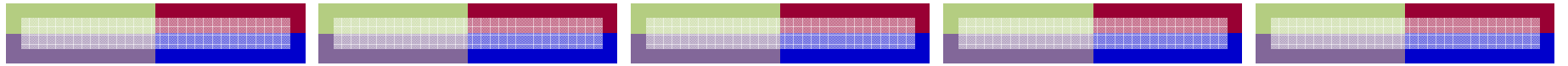


CS101

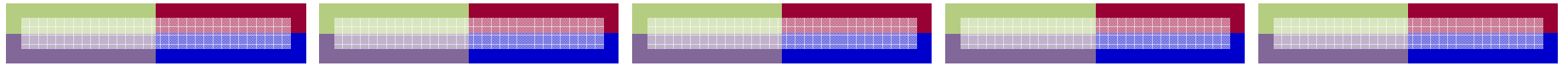
Problem Solving and Object-Oriented Programming

L13: Graphical Objects; Naming and Modifying Objects



- What is a constant?
- What is the advantage of using constant?
- What is a constructor?





Constructors

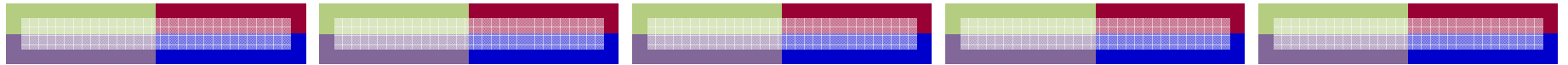
• **Similar to method.**

- it's called like a method
- has parameters like a method
- it returns.
-

• But it must have the same name as the class for which it is a constructor.

• Also, the type and return value are implicit.

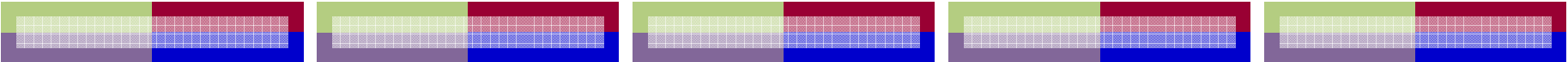




Stick Figure Class

```
public class StickFigure {  
    private static final int HEAD_SIZE = 100;  
    private static final int BODY_SIZE = HEAD_SIZE * 2;  
    ...  
    private static final int HEAD_LEFT = 100;  
    private static final int HEAD_TOP = 100;  
    private static final int BODY_LEFT = HEAD_LEFT + (HEAD_SIZE / 2);  
    private static final int BODY_TOP = HEAD_TOP + HEAD_SIZE;  
    ...  
    public StickFigure (DrawingCanvas figureCanvas) {  
        new FramedOval (HEAD_LEFT, HEAD_TOP,  
                        HEAD_SIZE, HEAD_SIZE, figureCanvas);  
        new Line (BODY_LEFT, BODY_TOP,  
                 BODY_LEFT, BODY_TOP + BODY_SIZE, figureCanvas);  
    }  
    ...  
}
```





Constructing a Stick Figure

- To construct a stick figure, you must call the constructor in the StickFigure class, defined as:

```
public StickFigure(DrawingCanvas figureCanvas) {  
    ...  
}
```

← Constructor header

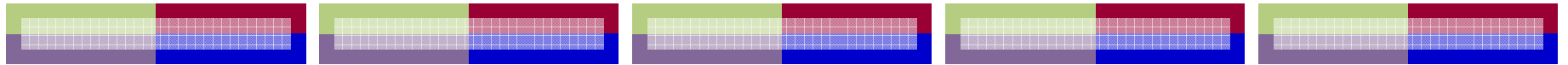
- To construct a stick figure, you must call the constructor in the StickFigure class, defined as:

```
new StickFigure(?????);
```

← Constructor call

new means to call a constructor





begin Method

- Like Alice's "when the world starts" event
- Called when program first starts running
- Used to create the initial scene
- Let's call the StickFigure constructor from the **begin method** of our program:

```
public class DrawAStickFigure extends WindowController
{
    public void begin() {
        new StickFigure (canvas);
    }
}
```




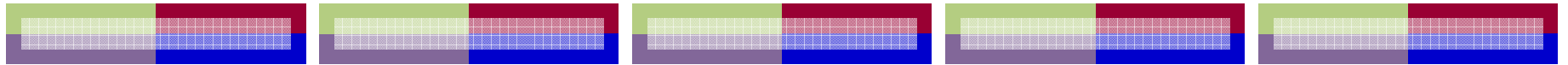


Putting the pieces together

```
public class DrawAStickFigure extends WindowController
{
    public void begin() {
        new StickFigure (canvas);
    }
}

public class StickFigure {
    private static final int HEAD_SIZE = 100;
    ...
    public StickFigure (DrawingCanvas figureCanvas) {
        new FramedOval (HEAD_LEFT, HEAD_TOP,
                        HEAD_SIZE, HEAD_SIZE, figureCanvas);
        ...
    }
}
```





Putting the pieces together

```
public class DrawAStickFigure extends WindowController
{
    public void begin() {
        new StickFigure (canvas);
    }
}
```

1. Program starts

```
public class StickFigure {
    private static final int HEAD_SIZE = 100;
    ...
    public StickFigure (DrawingCanvas figureCanvas) {
        new FramedOval (HEAD_LEFT, HEAD_TOP,
                        HEAD_SIZE, HEAD_SIZE, figureCanvas);
        ...
    }
}
```





Putting the pieces together

```
public class DrawAStickFigure extends WindowController
```

```
{  
    public void begin() {  
        new StickFigure (canvas);  
    }  
}
```

2. Execute 1st statement of
begin method.

Note: canvas is a predefined
variable for all classes that extend
WindowController

```
public class StickFigure {
```

```
    private static final int HEAD_SIZE = 100;
```

```
    ...
```

```
    public StickFigure (DrawingCanvas figureCanvas) {
```

```
        new FramedOval (HEAD_LEFT, HEAD_TOP,
```

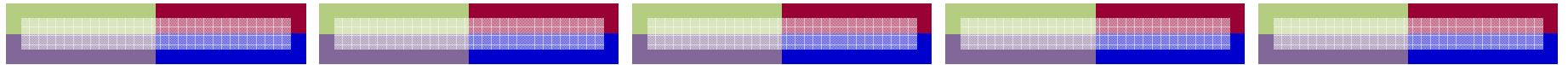
```
                        HEAD_SIZE, HEAD_SIZE, figureCanvas);
```

```
        ...
```

```
    }
```

```
}
```





Putting the pieces together

```
public class DrawAStickFigure extends WindowController
```

```
{  
    public void begin() {  
        new StickFigure (canvas);  
    }  
}
```

3. Call the StickFigure constructor.
The figureCanvas parameter gets
the value of canvas.

```
public class StickFigure {
```

```
    private static final int HEAD_SIZE = 100;
```

```
    ...
```

```
    public StickFigure (DrawingCanvas figureCanvas) {
```

```
        new FramedOval (HEAD_LEFT, HEAD_TOP,
```

```
                        HEAD_SIZE, HEAD_SIZE, figureCanvas);
```

```
        ...
```

```
    }  
}
```

```
}
```





Putting the pieces together

```
public class DrawAStickFigure extends WindowController
```

```
{
```

```
    public void begin() {
```

```
        new StickFigure (canvas);
```

```
    }
```

```
}
```

```
public class StickFigure {
```

```
    private static final int HEAD_SIZE = 100;
```

```
    ...
```

```
    public StickFigure (DrawingCanvas figureCanvas) {
```

```
        new FramedOval (HEAD_LEFT, HEAD_TOP,
```

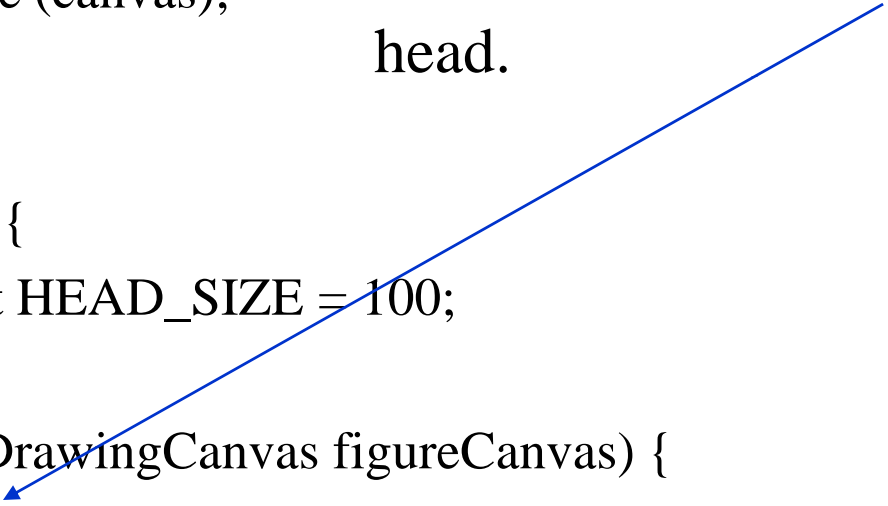
```
                        HEAD_SIZE, HEAD_SIZE, figureCanvas);
```

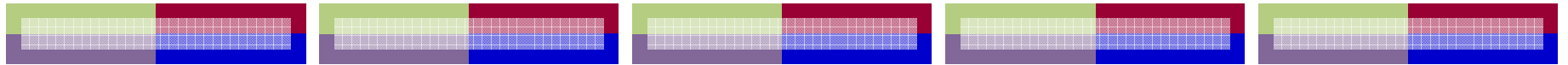
```
        ...
```

```
    }
```

```
}
```

4. Draw the framed oval for the head.





Putting the pieces together

```
public class DrawAStickFigure extends WindowController
```

```
{
```

```
    public void begin() {
```

```
        new StickFigure (canvas);
```

```
    }
```

```
}
```

5. Draw the other parts -
statements not shown here.

```
public class StickFigure {
```

```
    private static final int HEAD_SIZE = 100;
```

```
    ...
```

```
    public StickFigure (DrawingCanvas figureCanvas) {
```

```
        new FramedOval (HEAD_LEFT, HEAD_TOP,
```

```
                        HEAD_SIZE, HEAD_SIZE, figureCanvas);
```

```
        ...
```

```
    }
```

```
}
```





Putting the pieces together

```
public class DrawAStickFigure extends WindowController
```

```
{  
    public void begin() {  
        new StickFigure (canvas);  
    }  
}
```

```
public class StickFigure {
```

```
    private static final int HEAD_SIZE = 100;
```

```
    ...
```

```
    public StickFigure (DrawingCanvas figureCanvas) {
```

```
        new FramedOval (HEAD_LEFT, HEAD_TOP,
```

```
                        HEAD_SIZE, HEAD_SIZE, figureCanvas);
```

```
        ...
```

```
    }
```

```
}
```



6. Return to the begin method. There is nothing more to do in the begin method. At this point the stick figure appears on the screen.

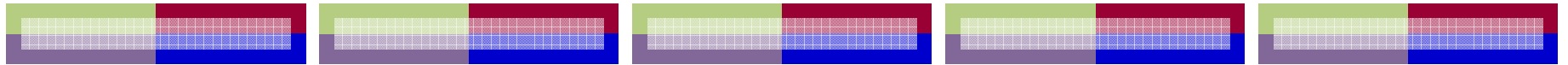


Reacting to the Mouse

- Let's have the stick figure say hi when the user presses the mouse button down
- Mouse handling events go in the class that extends WindowController

```
public class DrawAStickFigure extends WindowController {  
    public void begin () {  
        ...  
    }  
    public void onMousePress (Location point) {  
        new Text ("Hi!", 250, 150, canvas);  
    }  
}
```





Mouse Event Handling Methods

- public void onMouseClick (Location point)
- Called when mouse pressed and released without moving it
- Following methods have similar declarations

public void onMouseClick (Location point)

onMousePress

onMouseRelease

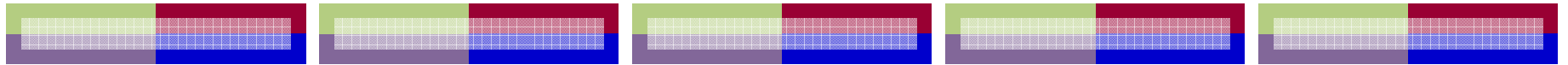
onMouseEnter

onMouseExit

onMouseMove

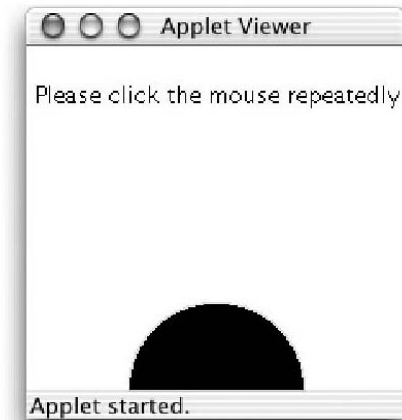
onMouseDown





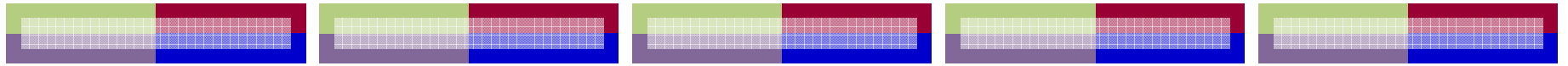
A Sample Program

A rudimentary solar simulation:



We want a click of the mouse in the window to make the 'sun' rise





Moving an Oval

- *Mutator* methods: change an object that's been created

```
move ( 0, -5 ); // parameters indicate distances to move in  
                // horizontal (ie. x) and vertical (ie. y) directions
```

- But how does the computer know what to move? Must give the object to be moved a name!





Some Building Blocks

```
private FilledOval sun; // instance variable declaration
```

```
// 'sun' is the name
```

- // private specifies *scope*
 - // FilledOval is the *type* of object that sun will be
 - // no value or object is yet specified with 'sun'
- ```
// container is set aside and associated with name
```

```
sun – new FilledOval(50, 150, 100, 100, canvas);
```

- //assignment statement: 1) evaluate statement on right
- // 2) set the contents of the object on the left to that
- // value

```
sun.move(0, -5); // send the move message to the sun object
```






# Giving Names

- Declare all instance variables

```
private FilledOval sun;
```

- Appropriate Names

- Start with letters (lowercase is best)
  - Case sensitive
  - Letters, digits, underscores
  - Not a word already in Java
- 



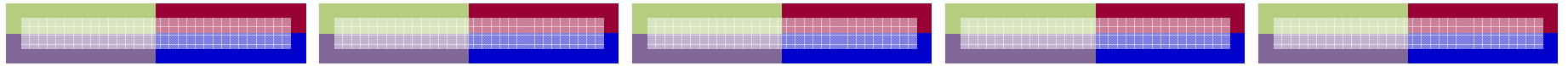
# Comments for Clarity

- In our line

```
private FilledOval sun; //sun is the name
```

- “sun is the name” is a comment



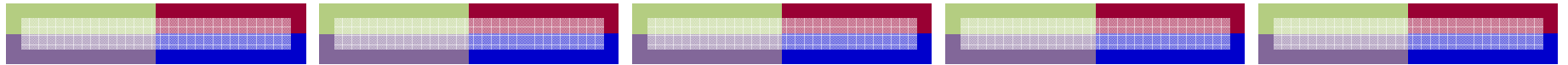


## Mutator Methods in Context

- We want to move the sun when the user clicks the mouse:

```
public void onMouseClick(Location point){
 sun.move(0, -5);
}
```





## More Mutator Methods

```
private Text instructions; //Display of instructions
```

- 
- 
- 

```
instructions = new Text (...);
```

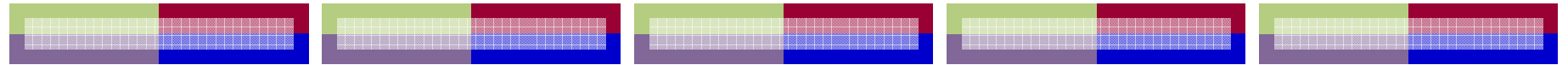
- 
- 
- 

```
instructions.hide();
```

- 
- 
- 

```
instructions.show();
```





```
public class RisingSun extends WindowController {
```

```
 private FilledOval sun; // Circle that represents the sun
```

```
 private Text instructions; //Display of instructions
```

```
 public void begin() { //Place the sun and brief instructions on screen
```

```
 sun = new FilledOval(50, 150, 100, 100, canvas);
```

```
 instructions = new Text("Please click the mouse ",
 20, 20, canvas);
```

```
 }
```

```
 //Move the sun up each click
```

```
 public void onMouseClick(Location point) {
```

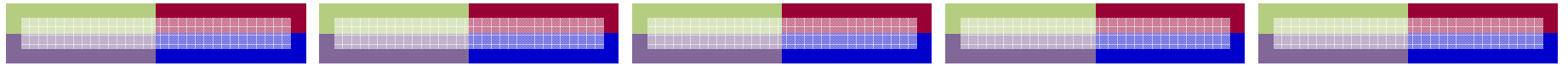
```
 sun.move(0, -5);
```

```
 instructions.hide();
```

```
 }
```

```
}
```

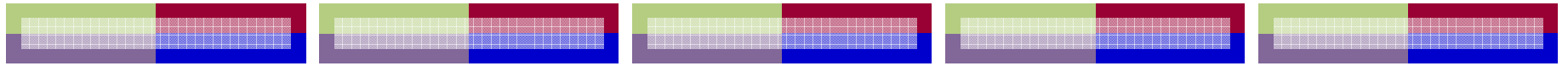




## More Classes

- Classes so far: Line, FilledOval, FramedOval, Text, FilledRect, FramedRect,
- Can also have nongraphical classes!
  - Color
  - Location





# Colors

```
private Color purple;
```

- 
- 
- 

```
purple = new Color (255, 0, 255);
```

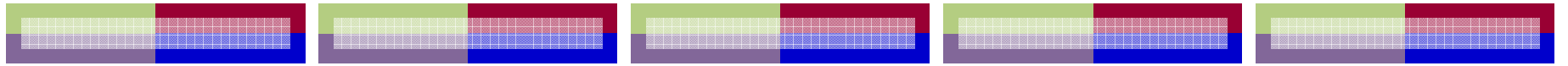
- 
- 
- 

```
sun.setColor(Color.YELLOW); //makes sun yellow
```

- 
- 
- 

```
instructions.setColor(purple); //makes instr purple
```





## Colors in Java

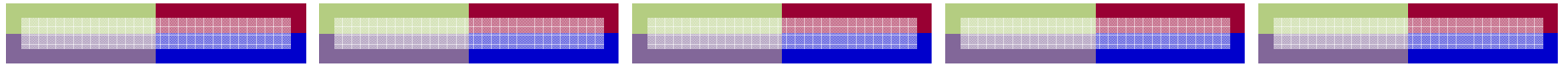
BLACK, BLUE, CYAN, DARK\_GRAY,

GRAY, GREEN, LIGHT\_GRAY, MAGENTA,

ORANGE, PINK, RED, WHITE,

YELLOW





# Locations

```
private Location initialPosition;
```

- 
- 
- 

```
initialPosition = new Location(50, 150);
```

- 
- 
- 

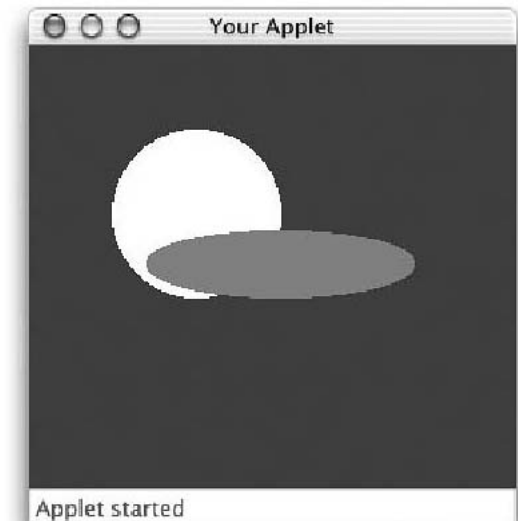
```
sun.moveTo (initialPosition);
```

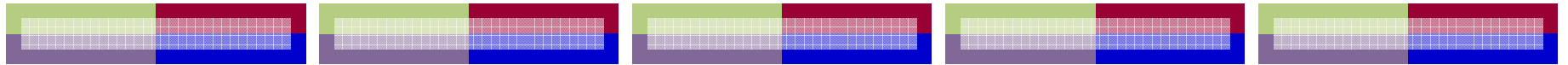




# Layering the Canvas

- Create two overlapping FilledOvals. Which one's on top?
  - Answer: The most recently constructed
- How do we change the order?
  - Answer: Mutator Methods
    - `sendBackward()`
    - `sendForward()`
    - `sendToBack()`
    - `sendToFront()`





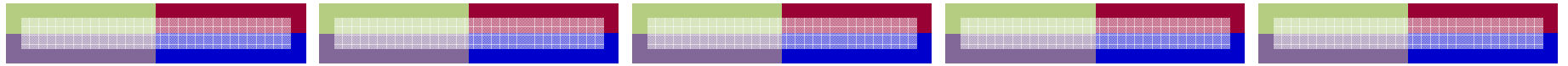
## Using a Mouse Point

- Recall `onMousePress( Location point );`
  - We can use the Location!
- Consider:

```
public void onMousePress (Location point) {
 new Text ("Pressed", point, canvas);
}
```

Displays "Pressed" wherever the mouse is clicked





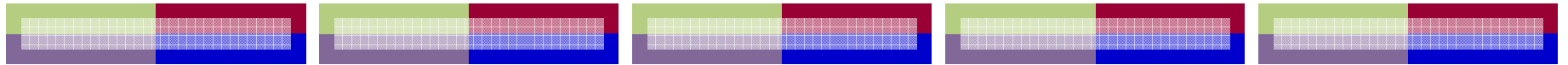
## Using Multiple Points

```
private Location firstPoint;

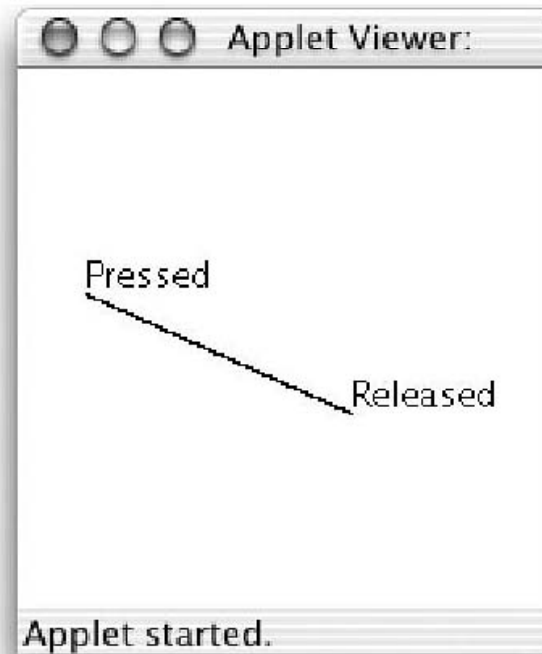
public void onMousePress(Location pressPt){
 new Text("Pressed", pressPt, canvas);
 firstPoint = pressPt;
}

public void onMouseRelease (Location releasePt){
 new Text("Released", releasePt, canvas);
 new Line(firstPoint, releasePt, canvas);
}
```



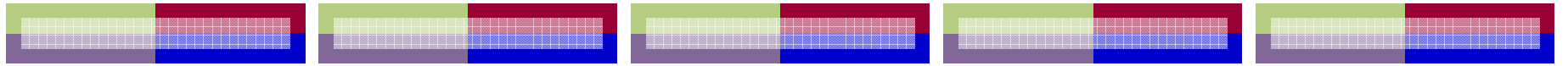


# Multiple Point Picture



- Can play connect the dots!





# Review

- Graphical and nongraphical objects
- Names and mutator methods
- Layering the canvas

