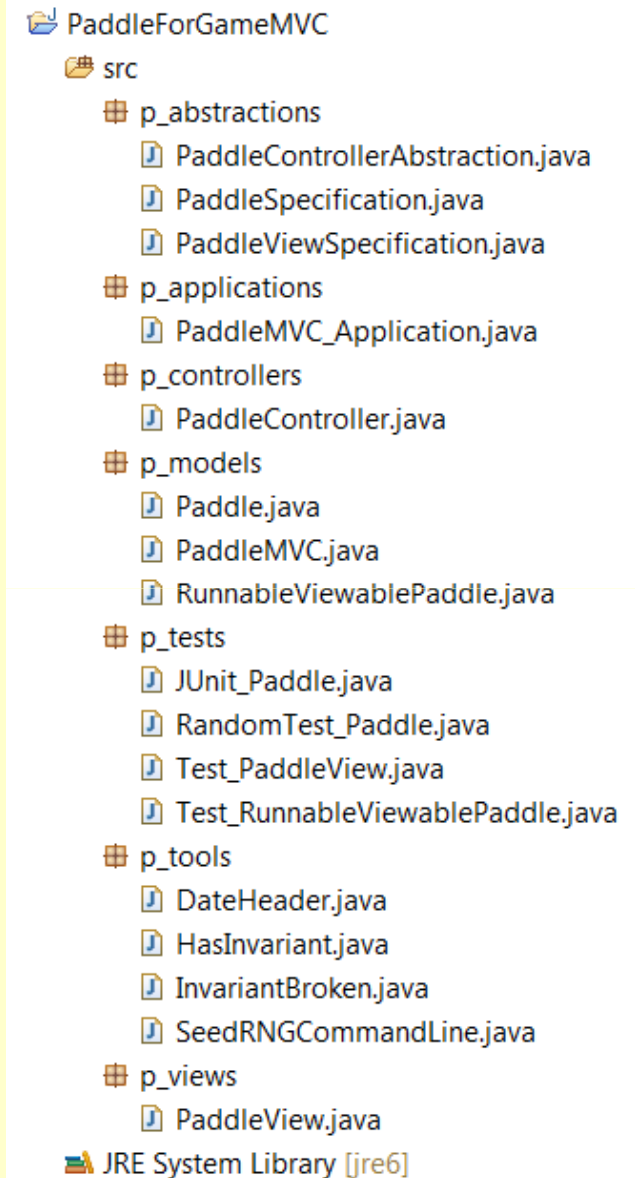
Download the code from the module web site:

...[~gibson/Teaching/CSC7322/Code/PaddleForGameMVC.zip](http://www.gibson/Teaching/CSC7322/Code/PaddleForGameMVC.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

Field Summary

static boolean	INVARIANT_OF_CLASS
static int	MAXIMUM_position The upper bound on the horizontal position of the paddle
static int	MINIMUM_position The lower bound on the horizontal position of the paddle

Method Summary

void	changeDirection() Changes direction from left to right, or right to left.
boolean	equals (java.lang.Object thing)
int	get_position()
boolean	goingRight()
java.lang.String	toString()
void	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

```
/**
 * 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 <code> MINIMUM_position </code> value is
 * less than the <code> MAXIMUM_position</code>
 */
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.
```

The Model View Design Pattern – PBL Session

Paddle Implementation: Paddle

 **p_models.Paddle**

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

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

Version:

1.0.0

Author:

J Paul Gibson

The Model View Design Pattern – PBL Session

Paddle Implementation: Paddle

Constructor Summary

[Paddle\(\)](#)

Tested by `JUnit_Paddle.testDefaultConstructor()`, which guarantees that the Paddle is constructed in a safe state as specified by [invariant\(\)](#).

[Paddle\(int pos\)](#)

Tested by `JUnit_Paddle.testNonDefaultConstructor()`, which guarantees that the Paddle is constructed in a safe state as specified by [invariant\(\)](#).

[Paddle\(java.util.Random rng\)](#)

Tested by [RandomTest_Paddle](#), which guarantees that the Paddle is constructed in a safe state as specified by [invariant\(\)](#).

TO DO: Examine the code and check that you understand it.

The Model View Design Pattern – PBL Session

Paddle Implementation: Paddle

Method Summary

void	changeDirection() Tested by <code>p_tests.JUnit_Paddle#testChangeDirection()</code> , which guarantees that the Paddle remains in a safe state as specified by invariant() .
boolean	equals (java.lang.Object thing)
int	get_position ()
boolean	goingRight ()
boolean	invariant () The position must be within the defined limit: <code>PaddleSpecification.MINIMUM_position ...</code>
java.lang.String	toString ()
void	updatePosition () Tested by <code>p_tests.JUnit_Paddle#testUpdatePosition()</code> , which guarantees that the Paddle remains in a safe state as specified by invariant() .

TO DO: Examine the code and check that you understand it.

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:

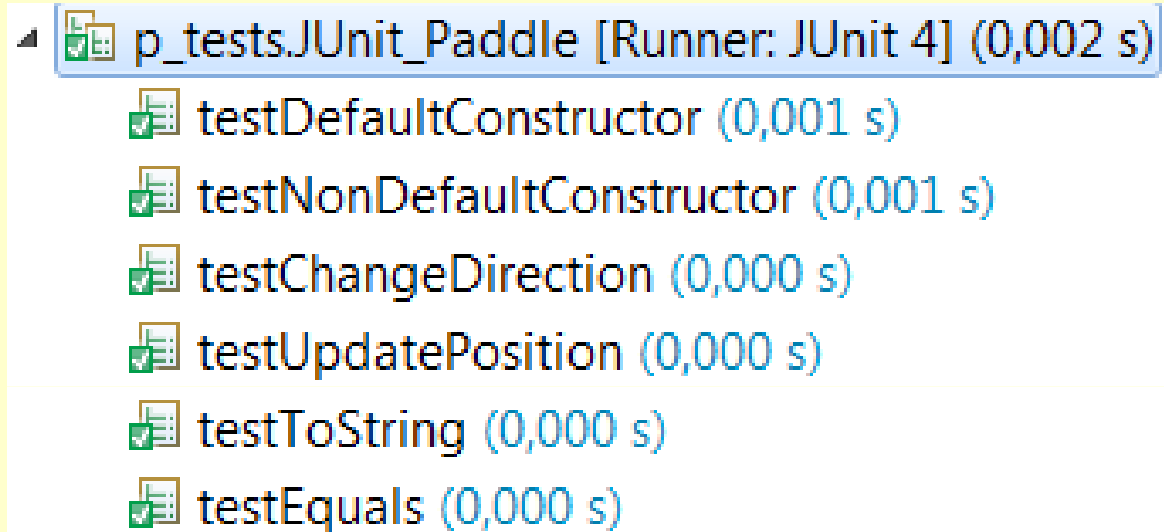
J Paul Gibson

See Also:

[JUnit Paddle](#)

The Model View Design Pattern – PBL Session

Unit Testing the Paddle Model



```
▲ p_tests.JUnit_Paddle [Runner: JUnit 4] (0,002 s)
  ✓ testDefaultConstructor (0,001 s)
  ✓ testNonDefaultConstructor (0,001 s)
  ✓ testChangeDirection (0,000 s)
  ✓ testUpdatePosition (0,000 s)
  ✓ testToString (0,000 s)
  ✓ testEquals (0,000 s)
```

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 `p_abstractions.PaddleViewSpecification`

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

- View is 640*640 square with a 20 pixel border around it

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	BORDER
static int	VIEW_HEIGHT
static int	VIEW_WIDTH

Method Summary	
javax.swing.JFrame	getFrame ()
void	updateView () 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 Test

 p_tests.Test_PaddleView

Tests [PaddleView](#)

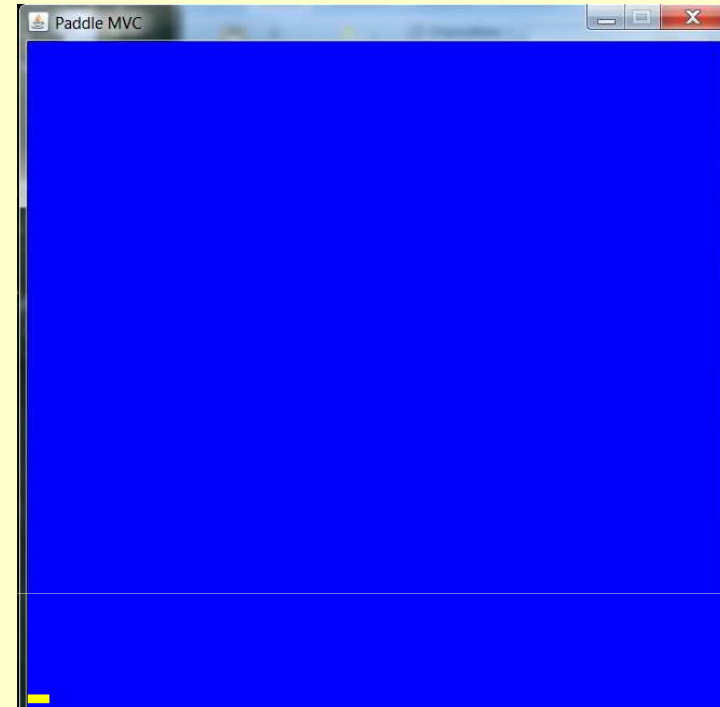
Should print out paddle in default position at bottom left of screen.

Version:

1.0.0

Author:

J Paul Gibson



```
public class Test_PaddleView {  
  
    public static void main(String args[]) {  
  
        Paddle paddleModel = new Paddle(0);  
        PaddleView paddleView = new PaddleView(paddleModel);  
        paddleView.updateView();  
    }  
  
}
```

The Model View Design Pattern – PBL Session

Paddle View – the implementation

 **p_views.PaddleView**

Implements [PaddleViewSpecification](#)

For teaching MVC design pattern.

Version:

1.0.0

Author:

J Paul Gibson

The Model View Design Pattern – PBL Session

Paddle View – the implementation

Constructor Detail

PaddleView

```
public PaddleView(PaddleSpecification rvPaddle)
```

Parameters:


rvPaddle - is the [Paddle](#) model which is to be associated with the view

```
public PaddleView(PaddleSpecification rvPaddle){  
  
    frame = new JFrame("Paddle MVC");  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    frame.setSize(VIEW_WIDTH+(BORDER/2),VIEW_HEIGHT+(BORDER/2));  
    frame.setVisible(true);  
    frame.setResizable(false);  
    canvas = new PaddleViewCanvas(rvPaddle, VIEW_WIDTH, VIEW_HEIGHT);  
  
    frame.getContentPane().add(canvas, BorderLayout.CENTER );  
}
```

TO DO:
Check your
understanding
of the code

The Model View Design Pattern – PBL Session

Paddle View Canvas – where the drawing is done inside the View frame

 **p_views.PaddleViewCanvas**

Local canvas class for inside of [PaddleView](#) frame responsible for graphical representation of the [Paddle](#) model

Version:

1.0.0

Author:

J Paul Gibson

TO DO:

Check your understanding of the code

The Model View Design Pattern – PBL Session

Paddle View – the implementation

Method Detail

getFrame

```
public javax.swing.JFrame getFrame()
```

Specified by:

[getFrame](#) in interface [PaddleViewSpecification](#)

Returns:

the frame which contains the canvas in which the view is to be painted and which generates the events that need to be handled by the controller

updateView

```
public void updateView()
```

Description copied from interface: [PaddleViewSpecification](#)


update the canvas on which the view is painted

Specified by:

[updateView](#) in interface [PaddleViewSpecification](#)

The Model View Design Pattern – PBL Session

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

 **p_models.RunnableViewablePaddle**

Implements [Runnable](#)

Extends [Paddle](#) with link back to the view and a run method

For teaching MVC design pattern.

Version:

1.0.0

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

Testing the animated view

 `p_tests.Test_RunableViewablePaddle`

Tests [RunableViewablePaddle](#)

Should initially display the paddle in the default initial position at bottom left of screen.

Then the paddle should move from left to right (then right to left) as it traverses the screen horizontally

```
public class Test_RunableViewablePaddle {

    public static void main(String args[]) {

        RunableViewablePaddle rvPaddle = new RunableViewablePaddle();
        PaddleView paddleView = new PaddleView(rvPaddle);
        rvPaddle.setView(paddleView);
        Thread paddleThread = new Thread(rvPaddle);
        paddleThread.run();
    }
}
```

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 PaddleMVC(){  
  
    // Construct model  
    rvPaddle = new RunnableViewablePaddle();  
  
    // Construct view which can see model  
    paddleView = new PaddleView(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:

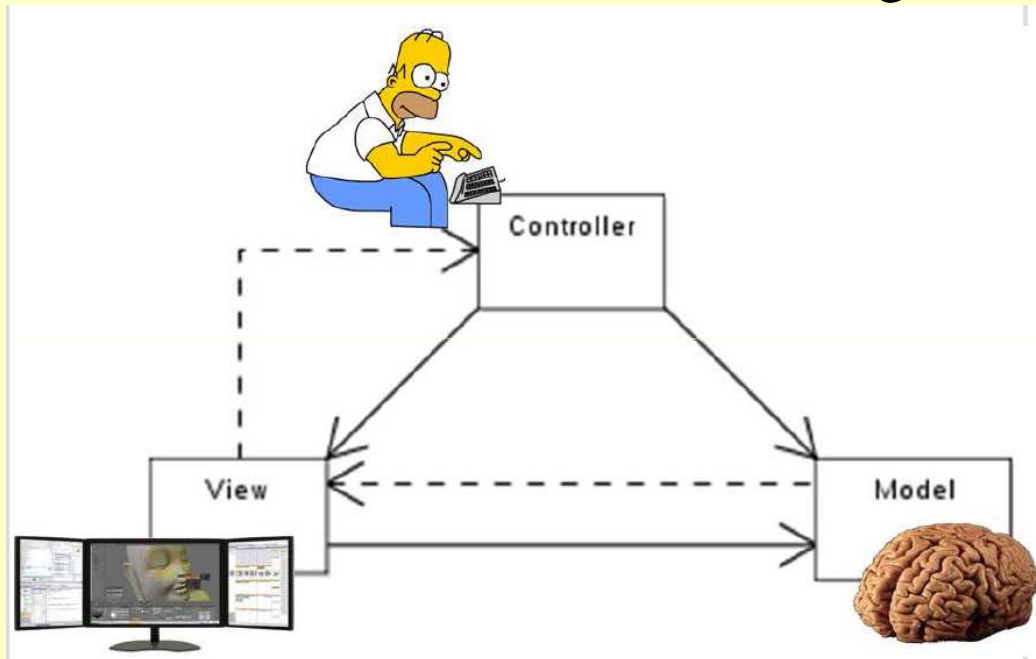
J Paul Gibson

```
public class PaddleMVC_Application {  
  
    public static void main(String[] args){  
  
        PaddleMVC application = new PaddleMVC();  
        application.startgame();  
    }  
  
}
```

The Model View Design Pattern – PBL Session

The MVC design pattern

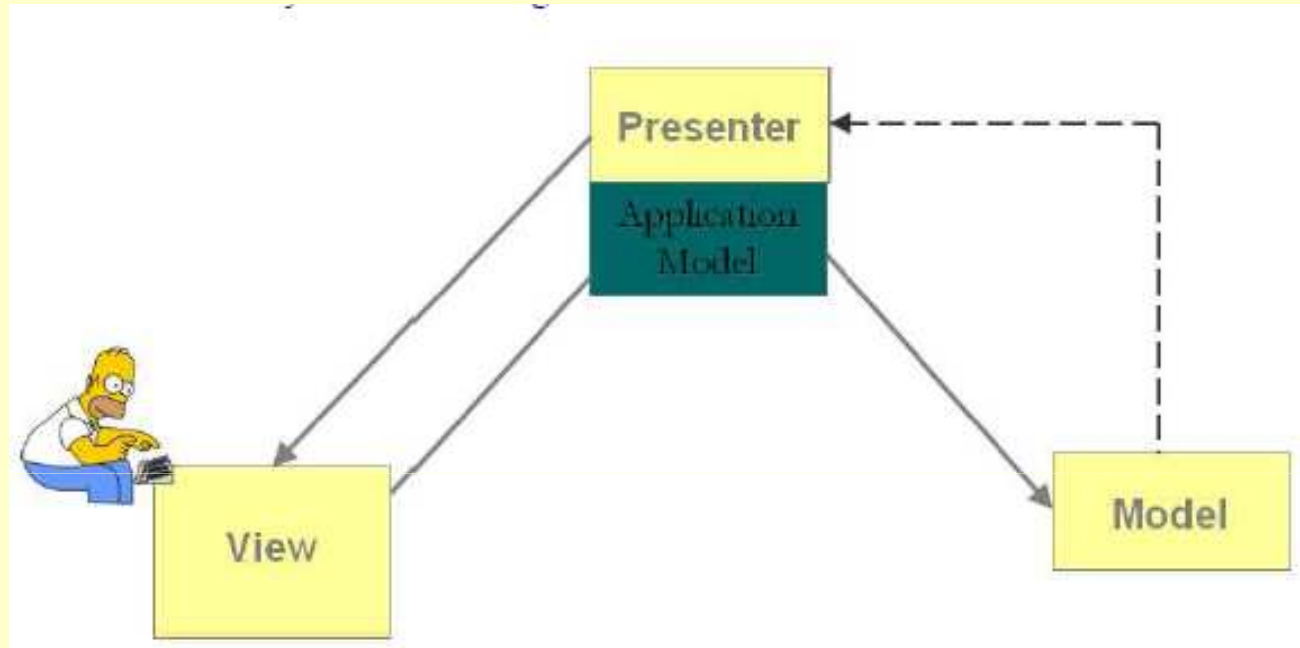
Helps to bridge the gap between the human user's mental model and the digital model of the computer



See <http://geekswithblogs.net/gregorymalcolm/archive/2009/07/14/user-interface-patterns.aspx>

The Model View Design Pattern – PBL Session

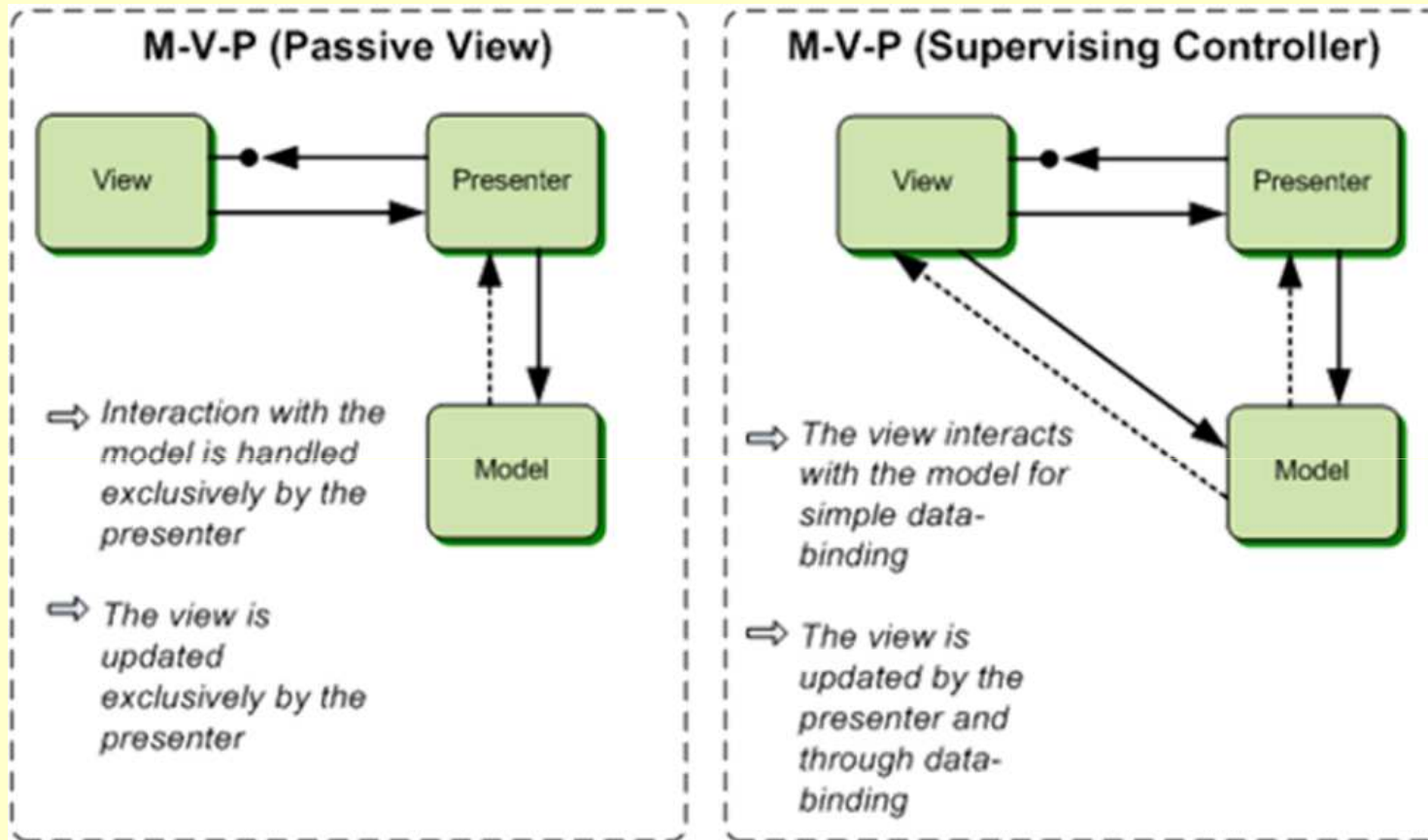
The MVC variations - Model View Presenter (MVP)



See <http://geekswithblogs.net/gregorymalcolm/archive/2009/07/14/user-interface-patterns.aspx>

The Model View Design Pattern – PBL Session

The MVC variations - Model View Presenter (MVP)



See <http://geekswithblogs.net/gregorymalcolm/archive/2009/07/14/user-interface-patterns.aspx>

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?