More on Classes

Adding behaviour

Produced by: Mairead Meagher
Dr. Siobhán Drohan

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Topics list

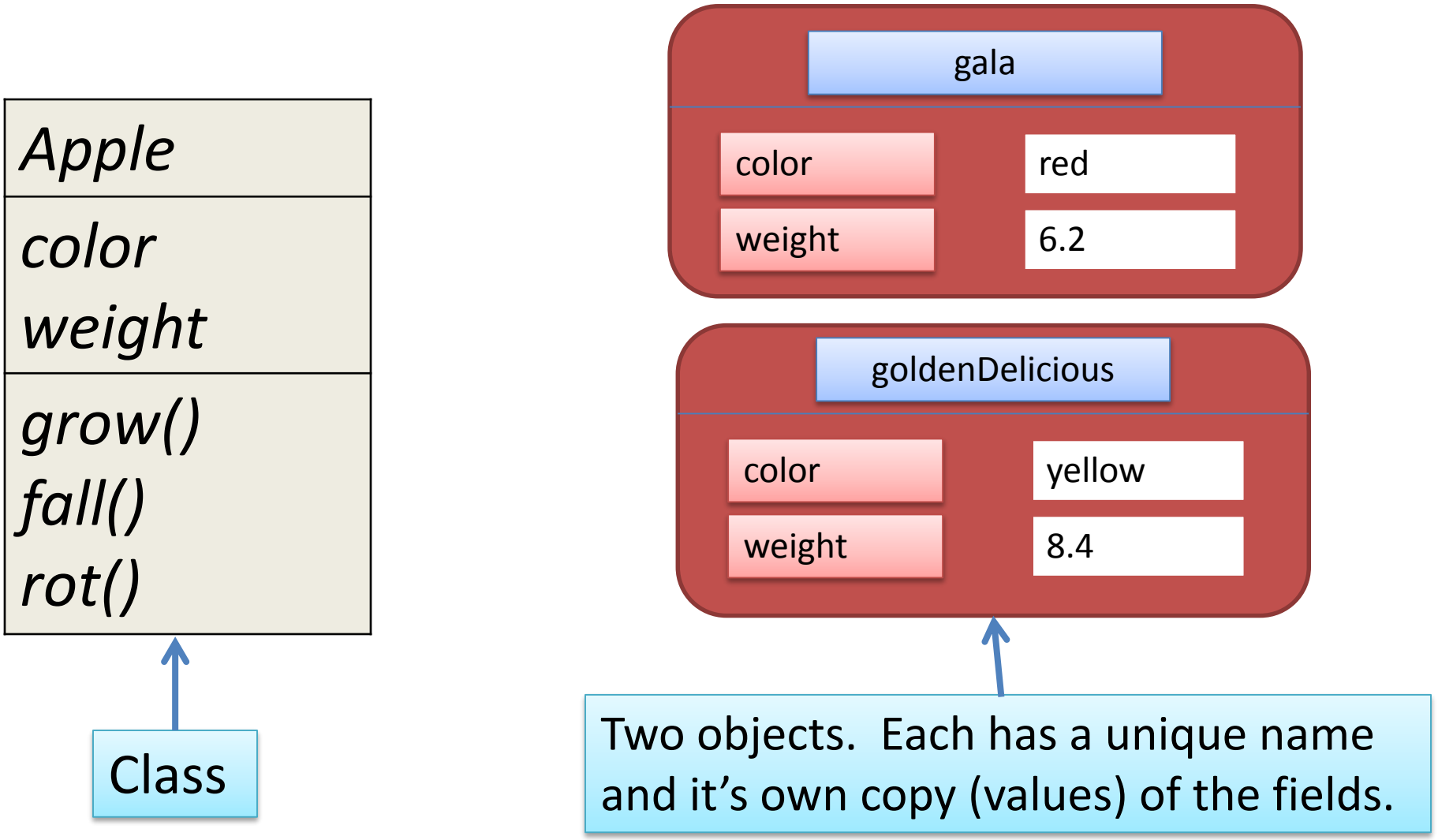
- Recap: Classes and Objects
- Recap on the Spot class:
  - Version 1.0 (Spot class with a default constructor)
  - Version 2.0 (writing a constructor with parameters)
  - Version 3.0 (overloading constructors)
- Adding behaviours to the Spot class:
  - Version 4.0 (display method)
  - Version 5.x (colour methods)
  - Version 6.0 (move method)
  - Version 6.1 (this keyword)

Recap: Classes and Objects

- A class defines a group of related methods (functions) and fields (variables).
- An object is a single instance of a class i.e. an object is created from a class.
- Objects can be related to real-world artefacts.
- Many objects can be constructed from a single class definition.
- Each object must have a unique name within the program.

Source: Reas & Fry (2014)

Recap: Apple Class and two Objects



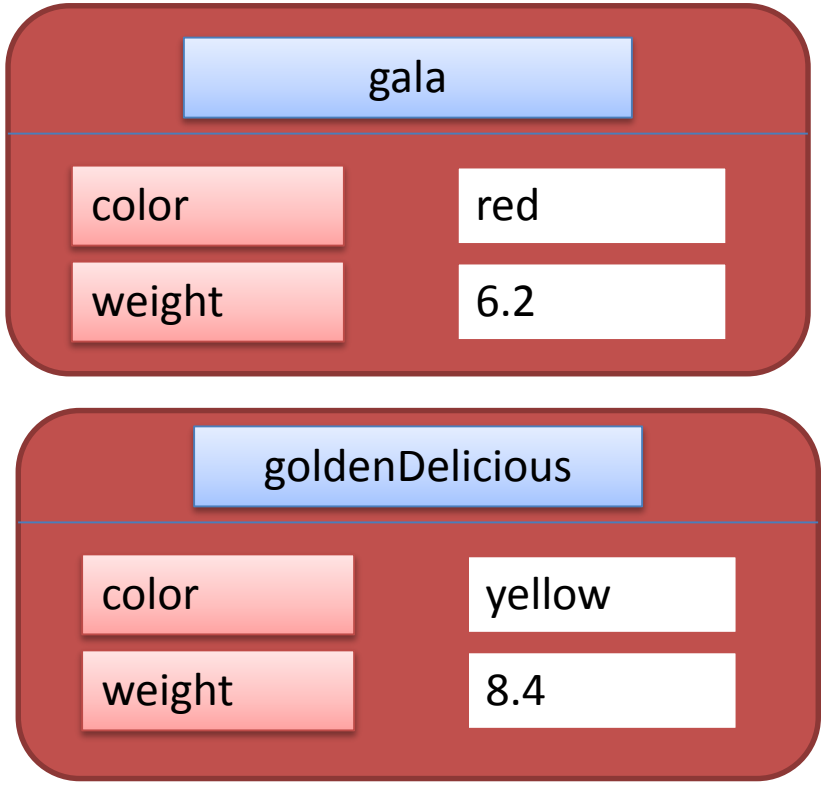
Source: Reas & Fry (2014)

Recap: Object State

There are two objects of type Apple.

Each object has a unique name:
- gala
- goldenDelicious

Each object has a different **object state**:
- each object has it's own copy of the fields (color and weight) in memory and has it's own data stored in these fields.



Source: Reas & Fry (2014)

Recap: Using an Object's fields and methods

- The fields and methods of an object are accessed with the dot operator i.e. external calls.

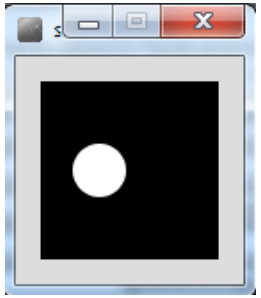
gala.color	Gives access to the color value in the gala object.
goldenDelicious.color	Gives access to the color value in the goldenDelicious object.
gala.grow()	Runs the grow() method inside the gala object.
goldenDelicious.fall()	Runs the fall() method inside the goldenDelicious object.

Source: Reas & Fry (2014)

Topics list

- Recap: Classes and Objects
- Recap on the Spot class:
  - Version 1.0 (Spot class with a default constructor)
  - Version 2.0 (writing a constructor with parameters)
  - Version 3.0 (overloading constructors)
- Adding behaviours to the Spot class:
  - Version 4.0 (display method)
  - Version 5.x (colour methods)
  - Version 6.0 (move method)
  - Version 6.1 (this keyword)

Recap: Spot Class – Version 1.0



Defining the class

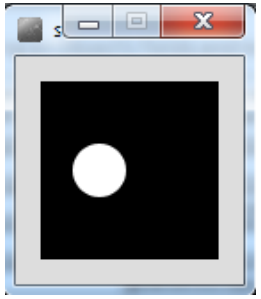
```
class Spot
{
  float xCoord, yCoord;
  float diameter;
}
```

Declaring the fields in the class

Place this code in a tab, called Spot

Source: Reas & Fry (2014)

Recap: Spot Class – Version 1.0



Declaring an object **sp**, of type **Spot**.

Calling the **Spot()** *constructor* to build the **sp** object in memory.

Initialising the fields in the **sp** object with a starting value.

Calling the ellipse method, using the fields in the **sp** object as arguments.

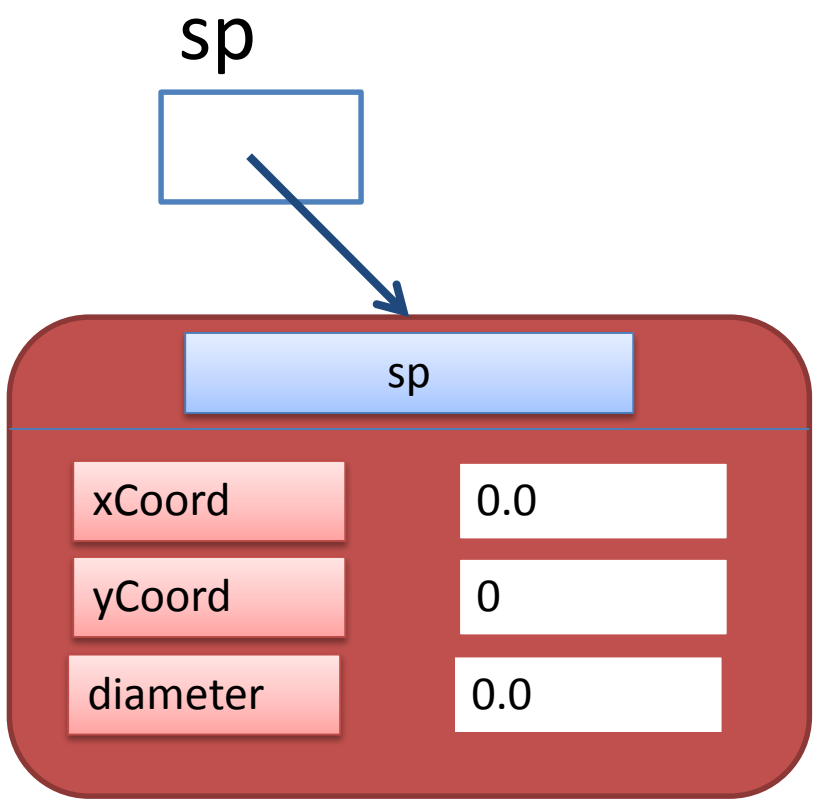
```
Spot sp;

void setup(){
  size (100,100);
  noStroke();
  sp = new Spot();
  sp.xCoord = 33;
  sp.yCoord = 50;
  sp.diameter = 30;
}

void draw(){
  background(0);
  ellipse(sp.xCoord, sp.yCoord,
                    sp.diameter, sp.diameter);
}
```

Recap: Constructors

Spot sp;

sp

sp = new **Spot()**;



Source: Reas & Fry (2014)

Recap: Constructors

```
Spot sp;
sp = new Spot();
```

The **sp** object is constructed with the keyword new.

- **Spot()** is the default constructor that is called to build the **sp** object in memory.
- A constructor is a method that has the same name as the class but has no return type

```
Spot()
{
}
```

Source: Reas & Fry (2014)

Recap: Default Constructor

```
class Spot
{
    float xCoord;
    float yCoord;
    float diameter;

     //Default Constructor
     Spot()
     {
     }
}
```

- The default constructor has an empty parameter list.
- If you don't include a constructor in your class, the compiler inserts a default one for you (in the background…you won't see it in your code).
- Here, the Spot() default constructor simply constructs the object.

Source: Reas & Fry (2014)

Topics list

- Recap: Classes and Objects
- Recap on the Spot class:
  - Version 1.0 (Spot class with a default constructor)
  - Version 2.0 (writing a constructor with parameters)
  - Version 3.0 (overloading constructors)
- Adding behaviours to the Spot class:
  - Version 4.0 (display method)
  - Version 5.x (colour methods)
  - Version 6.0 (move method)
  - Version 6.1 (this keyword)

Recap: Writing our first constructor

- Constructors can store initial values in an object's fields.
- They often receive external parameter values for this.
- In this code, we initialised the xCoord, yCoord and diameter after calling the Spot() constructor.

```
Spot sp;

void setup(){
  size (100,100);
  noStroke();
  sp = new Spot();
  sp.xCoord = 33;
  sp.yCoord = 50;
  sp.diameter = 30;
}

void draw(){
  background(0);
  ellipse(sp.xCoord, sp.yCoord,
                  sp.diameter, sp.diameter);
}
```

Source: Reas & Fry (2014)

Recap: Writing our first constructor

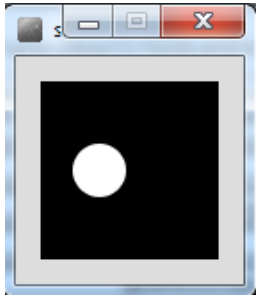
- We want to write another constructor that will take three parameters
  - xPos,
  - yPos
  - diamtr
- These values will be used to initialise the
  - xCoord,
  - yCoord and
  - diameter

variables.

```
Spot sp;

void setup(){
  size (100,100);
  noStroke();
  sp = new Spot();
  sp.xCoord = 33;
  sp.yCoord = 50;
  sp.diameter = 30;
}

void draw(){
  background(0);
  ellipse(sp.xCoord, sp.yCoord,
                  sp.diameter, sp.diameter);
}
```

Source: Reas & Fry (2014)

Recap: Spot Class – Version 2.0



```
Spot sp;

void setup()
{
  size (100,100);
  noStroke();
  sp = new Spot(33, 50, 30);
}

void draw()
{
  background(0);
  ellipse(sp.xCoord, sp.yCoord, sp.diameter, sp.diameter);
}
```

```
class Spot
{
  float xCoord, yCoord;
  float diameter;

  Spot(float xPos, float yPos, float diamtr)
  {
    xCoord = xPos;
    yCoord = yPos;
    diameter = diamtr;
  }
}
```

Source: Reas & Fry (2014)

Topics list

- Recap: Classes and Objects
- Recap on the Spot class:
  - Version 1.0 (Spot class with a default constructor)
  - Version 2.0 (writing a constructor with parameters)
  - Version 3.0 (overloading constructors)
- Adding behaviours to the Spot class:
  - Version 4.0 (display method)
  - Version 5.x (colour methods)
  - Version 6.0 (move method)
  - Version 6.1 (this keyword)

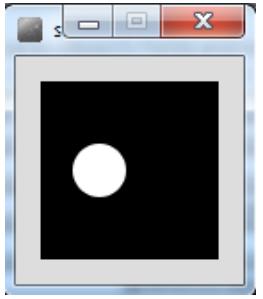
Recap: Overloading Constructors

- We can have as many constructors as our design requires, ONCE they have unique parameter lists.

- We are overloading our constructors in Version 3.0…

Source: Reas & Fry (2014)

Recap: Spot Class – Version 3.0



```
Spot sp;

void setup()
{
  size (100,100);
  noStroke();
  sp = new Spot(33, 50, 30);
}

void draw()
{
  background(0);
  ellipse(sp.xCoord, sp.yCoord, sp.diameter, sp.diameter);
}
```

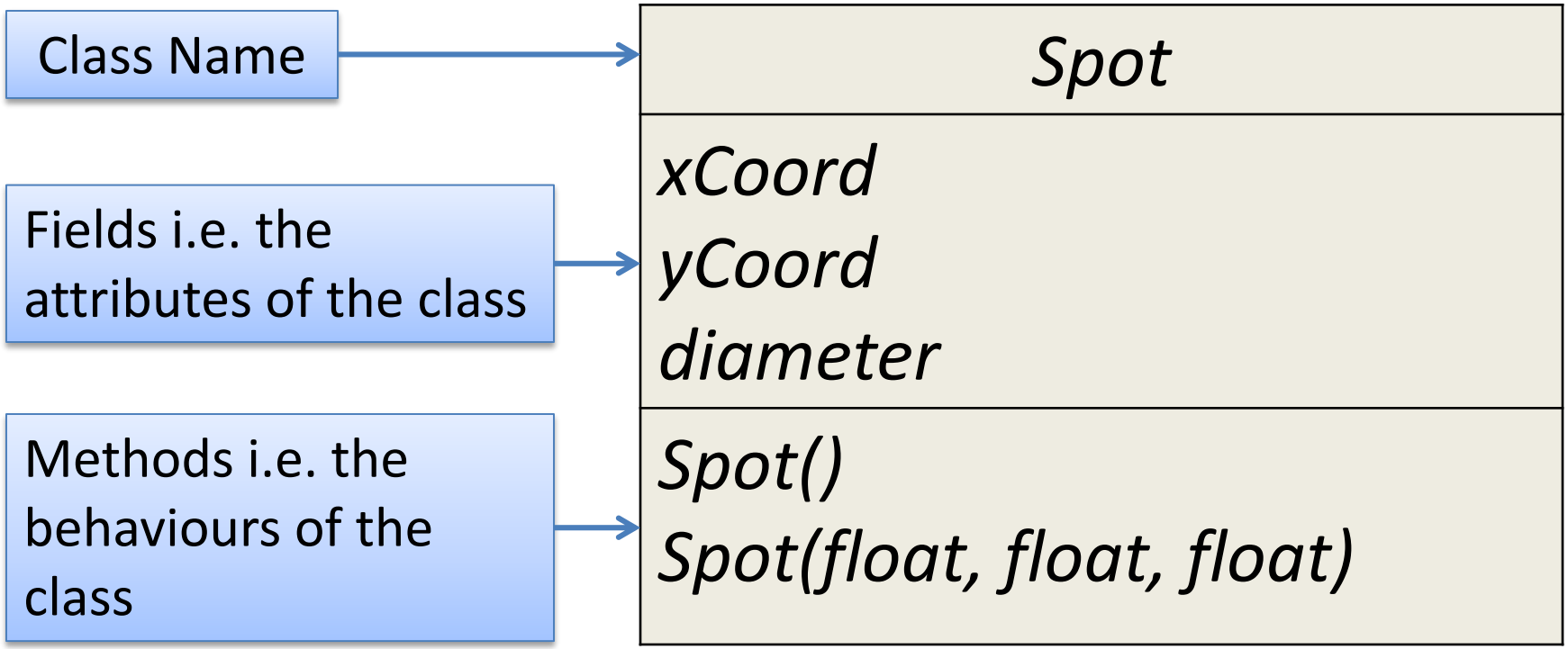
```
class Spot{
  float xCoord, yCoord;
  float diameter;

  Spot(){
  }

 Spot(float xPos, float yPos, float diamtr){
    xCoord = xPos;
    yCoord = yPos;
    diameter = diamtr;
  }
}
```

Source: Reas & Fry (2014)

Topics list

- Recap: Classes and Objects
- Recap on the Spot class:
  - Version 1.0 (Spot class with a default constructor)
  - Version 2.0 (writing a constructor with parameters)
  - Version 3.0 (overloading constructors)
- Adding behaviours to the Spot class:
  - Version 4.0 (display method)
  - Version 5.x (colour methods)
  - Version 6.0 (move method)
  - Version 6.1 (this keyword)

Class Diagram for Spot Version 3.0



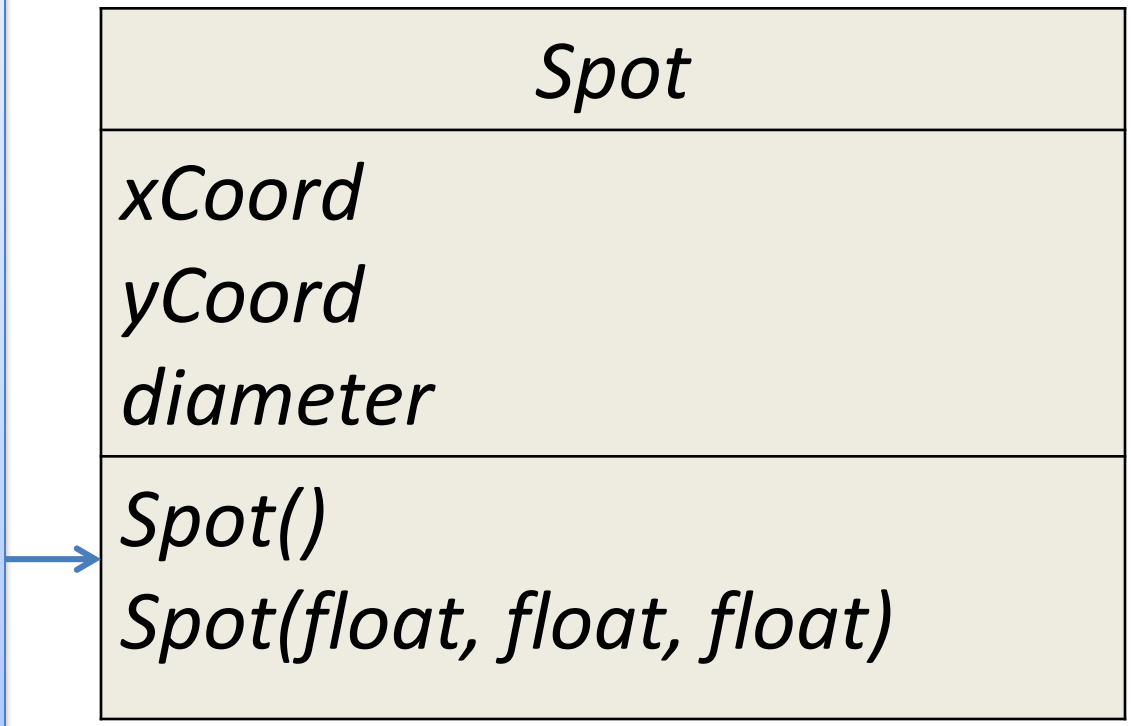
Source: Reas & Fry (2014)

Class Diagram for Spot Version 3.0

So far, we only have constructors for our class (they create the objects of our class).

We have not defined any behaviours for our class e.g.
- display the spot,
- colour the spot,
- move the spot,
- and so on.

As it stands, the Spot class is not very useful!



Source: Reas & Fry (2014)

Spot – adding a “display” behaviour

- We want to add a behaviour to the Spot class that will draw the Spot on the screen.
- To add behaviour to a class, we write a method inside the class.
- We will call this method display().

Source: Reas & Fry (2014)

display() method

- The method signature is:

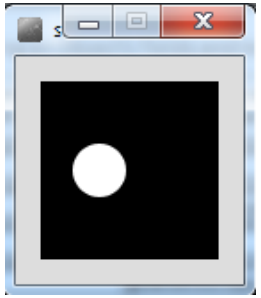
  void display()

- The method's job is to draw the spot on the display window.

```
void display()
{
   ellipse(xCoord, yCoord, diameter, diameter);
}
```

Source: Reas & Fry (2014)

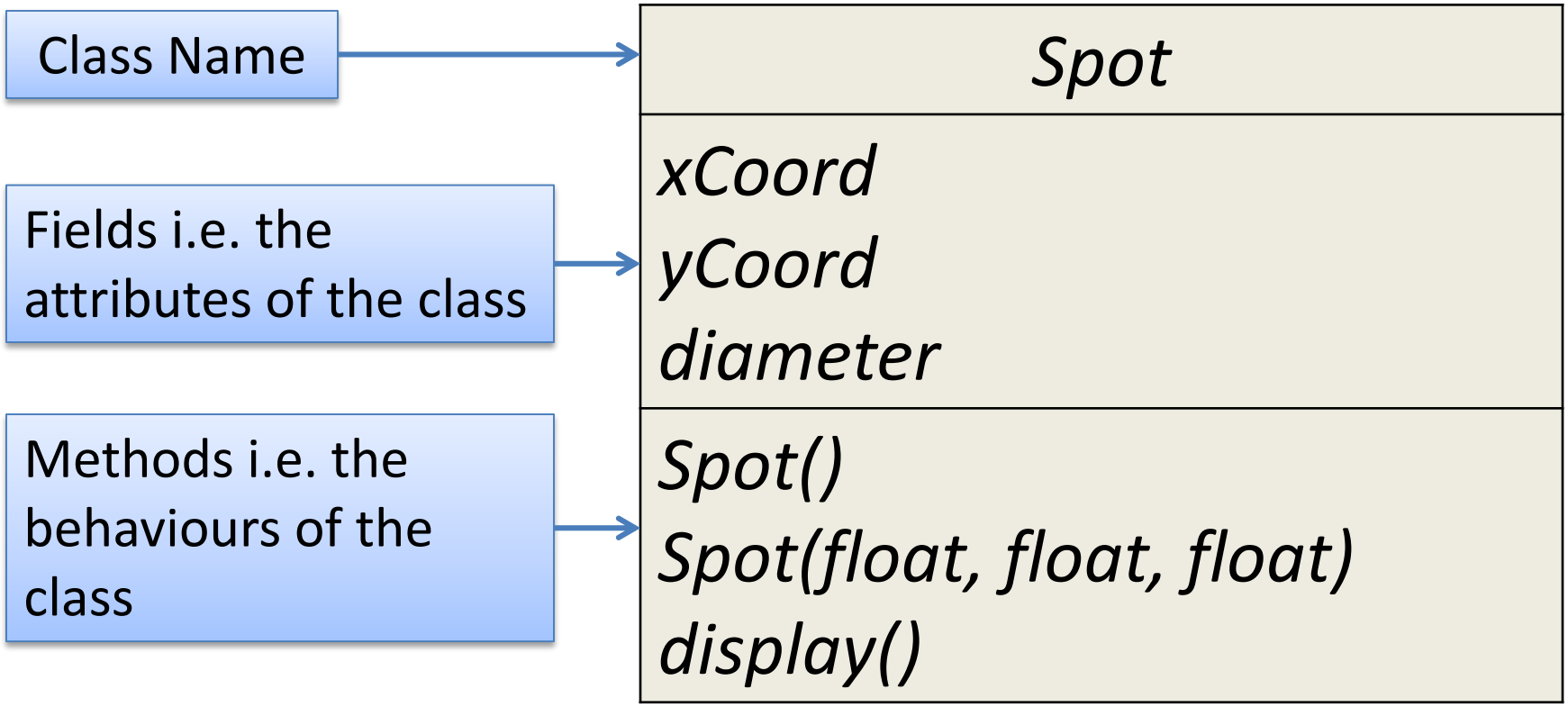
Spot Class – Version 4.0



```
Spot sp;

void setup()
{
  size (100,100);
  noStroke();
  sp = new Spot(33, 50, 30);
}

void draw()
{
  background(0);
  sp.display();
}
```

```
class Spot{
  float xCoord, yCoord;
  float diameter;

  Spot(){
  }

 Spot(float xPos, float yPos, float diamtr){
    xCoord = xPos;
    yCoord = yPos;
    diameter = diamtr;
  }

  void display(){
    ellipse(xCoord, yCoord, diameter, diameter);
  }
}
```

Class Diagram for Spot Version 4.0



Source: Reas & Fry (2014)

Topics list

- Recap: Classes and Objects
- Recap on the Spot class:
  - Version 1.0 (Spot class with a default constructor)
  - Version 2.0 (writing a constructor with parameters)
  - Version 3.0 (overloading constructors)
- Adding behaviours to the Spot class:
  - Version 4.0 (display method)
  - Version 5.x (colour methods)
  - Version 6.0 (move method)
  - Version 6.1 (this keyword)

Spot – adding **RGB** "colour" behaviour

- We now want to add a behaviour to the Spot class that will colour the Spot, using RGB values on the screen.

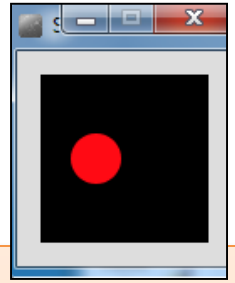
- To add this behaviour, we will need three extra attributes (fields / variables):

  int red
  int green
  int blue

- We will need to take in values for the red, green and blue fields using the parameters of our new method e.g.:

  colour(int redVal, int greenVal, int blueVal)

Spot Class – Version 5.0



```
Spot sp;

void setup()
{
  size (100,100);
  noStroke();
  sp = new Spot(33, 50, 30);
}

void draw()
{
  background(0);
  sp.colour(255,10,20);
  sp.display();
}
```

```
class Spot{
  float xCoord, yCoord;
  float diameter;
  int red, green, blue;

  // constructors…

  void display(){
    ellipse(xCoord, yCoord, diameter, diameter);
  }

  void colour(int redVal, int greenVal, int blueVal){
    red = redVal;
    green = greenVal;
    blue = blueVal;
    fill (red, green, blue);
  }
}
```

Spot – **Grayscale**"colour" behaviour

- We now want to add a behaviour to the Spot class that will colour the Spot, using a Grayscale value on the screen.

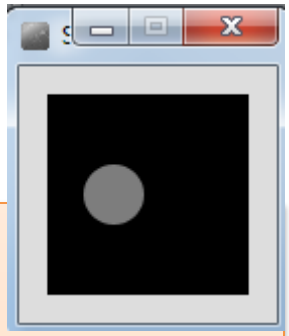
- To add this behaviour, we will need one extra attribute (field / variable):

  int gray

- We will need to take in a value for the gray field using the parameters of our new method e.g.:

  colour(int grayVal)

Spot Class – Version 5.1



```
Spot sp;

void setup()
{
  size (100,100);
  noStroke();
  sp = new Spot(33, 50, 30);
}

void draw()
{
  background(0);
  sp.colour(125);
  sp.display();
}
```

```
class Spot{
  float xCoord, yCoord;
  float diameter;
  int red, green, blue, gray;

  // constructors…
  //display method…
  void colour(int redVal, int greenVal, int blueVal){
    red = redVal;
    green = greenVal;
    blue = blueVal;
    fill (red, green, blue);
  }

 void colour(int grayVal){
    gray = grayVal;
    fill (gray);
  }
}
```

Spot – two colour behaviours

- We have overloaded the colour method i.e. we have two methods called colour that have different parameter lists:

  colour(int redVal, int greenVal, int blueVal)
  colour(int grayVal)

- Java knows which method to call based on matching the arguments passed to the method call.

Spot – two colour behaviours

```
void draw()
{
  background(0);
  sp.colour(255,10,20);
  sp.display();
}
```

```
void draw()
{
  background(0);
  sp.colour(125);
  sp.display();
}
```

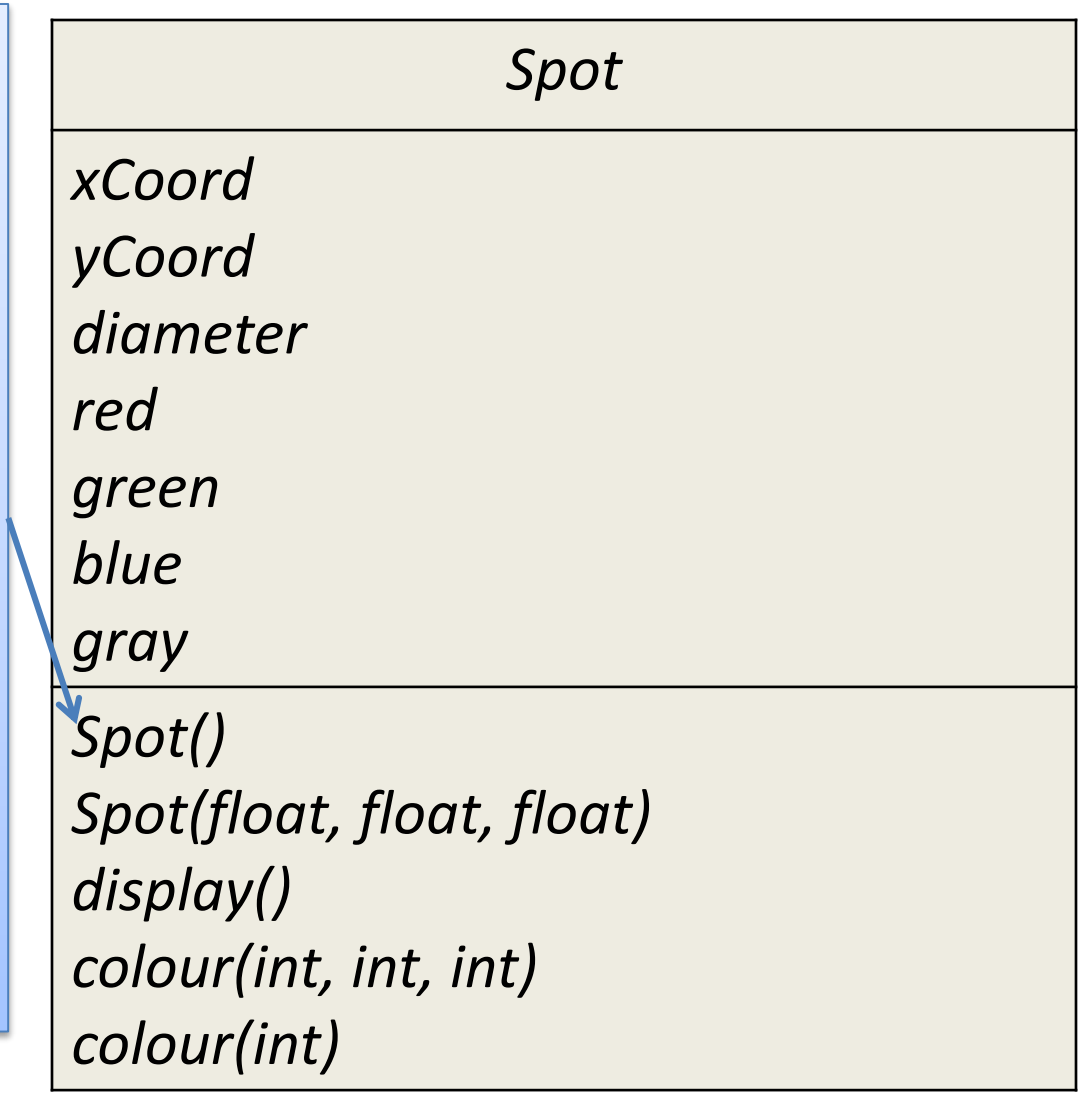
```
class Spot{
   //variables…
  // constructors…
  //display method…
  void colour(int redVal, int greenVal, int blueVal){
    red = redVal;
    green = greenVal;
    blue = blueVal;
    fill (red, green, blue);
   }

 void colour(int grayVal){
    gray = grayVal;
    fill (gray);
  }
}
```

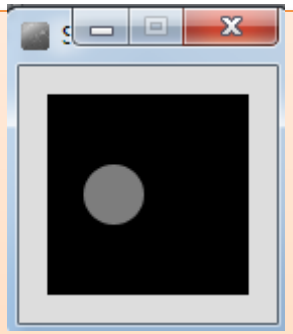
Class Diagram for Spot Version 5.1

We have two constructors in our class. But these constructors do not initialise our new fields, red, green, blue or grey.

We need to add two new constructors that will initialise our Spot object to a starting:

- gray colour.
- RGB colour.



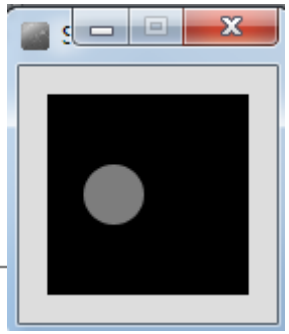


```
class Spot{
  // variables…
  // other constructors…
  Spot(float xPos, float yPos, float diamtr, int grayVal){
     xCoord = xPos;
     yCoord = yPos;
     diameter = diamtr;
     colour(grayVal);
   }

   Spot(float xPos, float yPos, float diamtr, int redVal, int greenVal, int blueVal){
     xCoord = xPos;
     yCoord = yPos;
     diameter = diamtr;
     colour(redVal, greenVal, blueVal);
   }
  // display method…
  // colour methods…
}
```

Spot Class – Version 5.2

Source: Reas & Fry (2014)

Using the “GrayScale” constructor

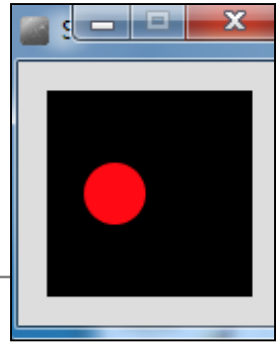


```
Spot sp;

void setup(){
  size (100,100);
  noStroke();
  sp = new Spot(33, 50, 30, 125);
}

void draw(){
  background(0);
  sp.display();
}
```

Spot Class – Version 5.2

Source: Reas & Fry (2014)

Using the “RGB” constructor

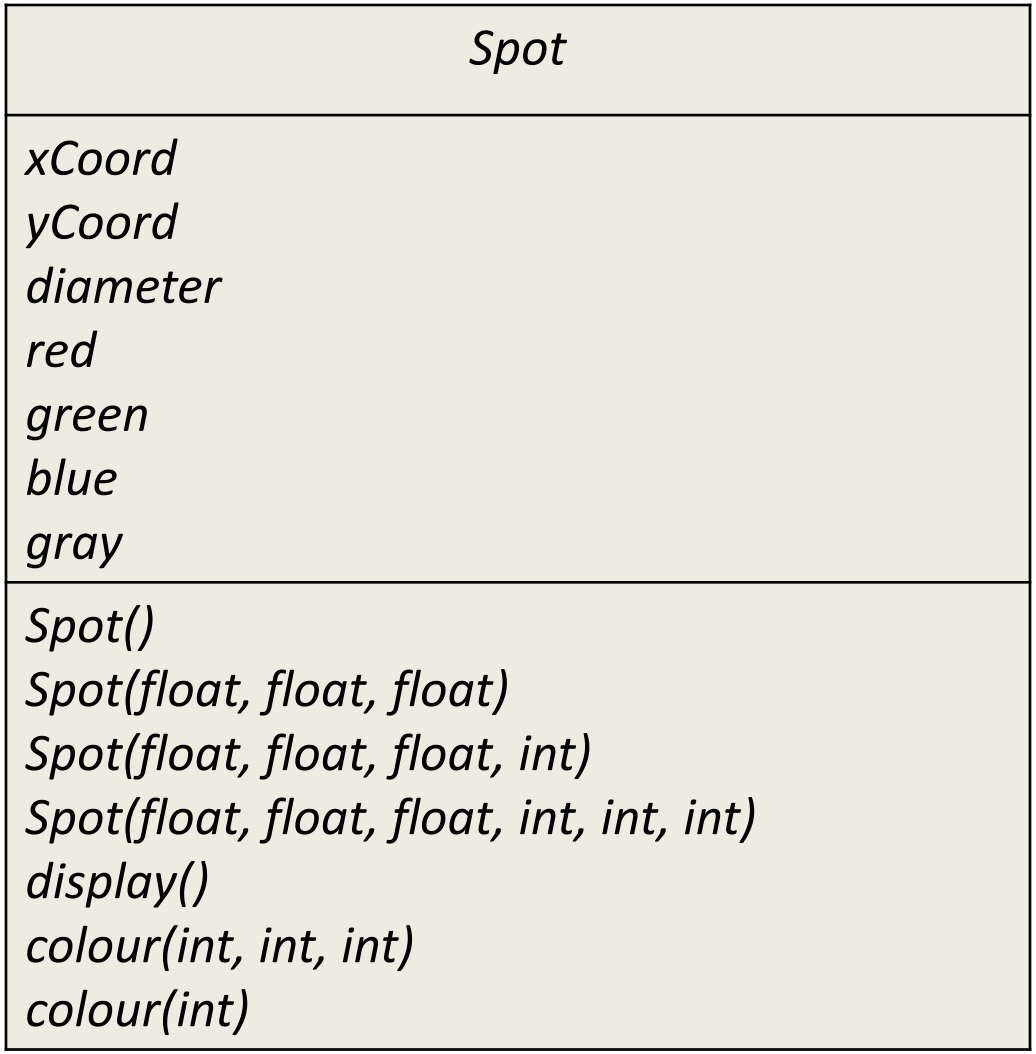


```
Spot sp;

void setup(){
  size (100,100);
  noStroke();
  sp = new Spot(33, 50, 30, 255,10,20);
}

void draw(){
  background(0);
  sp.display();
}
```

Spot Class – Version 5.2

Source: Reas & Fry (2014)

Class Diagram for Spot Version 5.2



Topics list

- Recap: Classes and Objects
- Recap on the Spot class:
  - Version 1.0 (Spot class with a default constructor)
  - Version 2.0 (writing a constructor with parameters)
  - Version 3.0 (overloading constructors)
- Adding behaviours to the Spot class:
  - Version 4.0 (display method)
  - Version 5.x (colour methods)
  - Version 6.0 (move method)
  - Version 6.1 (this keyword)

Spot – adding a “move” behaviour

- We now want to add a behaviour to the Spot class that will move the Spot around the screen.

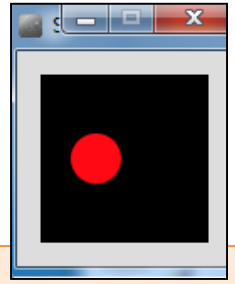
- To add this behaviour, we don’t need any extra attributes (fields / variables) as we already store the coordinates of the Spot:

  float xCoord
  float yCoord

- We will need to take in values for the new position of the Spot e.g.

  move(float xPos, float yPos)

Spot Class – Version 6.0



```
Spot sp;

void setup(){
  size (100,100);
  noStroke();
  sp = new Spot(33, 50, 30, 255,10,20);
}

void draw(){
  background(0);
  sp.display();
  sp.move(mouseX, mouseY);
}
```

```
class Spot{
  float xCoord, yCoord;
  float diameter;
  int red, green, blue;

  // constructors…
  // colour methods…
  void display(){
    ellipse(xCoord, yCoord, diameter, diameter);
  }

 void move(float xPos, float yPos)
   {
     xCoord = xPos;
     yCoord = yPos;
   }
}
```

Source: Reas & Fry (2014)

Class Diagram for Spot Version 6.0

Topics list

- Recap: Classes and Objects
- Recap on the Spot class:
  - Version 1.0 (Spot class with a default constructor)
  - Version 2.0 (writing a constructor with parameters)
  - Version 3.0 (overloading constructors)
- Adding behaviours to the Spot class:
  - Version 4.0 (display method)
  - Version 5.x (colour methods)
  - Version 6.0 (move method)
  - Version 6.1 (this keyword)

this keyword

- The class Spot contains many fields e.g.:
  - xCoord, yCoord, diameter
- One of the Spot constructors takes three parameters:
  - xPos, yPos, diamtr

```
class Spot{
  float xCoord, yCoord;
  float diameter;
  int red, green, blue;

  Spot(float xPos, float yPos, float diamtr)
  {
     xCoord = xPos;
     yCoord = yPos;
     diameter = diamtr;
  }
```

this keyword

- It would be nice to name the parameters passed into the Spot constructor the same names as the instance fields.

- This is called name overloading.
- But how will Java know which variable we are referring to?

```
class Spot{
  float xCoord, yCoord;
  float diameter;
  int red, green, blue;

  Spot(float xPos, float yPos, float diamtr)
  {
     xCoord = xPos;
     yCoord = yPos;
     diameter = diamtr;
  }
```

this keyword

- It would be nice to name the parameters passed into the Spot constructor the same names as the instance fields.

- This is called name overloading.
- But how will Java know which variable we are referring to?

```
class Spot{
  float xCoord, yCoord;
  float diameter;
  int red, green, blue;

  Spot(float xPos, float yPos, float diamtr)
  {
     xCoord = xPos;
     yCoord = yPos;
     diameter = diamtr;
  }
```

this keyword

- We can use the this keyword to distinguish between them.
- The expression this refers to the current object.

```
class Spot{
  float xCoord, yCoord;
  float diameter;
  int red, green, blue;

 Spot(float xCoord, float yCoord, float diameter)
 {
    this.xCoord = xCoord;
    this.yCoord = yCoord;
    this.diameter = diameter;
  }
```

this keyword

```
void colour(int red, int green, int blue)
 {
   this.red = red;
   this.green = green;
   this.blue = blue;
   fill (red, green, blue);
 }

 void colour(int gray){
   this.gray = gray;
   fill (this.gray);
 }
```

Questions?

References

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

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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