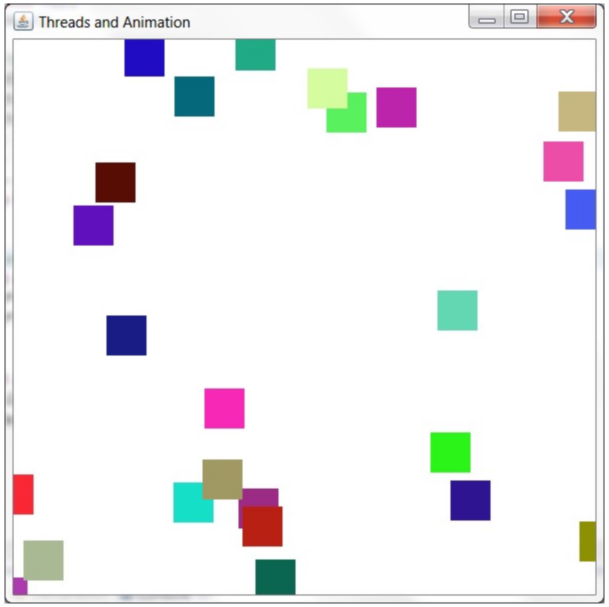
CT255 / NGT II
Digital Media / 2D Games Dev.

Week 3

sam.redfern@universityofgalway.ie
@psychicsoftware

Last Week's Assignment

- Create a program which performs simple random animation of coloured squares
- Use two classes:
  1. MovingSquaresApplication
     - extends JFrame
     - Implements Runnable
     - has main() method
     - Member data includes an array of GameObject instances
     - Constructor method does similar setup as last week's code, plus instantiates the GameObjects in the array, and creates+starts a Thread
     - Uses a Thread to perform animation of the GameObjects by calling their move() methods
     - Paint() method draws the GameObjects by calling their paint(Graphics g) methods
  2. GameObject
     - Member data includes x,y,color
     - Constructor method randomises the object's position and color
     - Public move() method is used to randomly alter x,y members
     - Public paint(Graphics g) method draws the object as a square using g.fillRect()



Topics this week

- Handling the keyboard in Java
- Loading and displaying raster images (.jpg, .png etc.)
- Moving a player’s game object under control of the keyboard

Handling Keyboard Input

- In GUI-based languages such as Java (with AWT) the mouse and keyboard are handled as ‘Events’
- They may happen at any time
- They are queued as they happen and are dealt with at the next free idle time
- AWT handles events coming in from the operating system by dispatching them to any *listeners* registered to those events

Handling Keyboard Input

- Make a class that implements KeyListener
- Make sure you have an instance of this class
- Add this instance as a key listener attached to the JFrame that receives the messages from the Operating System
- The simplest way is to make your JFrame-derived class itself handle the events it receives.. (see next slide)

Handling Keyboard Input

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyApplication extends JFrame implements KeyListener  {

   public MyApplication() {  // constructor
    // send keyboard events arriving into this JFrame to its own event handlers
       addKeyListener(this);
   }

   // Three Keyboard Event-Handler functions
    public void keyPressed(KeyEvent e) {
    }

    public void keyReleased(KeyEvent e) {
    }

    public void keyTyped(KeyEvent e) {
    }
    //
}
```

Notes:

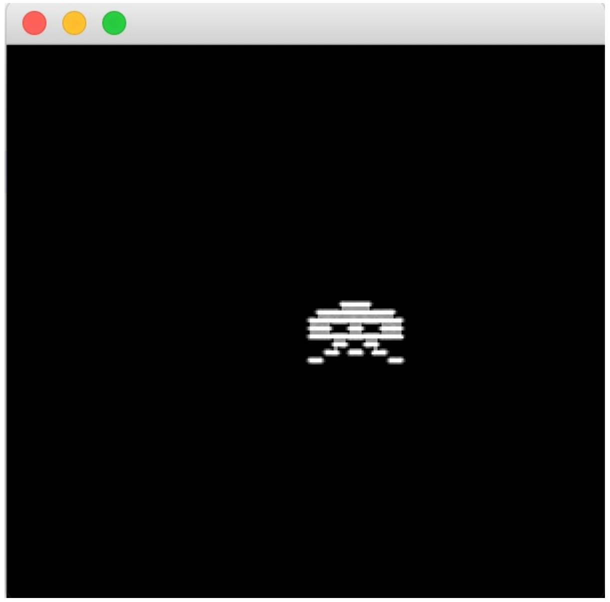
- The KeyEvent parameter 'e' provides the 'virtual keycode' of the key that has triggered the event, and constants are defined to match these values: e.g. KeyEvent.VK_Q or KeyEvent.VK_ENTER
- To get the keycode, use e.getKeyCode()
- For our game applications, our application class will implement both KeyListener and Runnable
- Note the extra import!! - java.awt.event.*

Loading and displaying raster images

- The constructor of the **ImageIcon** class (defined in javax.swing) loads an image from disk (.jpg, .gif, or .png) and returns it as a new instance of the **ImageIcon** class.
- The **getImage()** method of this **ImageIcon** object gives you a useable **Image** class object, which can be displayed in your **paint()** method by the **Graphics** class

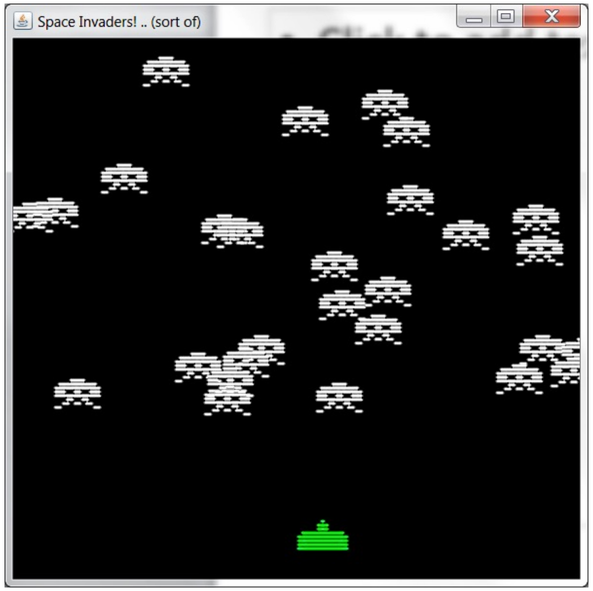
Example

```java
import java.awt.*;
import javax.swing.*;
public class DisplayRasterImage extends JFrame {

// member data
private static String workingDirectory;
private Image alienImage;

// constructor
public DisplayRasterImage() {
       // set up JFrame
       setBounds(100, 100, 300, 300);
       setVisible(true);

        // load image from disk. Make sure you have the path right!
        // NB Windows uses \\ in paths whereas MacOS uses / in paths
        ImageIcon icon = new ImageIcon(workingDirectory + "\\alien_ship_1.png");
        alienImage = icon.getImage();

        repaint();
}

// application's paint method (may first happen *before* image is finished loading, hence repaint() above)
public void paint(Graphics g) {
       // draw a black rectangle on the whole canvas
       g.setColor(Color.BLACK);
       g.fillRect(0, 0, 300, 300);
       // display the image (final argument is an ‘ImageObserver’ object)
       g.drawImage(alienImage, 150, 150, null);
}

// application entry point
public static void main(String[] args) {
       workingDirectory = System.getProperty("user.dir");
       System.out.println("Working Directory = " + workingDirectory);
       DisplayRasterImage d = new DisplayRasterImage();
}

}
```



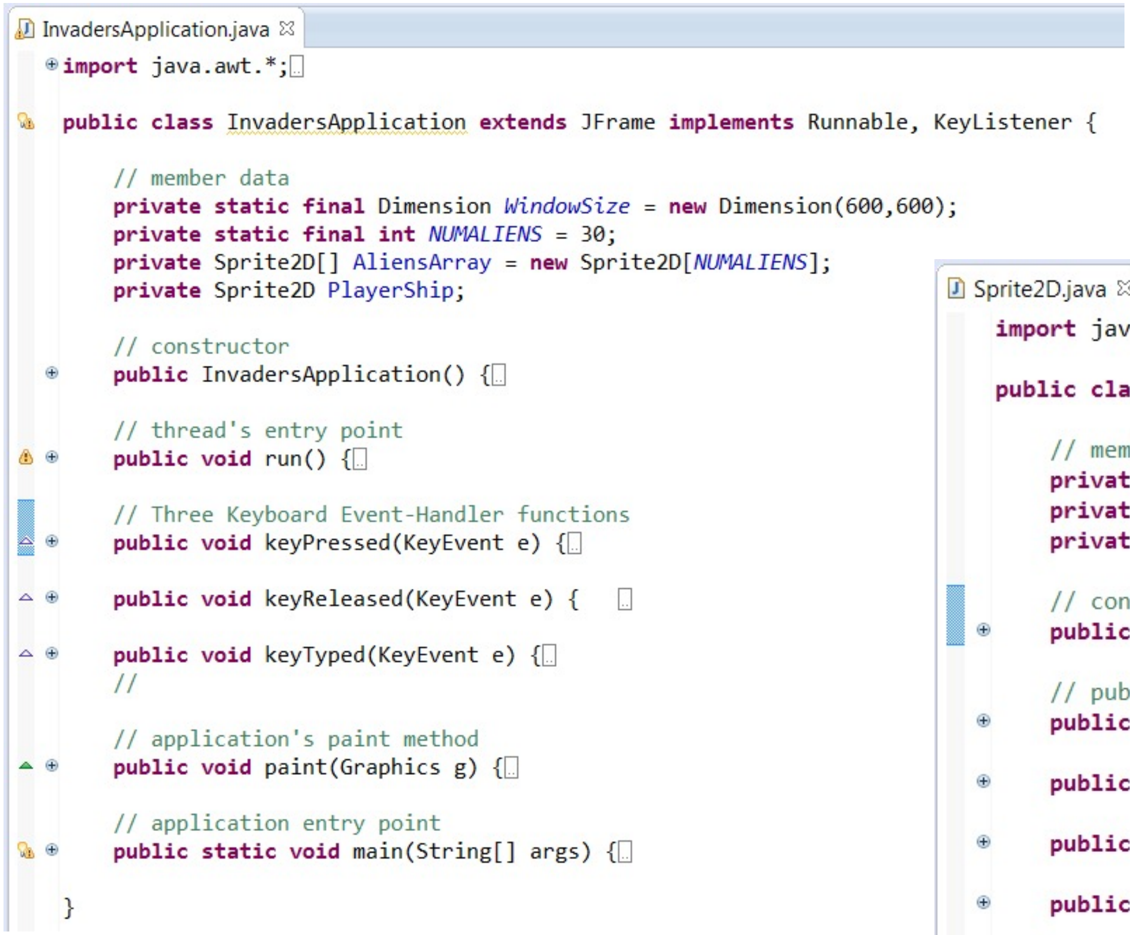
Week 3 exercise

- Create a JFrame-based, Runnable KeyListener application class and a separate class for handling game objects
- Use these names for your classes:
  - InvadersApplication
  - Sprite2D
- The InvadersApplication class should have, as its member data, an array of Sprite2D objects for aliens, and another single Sprite2D object for the player ship
- The InvadersApplication class should use Thread-based animation to move the aliens randomly (similar to last week)
- The Sprite2D objects display a raster image that you have loaded from disk (instead of a coloured square)
  - See **ct255-images.zip** for png files to use
- Use the left and right arrow keys to move the player spaceship, rather than moving it randomly like the aliens
  - Do NOT move the spaceship directly in the keyboard event handlers, since that will mean it will move in steps based on your keyboard repeat rate
  - The correct way to do it is to have the keyboard events notify the spaceship when movement should start and stop; the actual movement should be done every frame (i.e. 50 times per second) by the movePlayer() method suggested on the next slide

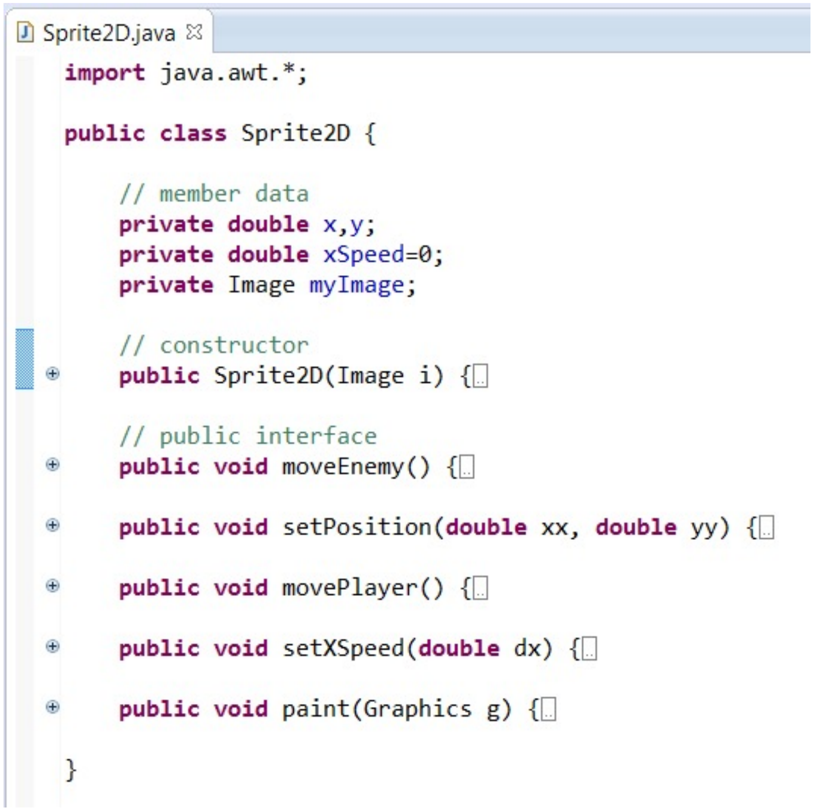


- ***Code should be submitted on Blackboard.***
- ***Deadline: before next lecture.***

Assignment #3
Suggested Class Interfaces



```java
import java.awt.*;

public class InvadersApplication extends JFrame implements Runnable, KeyListener {

    // member data
    private static final Dimension WindowSize = new Dimension(600,600);
    private static final int NUMALIENS = 30;
    private Sprite2D[] AliensArray = new Sprite2D[NUMALIENS];
    private Sprite2D PlayerShip;

    // constructor
    public InvadersApplication() {

    // thread's entry point
    public void run() {

    // Three Keyboard Event-Handler functions
    public void keyPressed(KeyEvent e) {

    public void keyReleased(KeyEvent e) {

    public void keyTyped(KeyEvent e) {
    //

    // application's paint method
    public void paint(Graphics g) {

    // application entry point
    public static void main(String[] args) {

}
```



```java
import java.awt.*;

public class Sprite2D {

    // member data
    private double x,y;
    private double xSpeed=0;
    private Image myImage;

    // constructor
    public Sprite2D(Image i) {

    // public interface
    public void moveEnemy() {

    public void setPosition(double xx, double yy) {

    public void movePlayer() {

    public void setXSpeed(double dx) {

    public void paint(Graphics g) {

}
```