Week 7 Discussion

Wednesday, 11/13/19

Reminders

Midterm review session tomorrow (8pm Solis Hall)

Midterm 2 next Monday, November 18

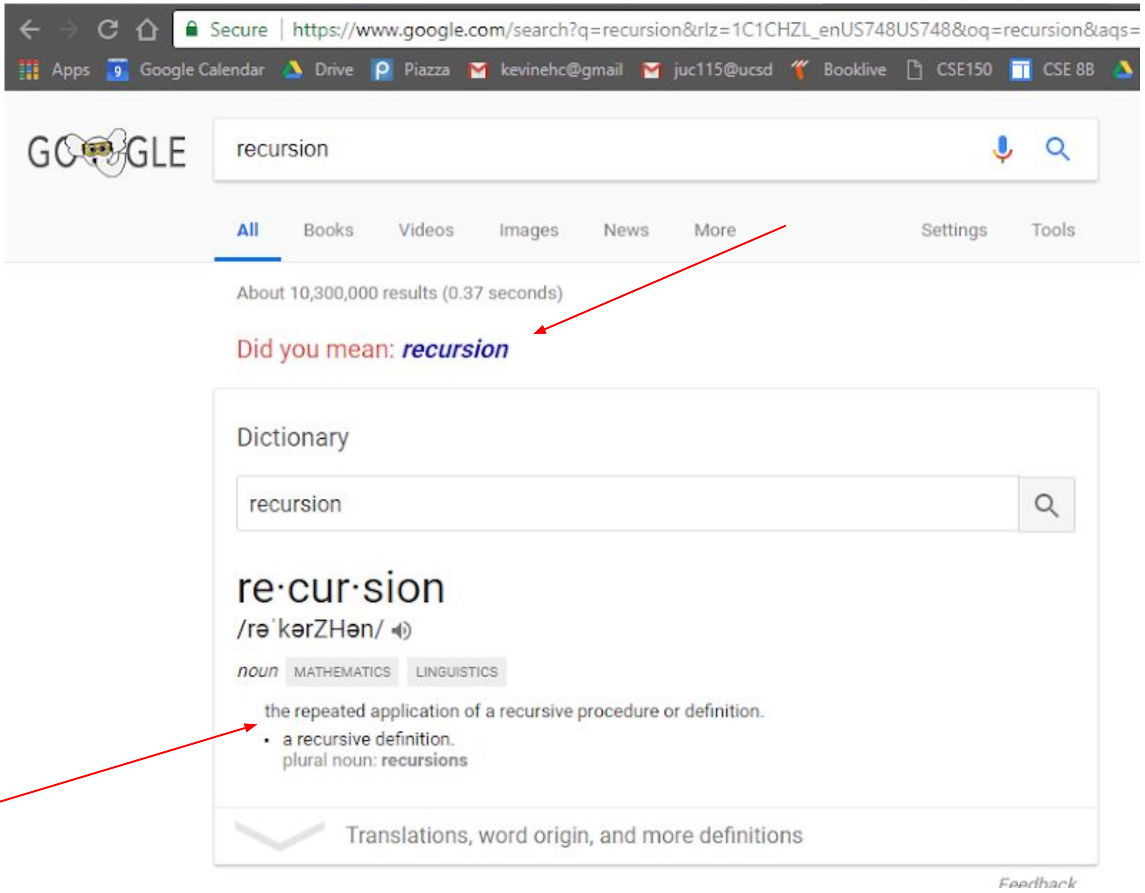
PSA5 Submission due **Wednesday, November 20 11:59pm**

Today’s agenda

- Overview of the PSA
    - Part 1 - Recursion exercises
    - Part 2 - Code review
    - Part 3 - Shapes

PSA5, Part 1
Recursion exercises

Recursion Visualized



Secure | https://www.google.com/search?q=recursion&rlz=1C1CHZL_enUS748US748&oq=recursion&aqs=

Apps Google Calendar Drive Piazza kevinehc@gmail juc115@ucsd Booklive CSE150 CSE 8B

recursion

All Books Videos Images News More Settings Tools

About 10,300,000 results (0.37 seconds)

Did you mean: ***recursion***

Dictionary

recursion

re·cur·sion

/rəˈkərZHən/

noun MATHEMATICS LINGUISTICS

the repeated application of a recursive procedure or definition.

- a recursive definition.
  plural noun: **recursions**

Translations, word origin, and more definitions

Feedback

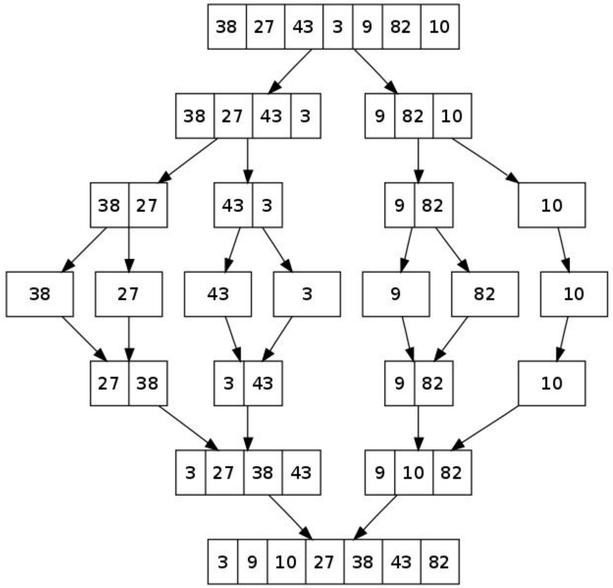
… well played Google

Recursion

- Recursion is when a method calls itself - often with altered arguments
- We will use it to make complex shapes like this:

Why Recursion?

- We can use this to "divide and conquer" complex problems by breaking them down into numerous simpler parts



Base case

- A recursive method must include a "base case", so that it knows when to stop calling itself and end the recursion.
- When recursively calling a method itself, it must be called with different parameters. It should eventually reach the base case

- FriEND
- BoyfriEND
- GirlfriEND
- BestfriEND
- Recursion

Only Recursion has no END.

Base case

Which of these is the base case?

```
public static void rec(int i) {
  if (i < 2) {
    return;          // A
  }
  rec(i-1);          // B
}
```

Base case

Which of these is the base case?

```
public static void rec(int i) {
  if (i < 2) {
    return;           // A
  }
  rec(i-1);           // B
}
```

Practice

What will the following code print?

```java
public class Program {
  public static void main(String[] args) {
    recurse(4, 100);
  }
  public static void recurse(int i, int m) {
    if (i < m) {
      System.out.print(i + ", ");
      recurse(i * 2, m);
    }
  }
}
```

A:
4, 8, 12, 16, …, 100,

B:
4, 8, 16, 32, 64,

C:
100, 96, 92, 88, …, 4,

D:
64, 32, 16, 8, 4,

Practice

What will the following code print?

```
public class Program {
  public static void main(String[] args) {
    recurse(4, 100);
  }
  public static void recurse(int i, int m) {
    if (i < m) {
      System.out.print(i + ", ");
      recurse(i * 2, m);
    }
  }
}
```

A:
4, 8, 12, 16, …, 100,

B:
4, 8, 16, 32, 64,

C:
100, 96, 92, 88, …, 4,

D:
64, 32, 16, 8, 4,

Bonus Question: What is the base case in this code?

Practice

What will the following code print?

```
public class Program {
  public static void main(String[] args) {
    recurse(4, 100);
  }
  public static void recurse(int i, int m) {
    if (i < m) {
      System.out.print(i + ", ");
      recurse(i * 2, m);
    }
  }
}
```

A:
4, 8, 12, 16, …, 100,

B:
4, 8, 16, 32, 64,

C:
100, 96, 92, 88, …, 4,

D:
64, 32, 16, 8, 4,

Bonus Question: The base case is when i >= m, in which case the code does not call the `recurse` method again.

Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

main   print(___)

Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

factorial	Num = 5
	return ___ * 5

main	print(___)

Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

factorial	Num = 4
	return ___ * 4

factorial	Num = 5
	return ___ * 5

main	print(___)

Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

factorial	Num = 3
	return ___ * 3

factorial	Num = 4
	return ___ * 4

factorial	Num = 5
	return ___ * 5

main	print(___)

Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

factorial	Num = 2
	return ___ * 2

factorial	Num = 3
	return ___ * 3

factorial	Num = 4
	return ___ * 4

factorial	Num = 5
	return ___ * 5

main	print(___)

Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

factorial	Num = 1
	return 1

factorial	Num = 2
	return ___ * 2

factorial	Num = 3
	return ___ * 3

factorial	Num = 4
	return ___ * 4

factorial	Num = 5
	return ___ * 5

main	print(___)

Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

factorial	Num = 1
	return 1

factorial	Num = 2
	return ___ * 2

factorial	Num = 3
	return ___ * 3

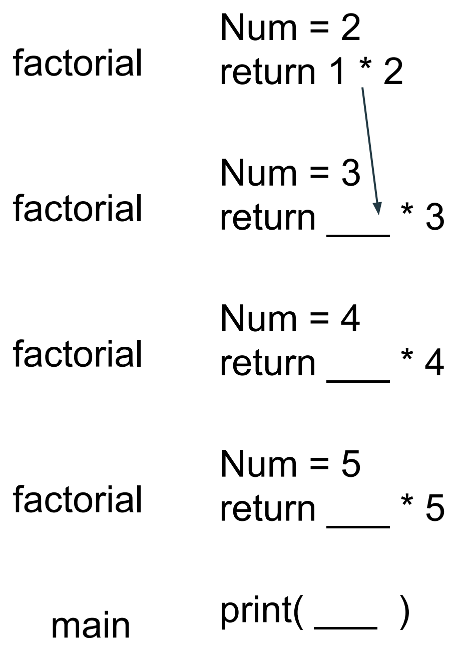
factorial	Num = 4
	return ___ * 4

factorial	Num = 5
	return ___ * 5

main	print(___)

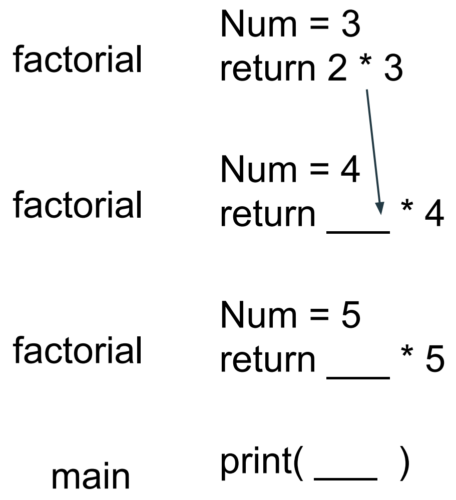
Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```



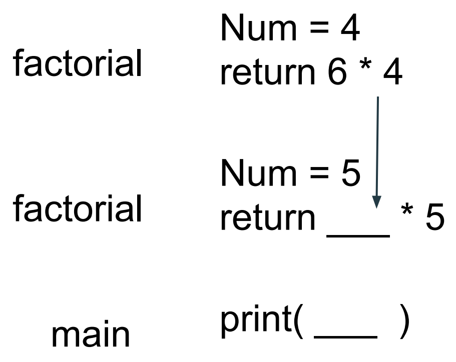
Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```



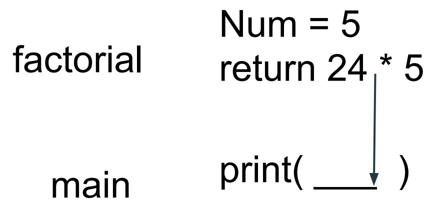
Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```



Recursion Tracing / Stack Frames

```java
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

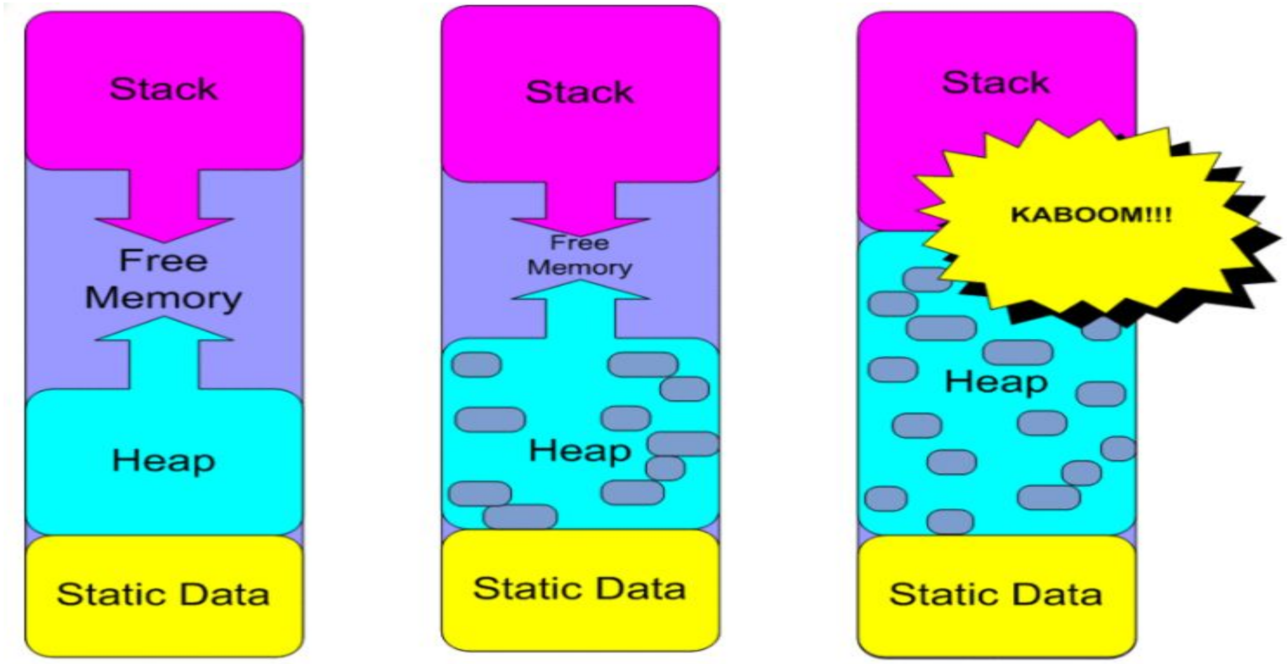


Recursion Tracing / Stack Frames

```
public class Factorial {
  public static void main(String[] args) {
    System.out.println(factorial(5));
  }
  public static int factorial(int num){
    if (num == 1 || num == 0) {
      return 1;
    }
    return factorial(num-1) * num;
  }
}
```

main   print(120)

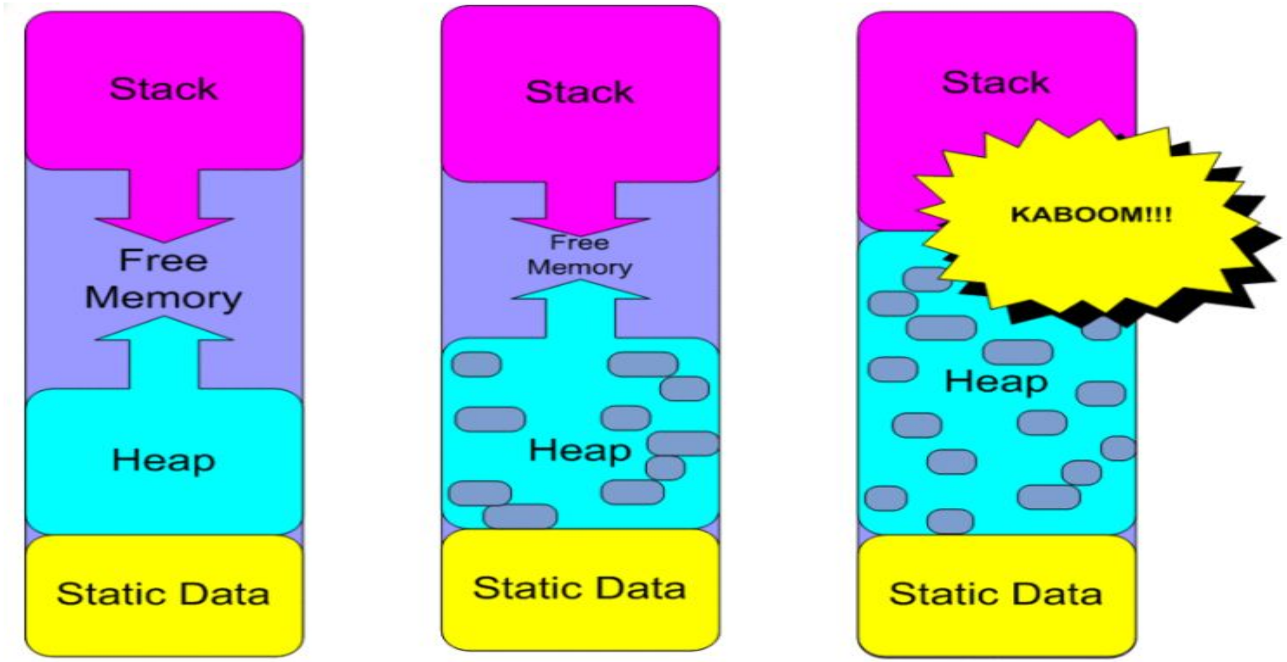
Caveat aka Pitfalls of Recursion



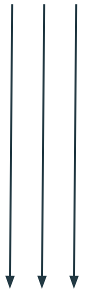
Stack Overflow:

Stack growing into Heap

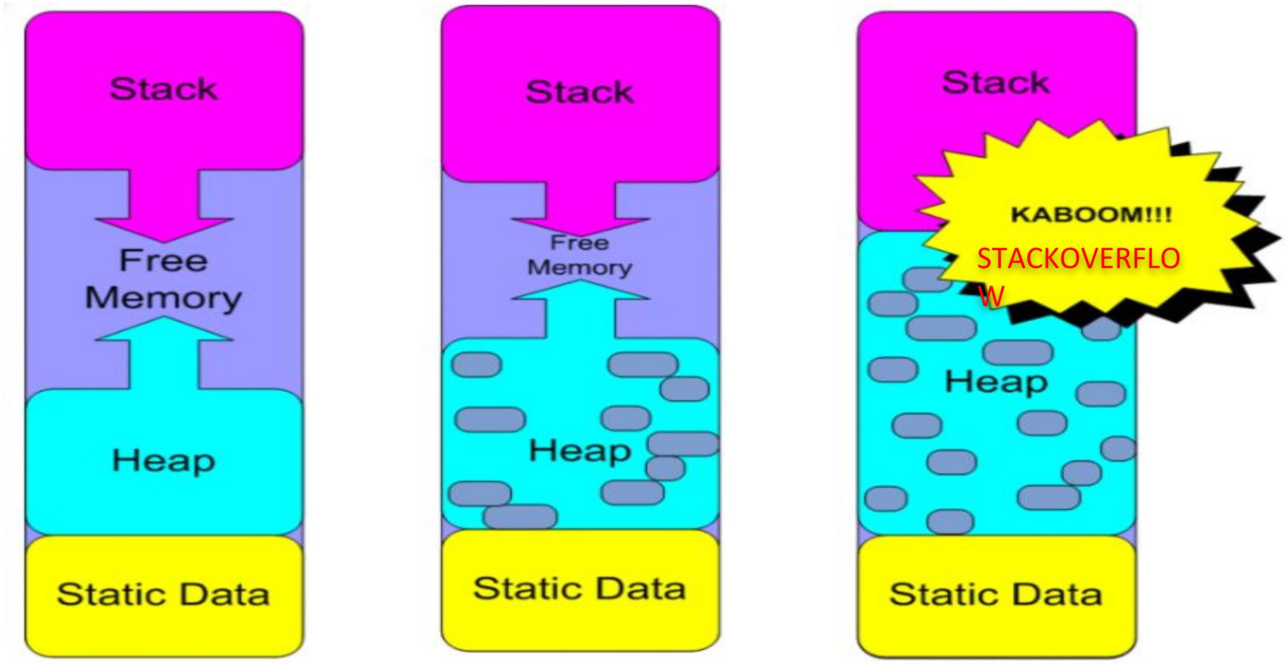
Caveat aka Pitfalls of Recursion



Stack Overflow:
Stack growing into Heap



Caveat aka Pitfalls of Recursion



Stack Overflow:
Stack growing into Heap

PSA5, Part 2
Code review

Code Review

- In industry, code reviews are performed to make sure other people's code is up to standards
  - It also helps you improve your own code
- Assigned two files (find assigned files linked in write-up) but just need to review one
- Three parts:
  - What’s good
  - Logic and functionality errors
  - Miscellaneous comments (other comments)

PSA5, Part 3
Shapes

Intro to JavaFX

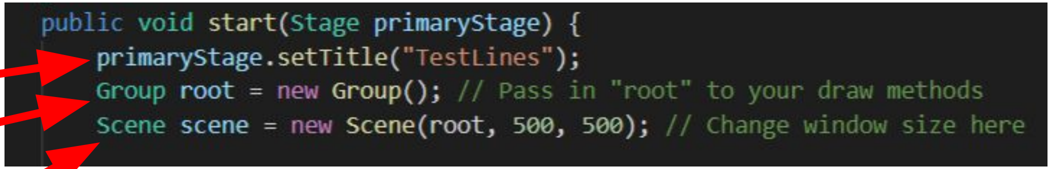
- JavaFX is a GUI library full of fun things to play around with, such as shapes, animations, and text
- We will be using JavaFX to create shapes in the `draw()` methods

General process:

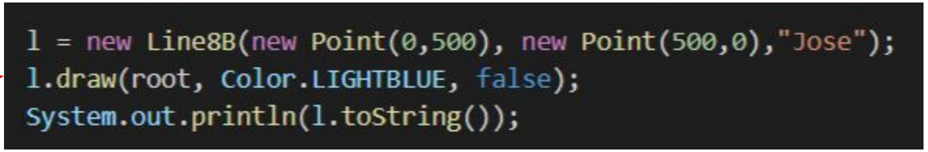
1. Given a stage
2. Create a group
3. Pass in the group to create a scene
4. Add children to the group
5. Set the scene on the stage
6. Show the stage!

A look at TestLines.java

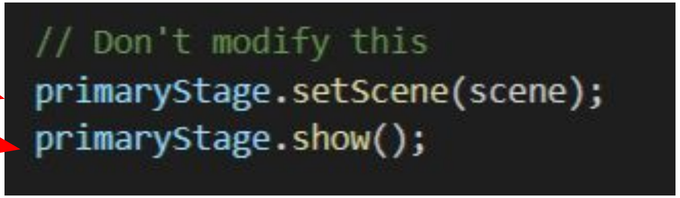
1. Given a stage
2. Create a group
3. Pass in the group to create a scene
4. Add children to the group
5. Set the scene on the stage
6. Show the stage!



```java
public void start(Stage primaryStage) {
    primaryStage.setTitle("TestLines");
    Group root = new Group(); // Pass in "root" to your draw methods
    Scene scene = new Scene(root, 500, 500); // Change window size here
```

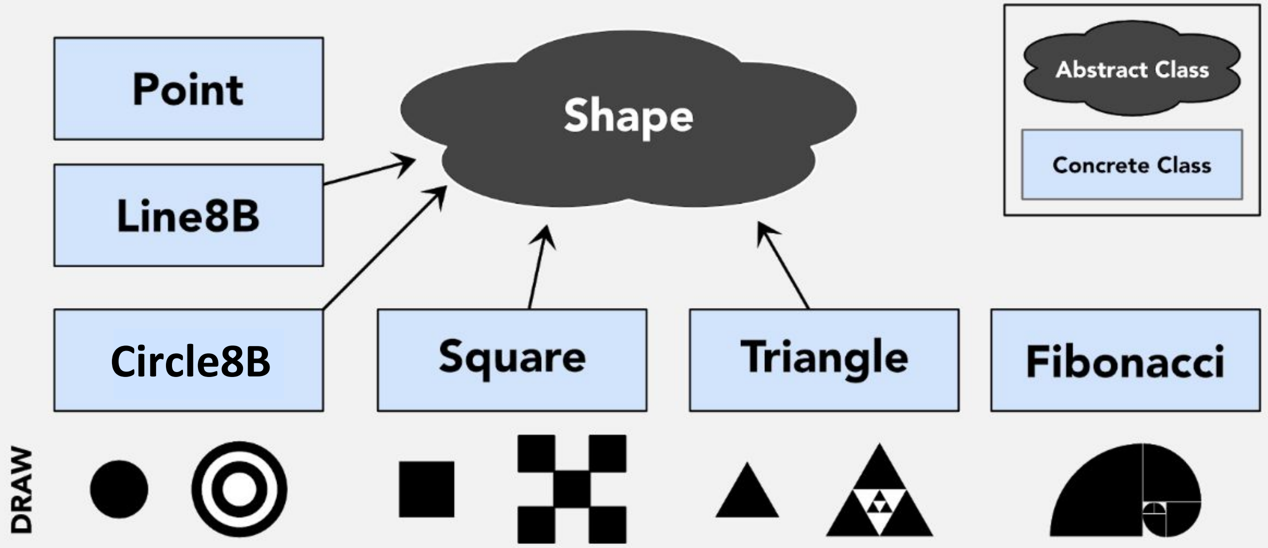


```java
l = new Line8B(new Point(0,500), new Point(500,0),"Jose");
l.draw(root, Color.LIGHTBLUE, false);
System.out.println(l.toString());
```

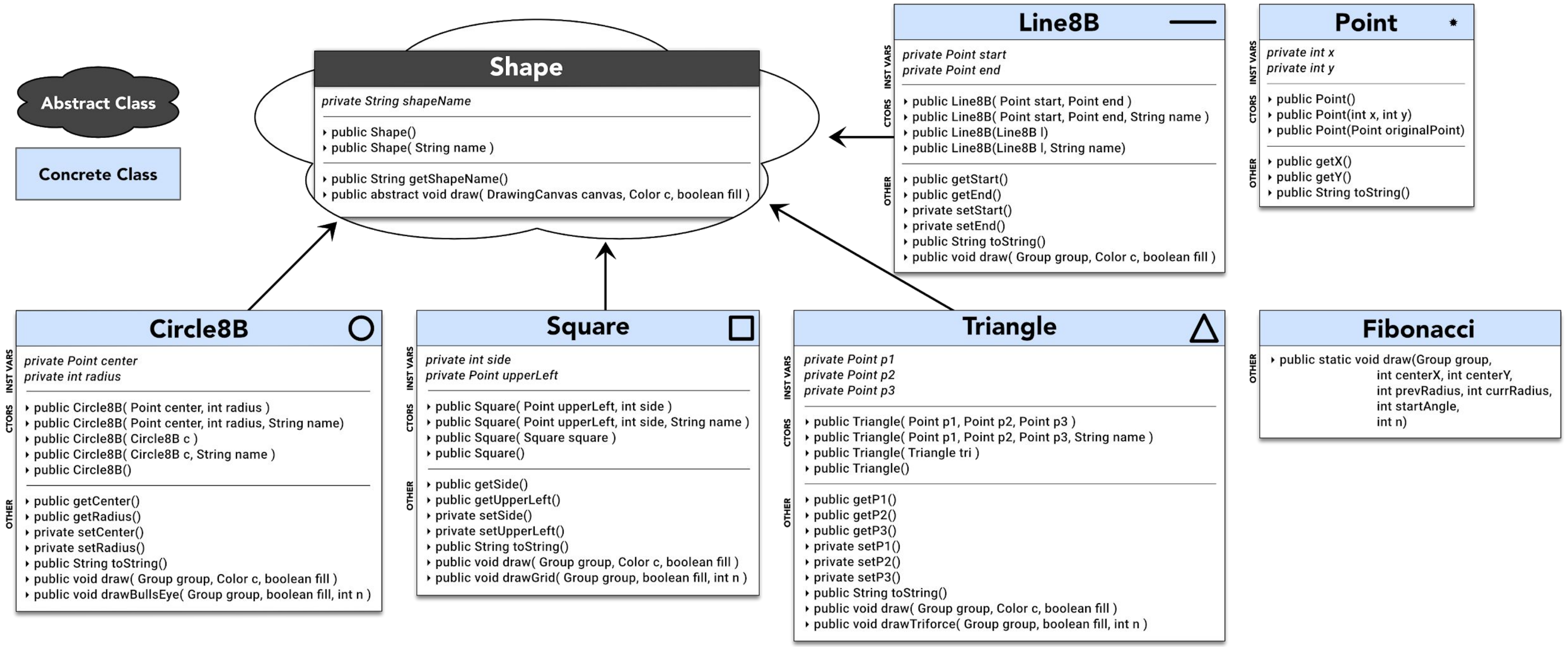


```java
// Don't modify this
primaryStage.setScene(scene);
primaryStage.show();
```

Introducing the Components of Shapes



Point.java and Line8B.java are provided files, you should NOT edit or change them.



Shape
- *private String shapeName*
- public Shape()
- public Shape(String name)
- public String getShapeName()
- public abstract void draw(DrawingCanvas canvas, Color c, boolean fill)

Line8B
- *private Point start*
- *private Point end*
- public Line8B(Point start, Point end)
- public Line8B(Point start, Point end, String name)
- public Line8B(Line8B l)
- public Line8B(Line8B l, String name)
- public getStart()
- public getEnd()
- private setStart()
- private setEnd()
- public String toString()
- public void draw(Group group, Color c, boolean fill)

Point
- *private int x*
- *private int y*
- public Point()
- public Point(int x, int y)
- public Point(Point originalPoint)
- public getX()
- public getY()
- public String toString()

Circle8B
- *private Point center*
- *private int radius*
- public Circle8B(Point center, int radius)
- public Circle8B(Point center, int radius, String name)
- public Circle8B(Circle8B c)
- public Circle8B(Circle8B c, String name)
- public Circle8B()
- public getCenter()
- public getRadius()
- private setCenter()
- private setRadius()
- public String toString()
- public void draw(Group group, Color c, boolean fill)
- public void drawBullsEye(Group group, boolean fill, int n)

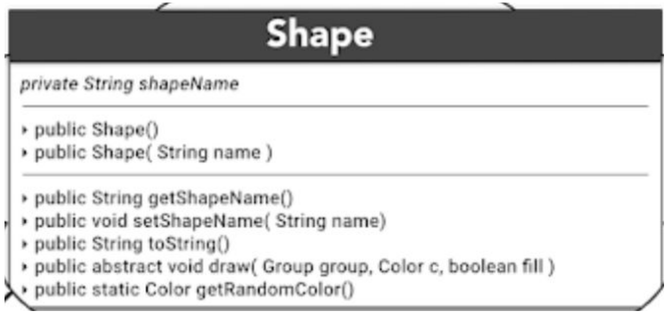
Square
- *private int side*
- *private Point upperLeft*
- public Square(Point upperLeft, int side)
- public Square(Point upperLeft, int side, String name)
- public Square(Square square)
- public Square()
- public getSide()
- public getUpperLeft()
- private setSide()
- private setUpperLeft()
- public String toString()
- public void draw(Group group, Color c, boolean fill)
- public void drawGrid(Group group, boolean fill, int n)

Triangle
- *private Point p1*
- *private Point p2*
- *private Point p3*
- public Triangle(Point p1, Point p2, Point p3)
- public Triangle(Point p1, Point p2, Point p3, String name)
- public Triangle(Triangle tri)
- public Triangle()
- public getP1()
- public getP2()
- public getP3()
- private setP1()
- private setP2()
- private setP3()
- public String toString()
- public void draw(Group group, Color c, boolean fill)
- public void drawTriforce(Group group, boolean fill, int n)

Fibonacci
- public static void draw(Group group, int centerX, int centerY, int prevRadius, int currRadius, int startAngle, int n)

Can be found on the writeup!

Shape



```
Shape
private String shapeName
‣ public Shape()
‣ public Shape( String name )
‣ public String getShapeName()
‣ public void setShapeName( String name)
‣ public String toString()
‣ public abstract void draw( Group group, Color c, boolean fill )
‣ public static Color getRandomColor()
```

- An abstract class
- private instance variable called `shapeName`
- Two constructors: one no-arg, one that takes a `String`
- public getter method `getShapeName()`
- public setter method `setShapeName()`
- public abstract method named `draw()` that takes in a `Group`, `Color`, and `boolean`
    - `boolean` determines whether or not the shape is filled
- public `toString()` method that prints out the shape name
- public method `getRandomColor()` implementation is in the write-up!

Testing Shape.java on Gradescope

- Use Gradescope to make sure your `Shape.java` is working!
- **Make sure you are using the correct modifiers and names**
- You should be able to compile `Line8B.java` after implementing Shape
- If there is an error, **DO NOT** modify `Line8B.java`
    - Debug your `Shape.java` instead

Meet the Shapes

(these are given to you)

`Point`

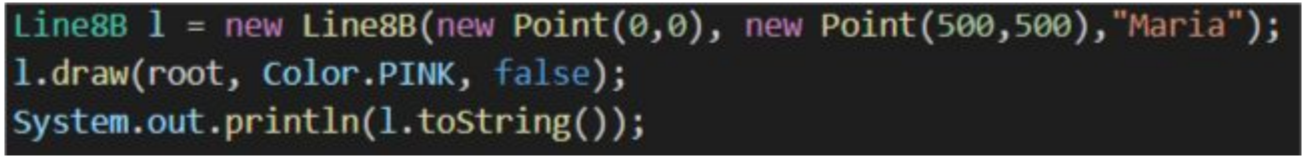
- Has two private int instance variables, `x` and `y`
- Constructors:
  - First constructor takes in a `x` and `y` coordinate as a pair of ints
  - The second constructor takes in no arguments and creates a point at (0,0)
  - The third constructor is a copy constructor that takes in a `Point`
- Getters and setters
- `toString()` method which gets called when the `Point` object is used as an argument inside a print statement
  - `System.out.println(new Point())` is equivalent to
  - `System.out.println(new Point().toString())`

Line8B `extends` Shape

- Two `Point` objects are used to define a single line
- Constructors:
  - Takes in two `Point` objects without a name. Default name is "NoName".
  - Takes in two `Point` objects with a `String` as a name
  - Deep copying of a `Line8B` object. One with a name input and one without
- Getters and setters
- `toString()` method that prints a description of the line

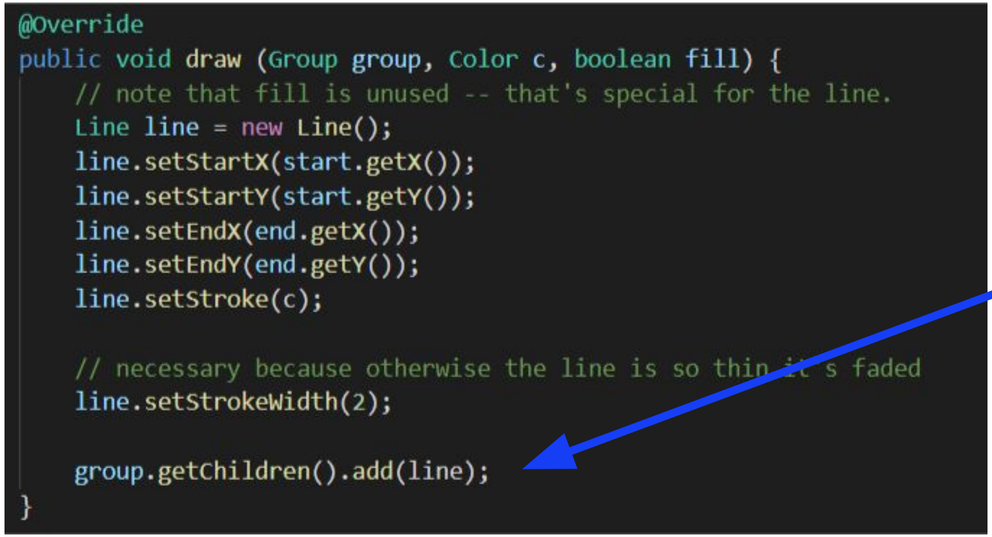
A look at Line8B.java

TestLines.java:



```java
Line8B l = new Line8B(new Point(0,0), new Point(500,500),"Maria");
l.draw(root, Color.PINK, false);
System.out.println(l.toString());
```

Line8B.java:



```java
@Override
public void draw (Group group, Color c, boolean fill) {
    // note that fill is unused -- that's special for the line.
    Line line = new Line();
    line.setStartX(start.getX());
    line.setStartY(start.getY());
    line.setEndX(end.getX());
    line.setEndY(end.getY());
    line.setStroke(c);

    // necessary because otherwise the line is so thin it's faded
    line.setStrokeWidth(2);

    group.getChildren().add(line);
}
```

Why do we have to do this?

Meet the Shapes

(that you need to implement)

Shapes Overview

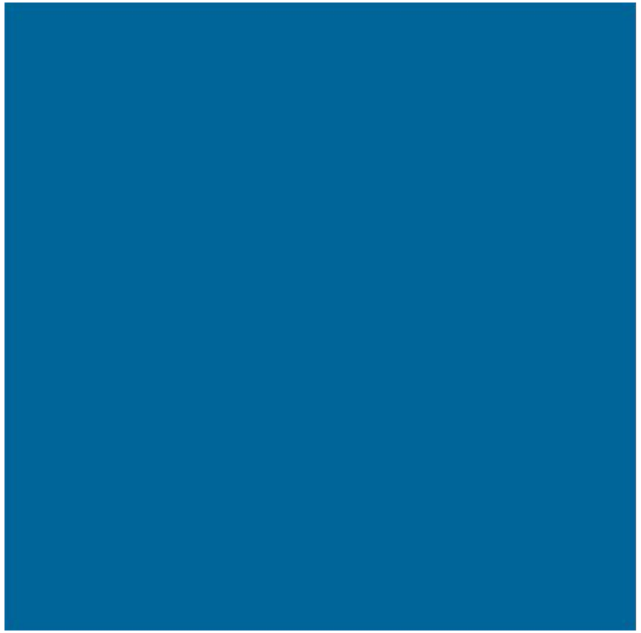
- Three different shapes that must be implemented
  - Square.java
  - Triangle.java
  - Circle8B.java
- Extends the `Shape` abstract class
- `draw()` method to display the normal shape on canvas
- Special draw method that uses recursion to display a special pattern formed by the specific shape

`Circle8B extends Shape`

- Defined by
  - `Point`: center of circle
  - `int`: length of radius
- public getter methods to get the center point and radius
- private setter methods for them
- public `toString()` method
- Implement the `draw` method defined in abstract class `Shape`
  - Use the JavaFX library!
  - Remember to add the `Circle8B` object to the `group`’s children in draw

Square `extends` Shape

- Defined by
  - `Point`: coordinates of <u>upper-left</u> corner
  - `int`: <u>length</u> of the sides
- public getter methods
- private setter methods
- public `toString()` method
- Implement the `draw` method defined in abstract class `Shape`



Triangle `extends` Shape

- Given 3 points, draw 3 lines connecting the points.
- To draw the Triangle, which JavaFX shape can we use?
  - Hint: what's the most generic shape type that Triangle belongs to?
- Fill or make the triangle an outline based off the `fill` boolean

Meet the Fancy Draw Methods

Fancy draw methods

- `Circle`, `Square`, and `Triangle` each have a unique method that will draw an artistic pattern with recursion!
- Each method is different and each shape only has access to one of them.

`drawBullsEye(Group group, boolean fill, int n)`

- We start by drawing the circle normally
  - Then, we recurse
- Recursively draw the circle over and over again, reducing the radius each time by 13 until `n` reaches 0.
  - Hint: Think about how to change the radius for each recursive call

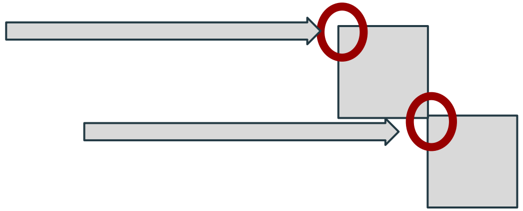
`drawTriforce(Group group, boolean fill, int n)`

- Similar logic to `Circle`
- Draw another `Triangle` inside of the original `Triangle` - and then draw a `Triangle` inside of that……
- Remember: `Triangle` has instance variables `p1`, `p2`, and `p3` (which are `Point`s).

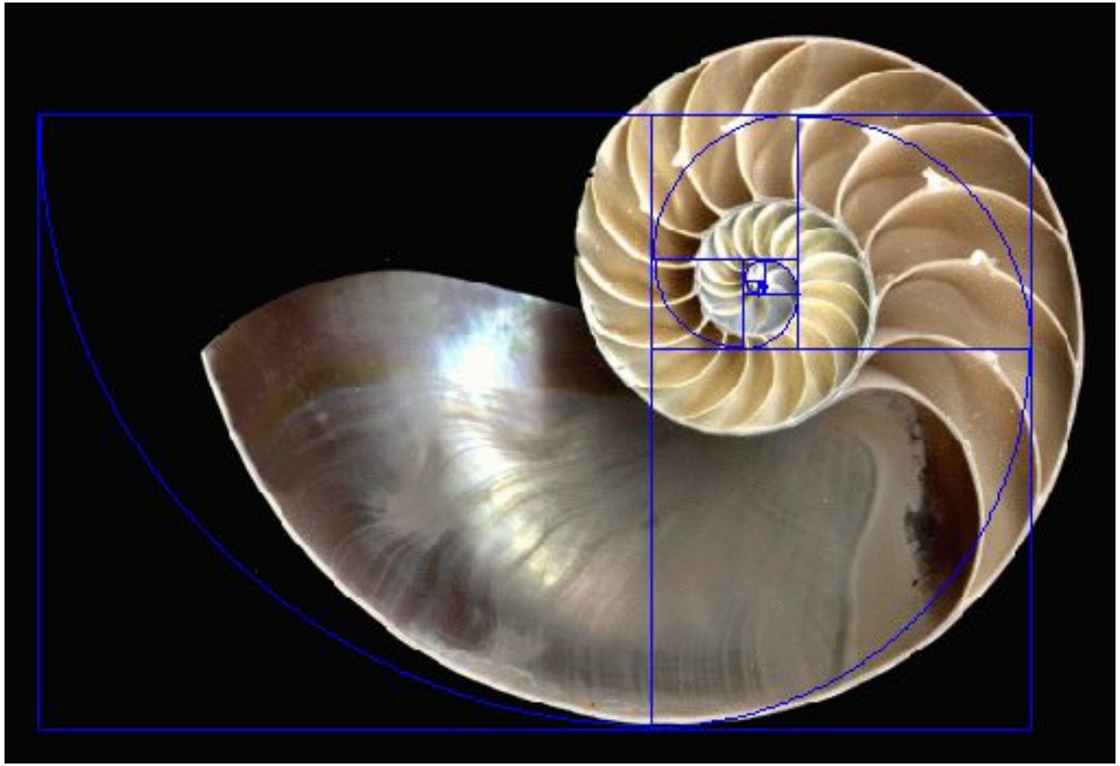
`drawGrid(Group group, boolean fill, int n)`

- Similar logic to `Circle`
- This time, we draw four `Squares`
  - One on each corner
  - Hint: Think about which `Square` constructor you want to use given you have access to instance vars `upperLeft` and `side`

- Ex: How to calculate this point?
- Given this `upperLeft` coordinate and `side`?



Fibonacci



What is a fibonacci sequence?

- It is a sequence where each element is the sum of the previous 2 elements
- 0, 1, 1, 2, 3, 5, 8, 13, 21, …

What is a fibonacci sequence?

- It is a sequence where each element is the sum of the previous 2 elements
- $\underline{0, 1}$, 1, 2, 3, 5, 8, 13, 21, …
- 0 + 1 = 1

What is a fibonacci sequence?

- It is a sequence where each element is the sum of the previous 2 elements
- 0, $\underline{1, 1}$, 2, 3, 5, 8, 13, 21, …
- 1 + 1 = 2

What is a fibonacci sequence?

- It is a sequence where each element is the sum of the previous 2 elements
- 0, 1, $\underline{1, 2}$, 3, 5, 8, 13, 21, …
- 1 + 2 = 3

What is a fibonacci sequence?

- It is a sequence where each element is the sum of the previous 2 elements
- 0, 1, 1, <u>2, 3</u>, 5, 8, 13, 21, …
- 2 + 3 = 5

What is a fibonacci sequence?

- It is a sequence where each element is the sum of the previous 2 elements
- 0, 1, 1, 2, $\underline{3, 5}$, 8, 13, 21, …
- 3 + 5 = 8

What is a fibonacci sequence?

- It is a sequence where each element is the sum of the previous 2 elements
- 0, 1, 1, 2, 3, $\underline{5, 8}$, 13, 21, …
- 5 + 8 = 13

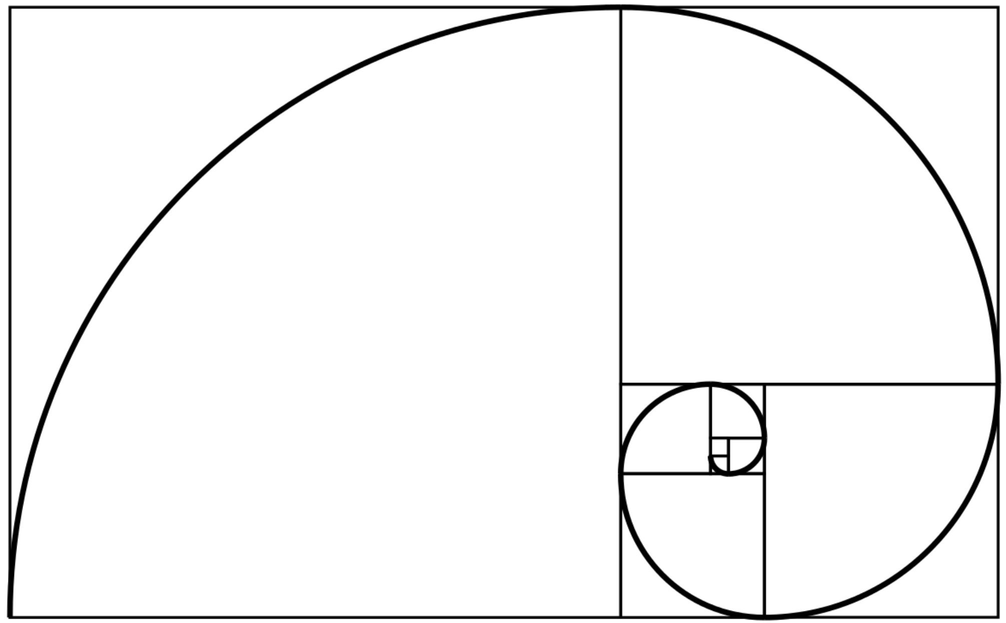
What is a fibonacci sequence?

- It is a sequence where each element is the sum of the previous 2 elements
- 0, 1, 1, 2, 3, 5, <u>8, 13</u>, 21, …
- 8 + 13 = 21

Fibonacci

- Instance Variable:
  - `static final int arcLength = 90; // every arc is a quarter circle`
- Method:
  - `public static void draw(Group group, int centerX, int centerY, int prevRadius, int currRadius, int startAngle, int n)`

What you have to draw



Example:

`TestGoldenRatio.java`

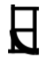
What you have to draw

- 1, 1, 2, 3, 5, 8, 13, 21, ... (we're omitting 0 for the purposes of this assignment)

Length = 1

What you have to draw

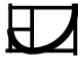
- 1, 1, 2, 3, 5, 8, 13, 21, …



Length of new
Square = 1

What you have to draw

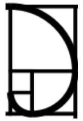
- 1, 1, 2, 3, 5, 8, 13, 21, …



Length of new
Square = 2

What you have to draw

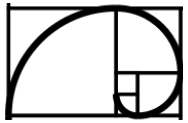
- 1, 1, 2, 3, 5, 8, 13, 21, …



Length of new Square = 3

What you have to draw

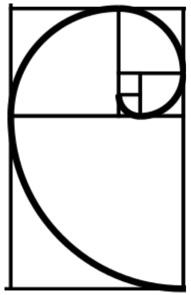
- 1, 1, 2, 3, 5, 8, 13, 21, …



Length of new
Square = 5

What you have to draw

- 1, 1, 2, 3, 5, 8, 13, 21, …

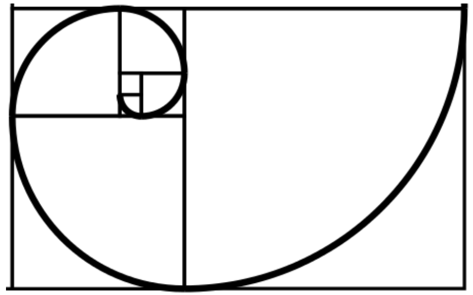


Length of new Square = 8

What you have to draw

- 1, 1, 2, 3, 5, 8, 13, 21, …

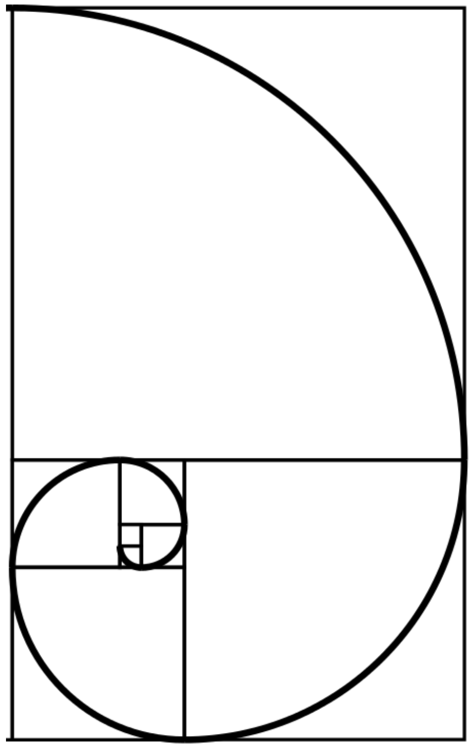
Length of new
Square = 13



What you have to draw

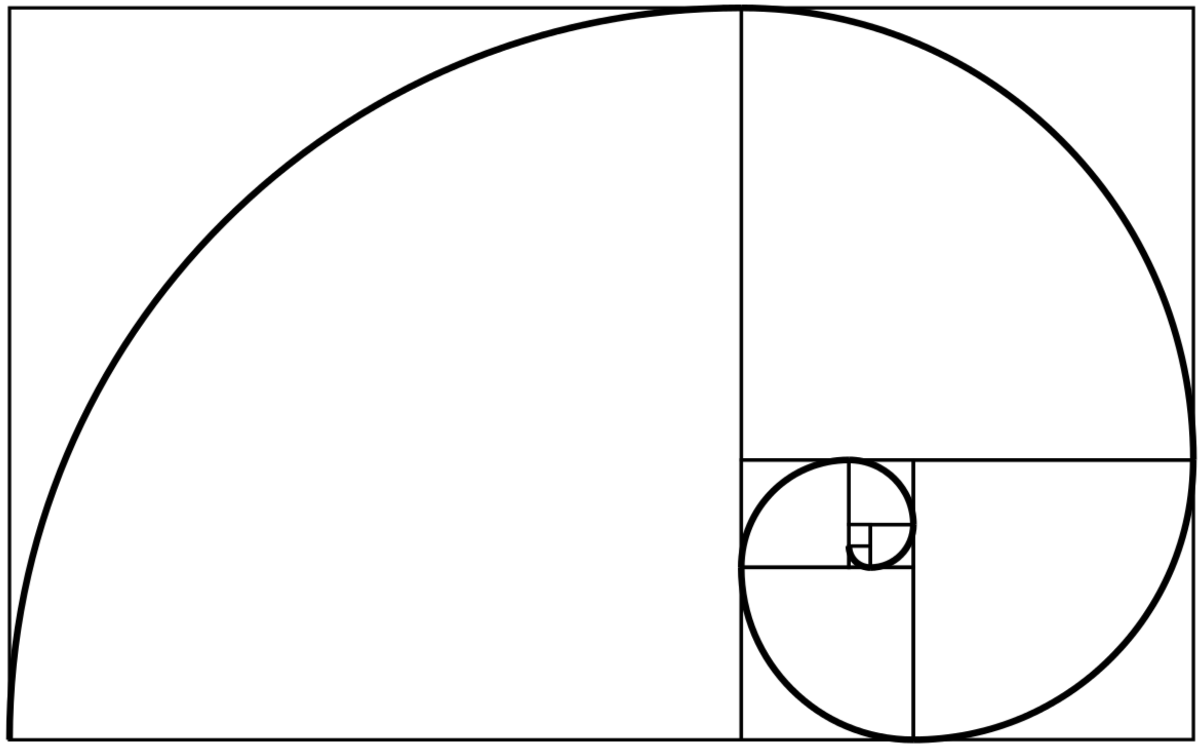
- 1, 1, 2, 3, 5, 8, 13, 21, …

Length of new
Square = 21

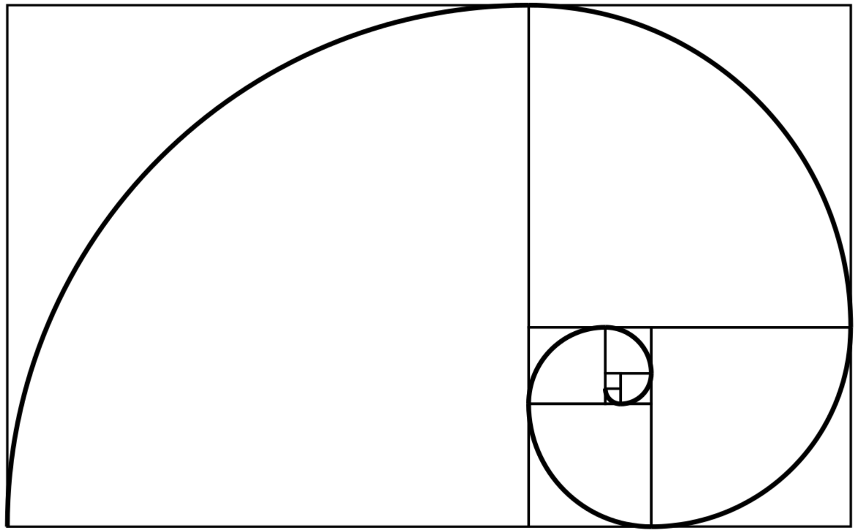


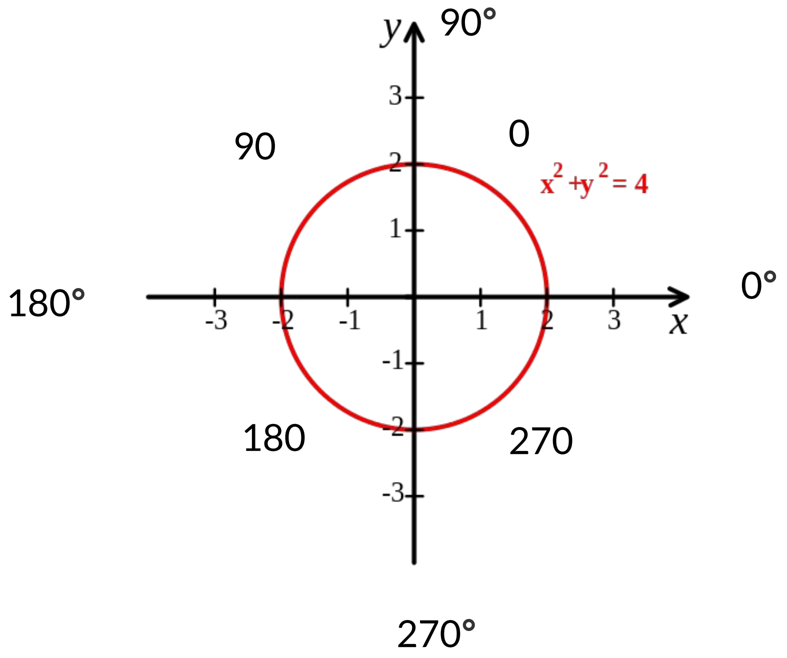
What you have to draw

Length of new Square = 43



```
Arc a = new Arc(centerX, centerY, currRadius, currRadius,
        startAngle, arcLength);
```





`draw(Group group, int centerX, int centerY, int prevRadius, int currRadius, int startAngle, int n)`

- Draw a Fibonacci diagram!
- Draw "`n`" arcs in the diagram
- `centerX` and `centerY` are the coordinates of the center of the **square** the arc is in
- `startAngle` ranges from 0 - 360°

How to select next `startAngle`?

- The arc grows counterclockwise from the middle
- It is rotated left during each iteration
- What would you add to the current angle?
- Make sure it does not go out of bounds - 360 degrees

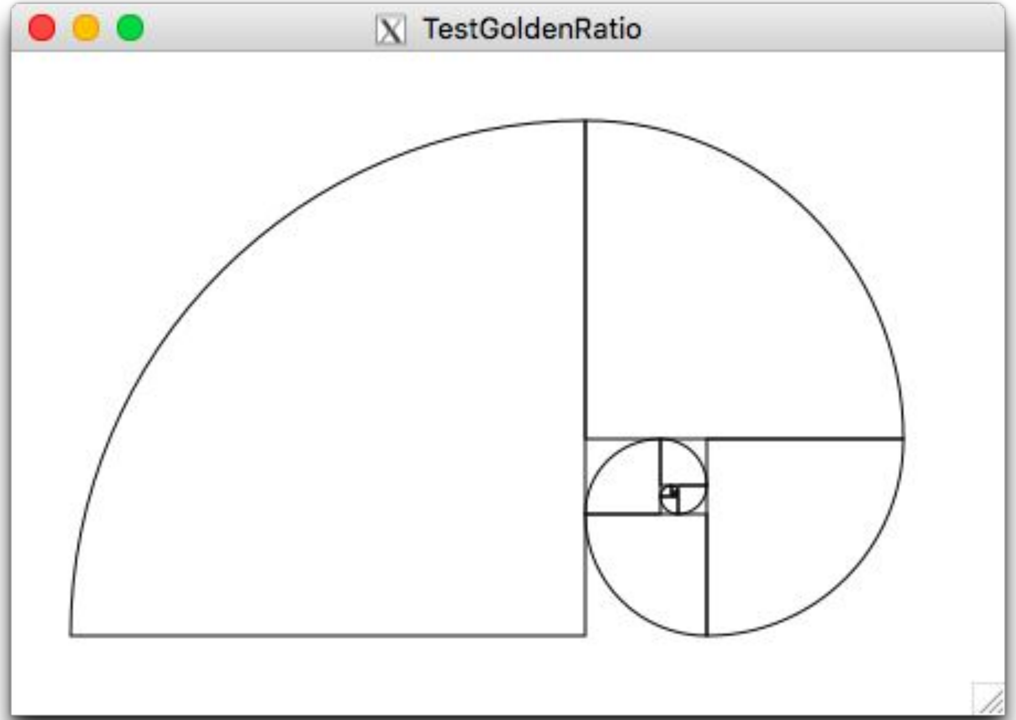
How to select next `centerX` and `centerY`?

- Depends on what the current angle is
- Think of what the four different scenarios are
- Accordingly, we want to move the center along the x and y axis

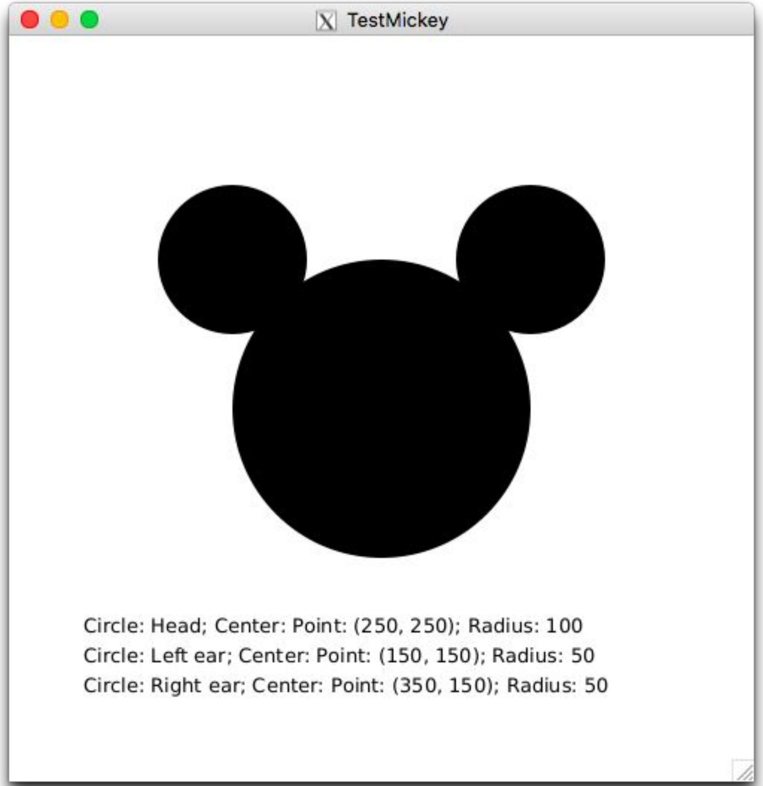
Now what?

- What will the new radius be?
  - Hint: what property does our sequence have?
- Once you have all the values for the **next** arc that will be drawn, recurse!

TestGoldenRatio.java

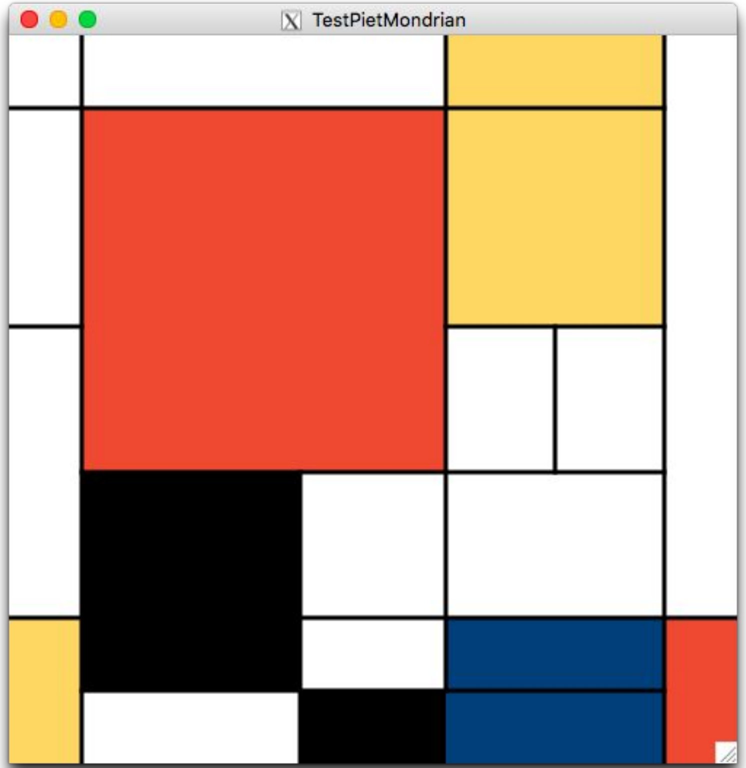


TestMickey.java



```
Circle: Head; Center: Point: (250, 250); Radius: 100
Circle: Left ear; Center: Point: (150, 150); Radius: 50
Circle: Right ear; Center: Point: (350, 150); Radius: 50
```

TestPietMondrian.java



… there are more tests given to you in the starter files

Write your own to see if they match intended behavior!