Game of Pong

Starting development

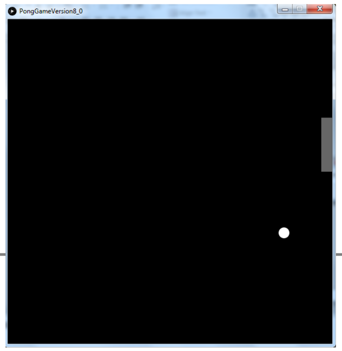
Produced by:

Dr. Siobhán Drohan
Mr. Colm Dunphy
Mr. Diarmuid O’Connor

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Pong Versions - introduction



v1 - **Ball moving** from left to right of screen. Can bounce off top or bottom

v2 - **Mouse controlling the Paddle**

v3 - **Collision detection** (ball bounces back). Changes made only to PongGame

v4 - **Game Over** (when 3 lives gone), Score (lives Lost). Output to Console. Changes made only to PongGame.

v5 - **Tournament** (no of games per tournament default is 5). Changes made only to PongGame.

v6 - new **Player class using arrays** (no statistics)

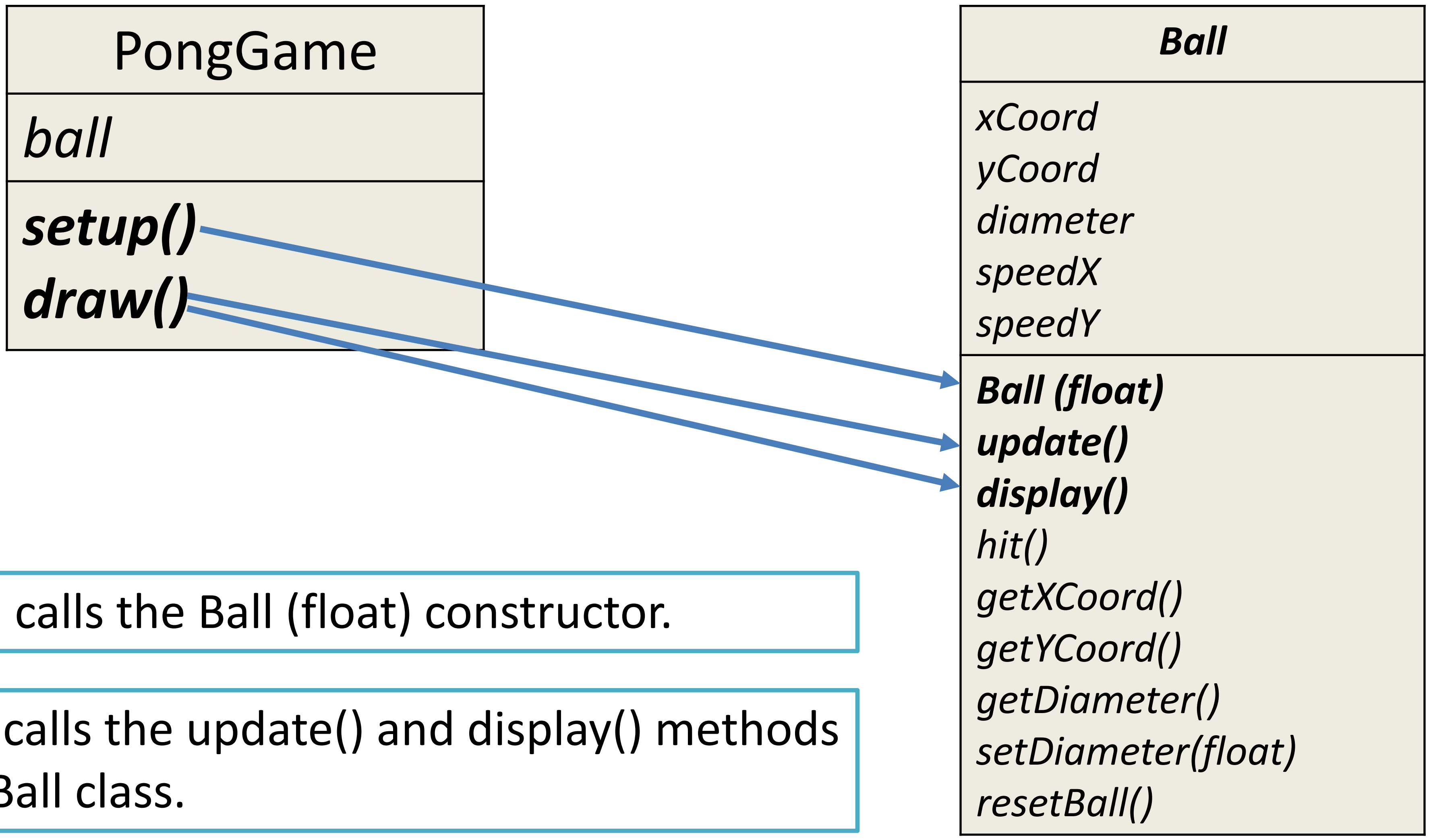
v7 - Player class using arrays (with **statistics** (Tournament Over - highest, lowest, average score))

v8 - **JOptionPane for I/O** instead of console

v9 - alternative algorithm using **Pythagoras Theorem**

Demo of
Pong Game V1.0

Classes in the PongGameV1.0

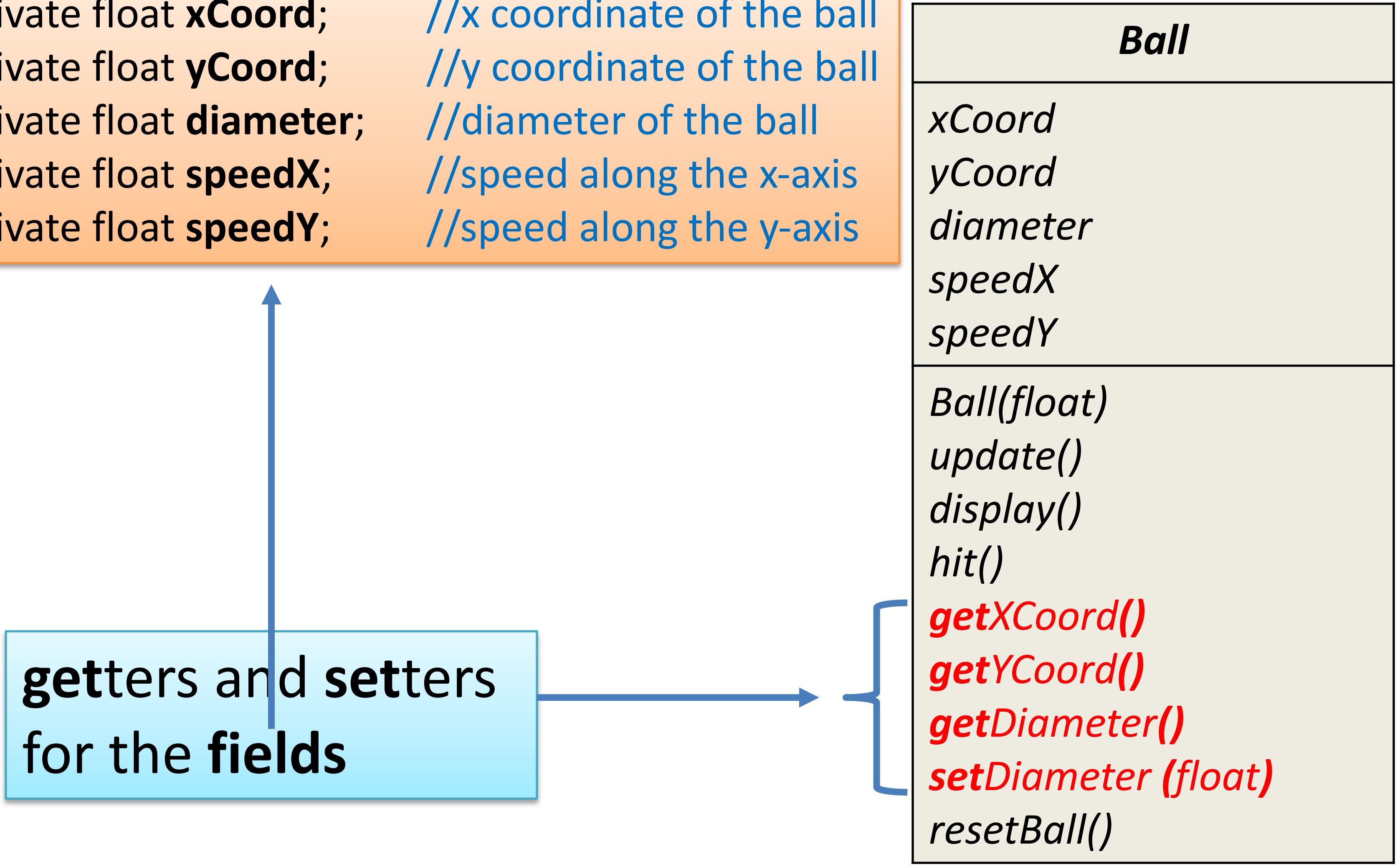


setup() calls the Ball (float) constructor.

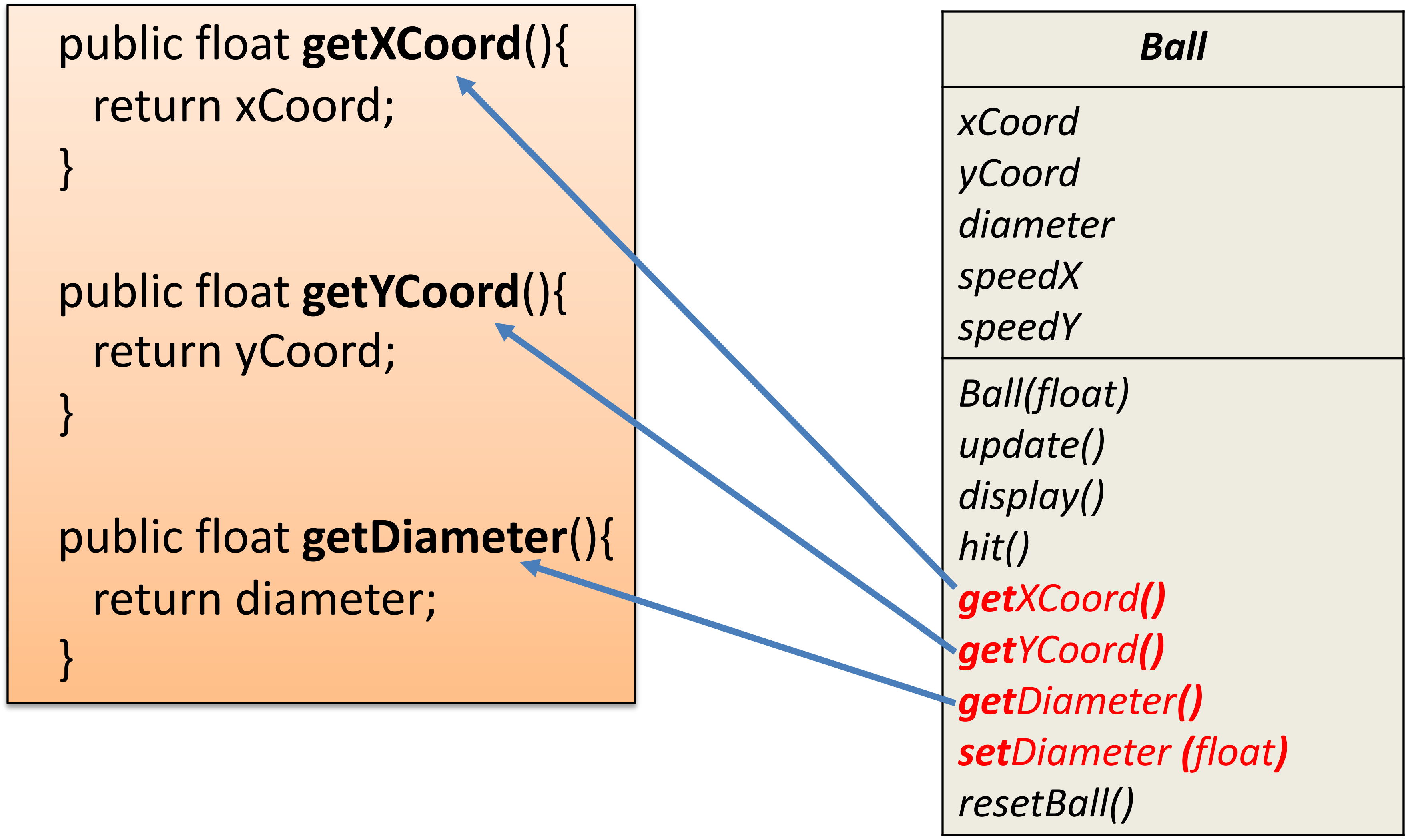
draw() calls the update() and display() methods in the Ball class.

Ball Class – instance **fields**

```
private float xCoord;      //x coordinate of the ball
private float yCoord;      //y coordinate of the ball
private float diameter;    //diameter of the ball
private float speedX;      //speed along the x-axis
private float speedY;      //speed along the y-axis
```



Ball Class – **getters**



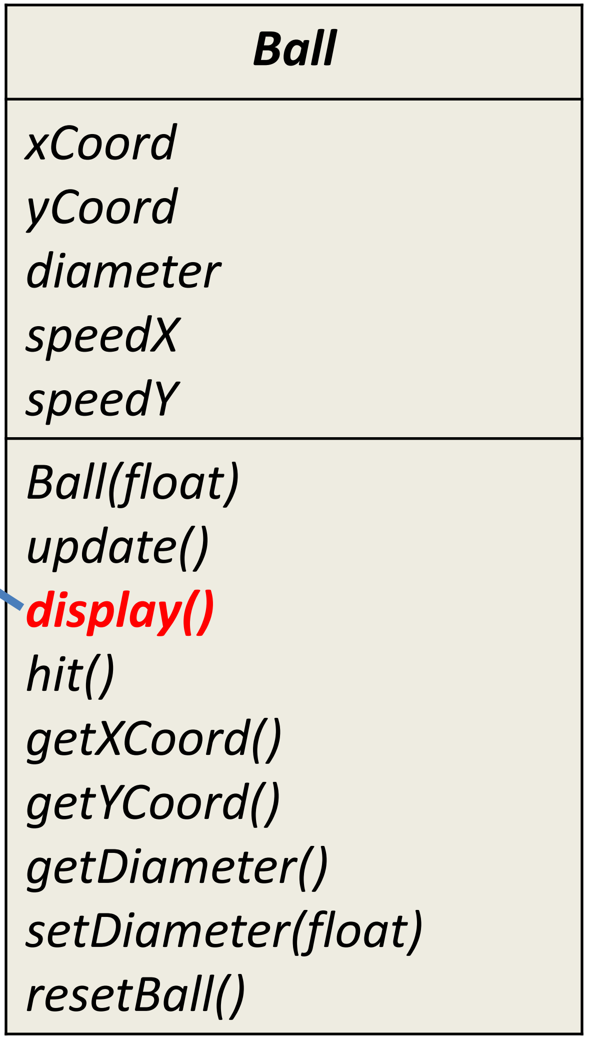
```
public float getXCoord(){
  return xCoord;
}

public float getYCoord(){
  return yCoord;
}

public float getDiameter(){
  return diameter;
}
```

Ball Class – **setter**

```
public void setDiameter (float diameter){

  //The ball diameter must be between 20 and height/6 (inclusive)
   if ((diameter >= 20) && (diameter <= height/6)){
     this.diameter = diameter;                          VALIDATION
   }
   else {
    // If an invalid diameter is passed as a parameter, a default of 20 is imposed.
    // With this animation, if we do not supply a default value for the diameter,
    // a ball may not be drawn on the display window.
    // Important note:
    // it is not always appropriate to provide a default value at setter) level;
    // this will depend on your design.
     this.diameter = 20;                                INITIALISATION
   }
 }
```

display() method

```
public void display(){
  fill(255);
  noStroke();
  ellipse(xCoord, yCoord, diameter, diameter);
}
```



Ball

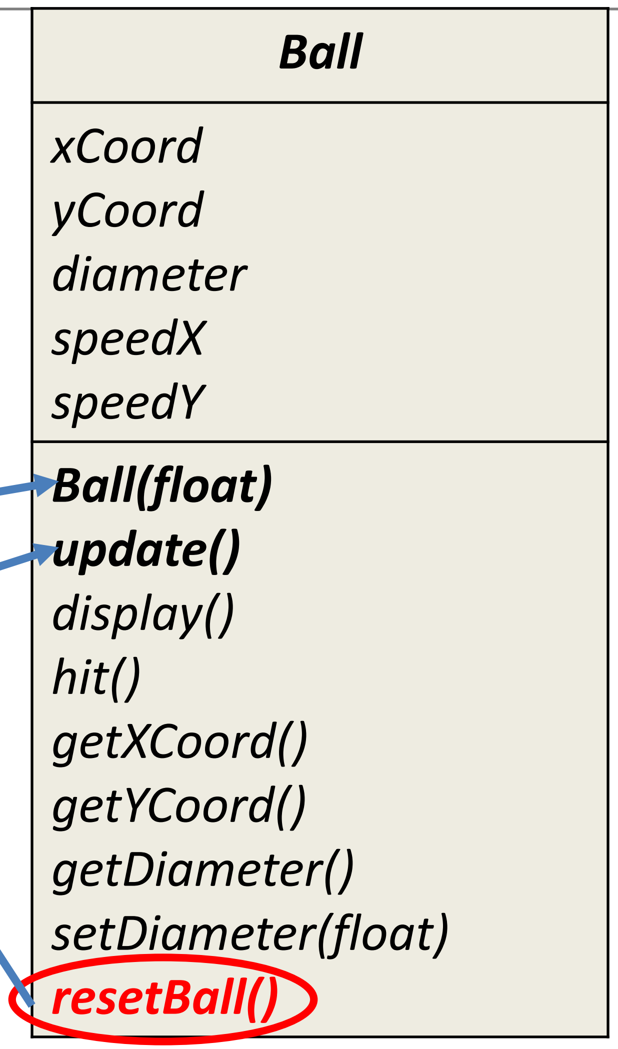
xCoord
yCoord
diameter
speedX
speedY

Ball(float)
update()
display()
hit()
getXCoord()
getYCoord()
getDiameter()
setDiameter(float)
resetBall()

Draws a white ball,
with no outline
on the display window.

private helper method – resetBall()

```
private void resetBall(){
   xCoord = 0;
   yCoord = random(height);
   speedX = random(3, 5);
   speedY = random(-2, 2);
}
```



Ball

xCoord
yCoord
diameter
speedX
speedY

Ball(float)
update()
display()
hit()
getXCoord()
getYCoord()
getDiameter()
setDiameter(float)
resetBall()

The **resetBall** method is used by the **Ball** constructor and the **update** method.

private helper method
→ **private** to the class you are in

i.e. can't use it outside of the current class.

A note on **random()**

```
private void resetBall(){
  xCoord = 0;
  yCoord = random (height);
  speedX = random (3, 5);
  speedY = random (-2, 2);
}
```

random (high)

returns a random float between **zero** (inclusive) and high (exclusive).

random (low, high)

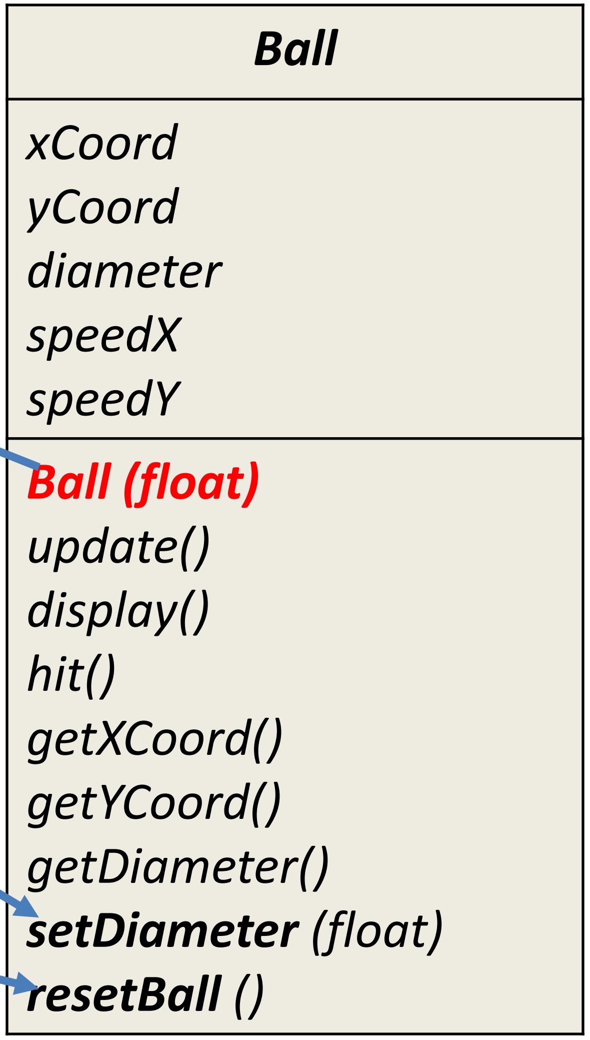
returns a random float between **low** (inclusive) and high (exclusive).

Ball constructor

```
public Ball (float diameter){
   setDiameter(diameter);
   resetBall();
}
```

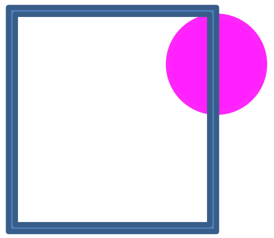
Constructor takes in the diameter of the ball and uses the **setDiameter** *setter method* to update the diameter instance field.

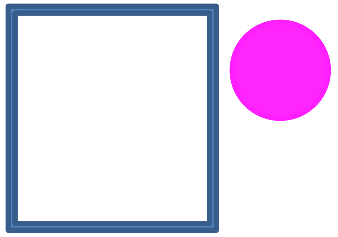
private helper method **resetBall** is called to set up the xCoord with zero and yCoord, speedX and speedY with random values



Recap – Drawing Modes: **ellipse**

- The default ellipse mode is CENTER
  - This means x & y positions for ellipse() specify the **center** of the ellipse
  - At the max width of the window, half the ellipse is seen
  - If we specify an x value > width + radius of the circle the circle has left the screen





update() method

update() changes the ball position.

if the ball…

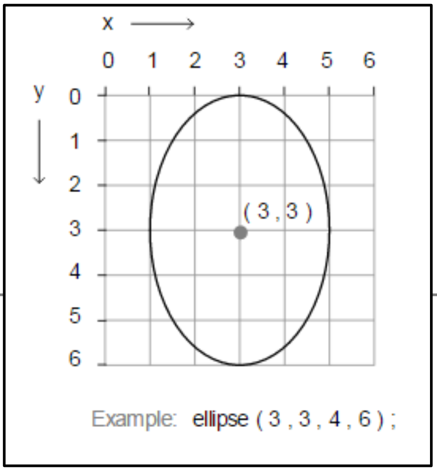
goes **off the screen**
return *true* (i.e. a life was lost)

hits the **left edge**
Change **xCoord** direction

hits the **top or bottom**
Change **yCoord** direction

```
public boolean update(){

  boolean lifeLost = false;

  //update ball coordinates
  xCoord = xCoord + speedX;
  yCoord = yCoord + speedY;

  //reset position if ball leaves the screen
  if (xCoord > width + diameter/2){
    resetBall();
    lifeLost = true;
  }
  // If ball hits the left edge of the display
  // window, change direction of xCoord
  if (xCoord < diameter/2)
    xCoord = diameter/2;
    speedX = speedX * -1;
  }

  // If ball hits top or bottom of the display
  //  window, change direction of yCoord
  if (yCoord > height - diameter/2){
    yCoord = height - diameter/2;
    speedY = speedY * -1;
  }
  else if (yCoord < diameter/2){
    yCoord = diameter/2;
    speedY = speedY * -1;
  }
  return lifeLost;
}
```

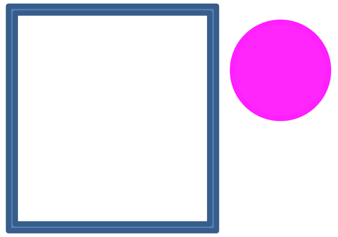
update() – explained 1



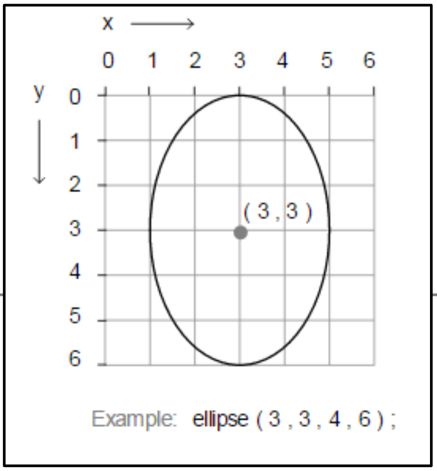
```
//reset position if ball leaves the screen
if (xCoord > width + diameter/2){
  resetBall();
  lifeLost = true;
}
```

(width + diameter/2)

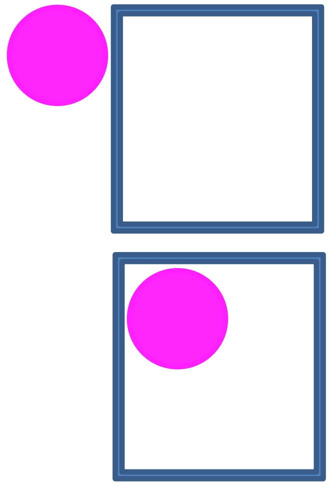
In this check, we add diameter/2 *(i.e. the radius)*
onto the width of the window
so that the ball is completely off the screen
because the x,y values specify the CENTER of the circle



update() – explained 2



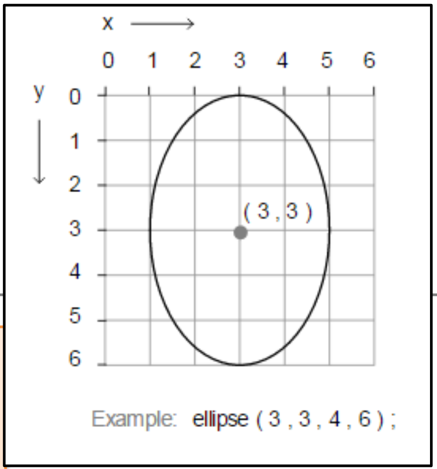
```
// If ball hits the left edge of the display
// window, change direction of xCoord
if (xCoord < diameter/2)
  xCoord = diameter/2;
  speedX = speedX * -1;
}
```

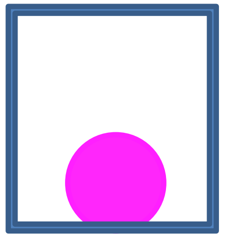


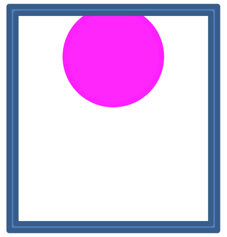
If the **xCoord** is less than the radius of the circle,
the circle has hit the left side

→ reset the xCoord to the radius of the circle
and reverse the speedX variable by multiplying by -1.

update() – explained 3







```
// If ball hits top or bottom of the display window,
// change direction of yCoord
if (yCoord > height - diameter/2){   // bottom
  yCoord = height - diameter/2;
  speedY = speedY * -1;
}
else if (yCoord < diameter/2){       // top
  yCoord = diameter/2;
  speedY = speedY * -1;
}
```

The **yCoord** is investigated to see if the ***top*** or ***bottom*** of the screen was hit.

(yCoord < diameter/2)

(yCoord > height - diameter/2)

hit() method

```
public void hit (){
    speedX = speedX * -1;
    xCoord = xCoord + speedX;
}
```

We’re not using this method in this version of Pong.

We’re preparing our class for **collision detection** in V3.0.

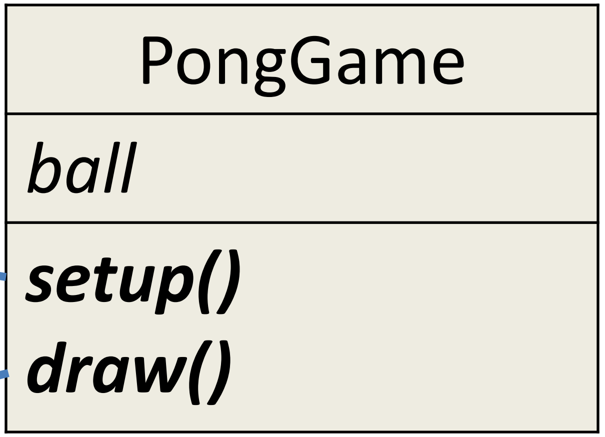
This method **changes the ball direction** when it hits the paddle.
It **bumps it back to the edge of the paddle.**

Ball
xCoord *yCoord* *diameter* *speedX* *speedY*

Ball(float)
update()
display()
hit()
getXCoord()
getYCoord()
getDiameter()
setDiameter(float)
resetBall()

PongGame **V1.0**

```
Ball ball;

void setup() {
  size(600,600);
  noCursor();
  //setting up the ball with hard-coded sizes.
  ball = new Ball(20.0);
}

void draw() {
  background(0);
  //Update the ball position and display it.
  ball.update();
  ball.display();
}
```



Questions?

References

- Reas, C. & Fry, B. (2014) Processing – A Programming Handbook for Visual Designers and Artists, 2nd Edition, MIT Press, London.