


Classes

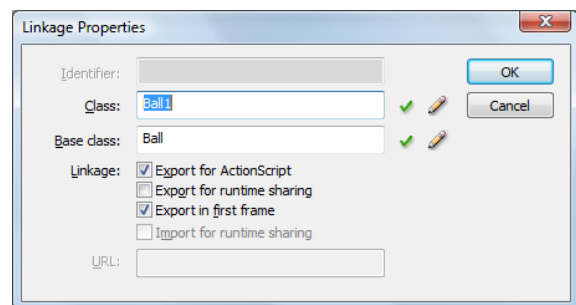
In an earlier lesson we created a ball that bounced around the stage. Instead of putting a ball on the stage and then writing code to make the ball move we will create a Ball class that moves around the stage. After creating the Ball class, we can add as many instances of the Ball as we want and they will all bounce. When we create a class we are using object oriented programming to make it easier to write large programs and to modify one thing without breaking something else.

- Start a new **Flash (ActionScript3)** file and save it as **bounces fla**.
- Draw a circle and make it a movie clip named Ball1 with the registration point in the center. 
- Start a new **ActionScript file** and save it in the same directory as **Ball.as** (Case as shown!) Note that the class name must be the same as the file name.
- Enter the code below for **Ball.as** as shown below:

```
package {
    import flash.display.MovieClip;
    import flash.events.*;
    public class Ball extends MovieClip {
        private var dx:int=3;
        private var dy:int=-2;
        function Ball() {
            addEventListener(Event.ENTER_FRAME,frames);
        }
        private function frames(e:Event):void {
            this.x+=dx;
            this.y+=dy;
            if(this.x<0 || this.x> MovieClip(root).stage.stageWidth) dx*=-1;
            if(this.y<0 || this.y> MovieClip(root).stage.stageHeight) dy*=-1;
        } //frames
    } //Ball
} //package
```

Make sure you save this file, then go back to bounces fla.

- In the library, right click (control+click on the Mac) Ball1 and select linkage from the pop-up menu.
- Select Export for ActionScript.
- The class is Ball1, make the base class **Ball** instead of flash.display.movieClip.



- Test the movie. The ball should bounce around the stage, if not correct any errors in Ball.as
- If it works, drag a few more instances of Ball1 onto the stage. They will all move!

When you add additional balls to the stage they all bounce in the same way because as instances of the Ball class, they have the movement built-in. You may have noticed that they do not bounce off the edge the way they should because we put the registration point in the center. Change the code in function frames (in **Ball.as**) as shown below:

```
if(this.x<this.width/2 || this.x> MovieClip(root).stage.stageWidth-this.width/2)
    dx*=-1;
if(this.y<this.height/2 || this.y> MovieClip(root).stage.stageHeight-this.height/2)
    dy*=-1;
```

Notice that when we are working in the ball class, this refers to the ball, not the movie. When we refer to the movie we must reference it using MovieClip(root). Each Ball has its own properties dx and dy, but they are the same for each ball. Let's make the movie a bit more interesting by adding a function to set the speed for a ball. Add the code below to Ball.as. This code can go after function frames but before the two closing } for the class and the package:

```
public function setSpeed(dx:int, dy:int):void {
    this.dx=dx;
    this.dy=dy;
} //setSpeed
```

Make sure you save, then go back to bounces fla. Name one of the instances of the ball **ball1** in the properties window. Make it a little bigger so that you can tell which one it is. Put the code below in the ActionScript for the movie (It is the only code in the movie.):

```
ball1.setSpeed(0,2);
```

Test the movie. The instance of the ball that we named goes up and down because its **dx** property has been set to 0.

Draw another ball on the stage in a different color and size. Double click to select it, then select Modify, Convert to symbol from the pop-up menu. In the Convert to symbol window, name it **Ball2**. You can select Export for ActionScript and put in the base class as **Ball** all at once instead of opening the linkage window later.

The new ball moves around just like the others with no extra work.

Experiment: Name each of the balls on the stage and give them each a different speed.

In the ball class, add another event listener: `addEventListener(MouseEvent.CLICK,clicked);`

Write the function **clicked** that will add one to both dx and dy when the ball is clicked.