What are we covering today?

1. A new design pattern (Composite)
2. Mouse Event handler
Lots of things are Hierarchical

How could you manage the hierarchy in a document so that you could perform functions (such as resize) groups of objects together?

An Entire Book

In the real world and in computer systems we often see examples of compositional hierarchies.
Whole-Part Relationships

**Problem:**
How can we treat a group or composition structure of objects the same way (polymorphically) as a non-composite (atomic) object?

**Solution:**
Define classes for composite and atomic objects so that they implement the same interface.

---

**Composite design pattern**

```java
public void operation() {
    for every element in components
        element.operation()
}
```
A Code Example

```java
// leaf node
class Block implements Group {
    String blockName;
    public Block(String blockName) {
        this.blockName = blockName;
    }
    public void assemble(int level) {
    }
}

// composite node
class Structure implements Group {
    // collection of child groups.
    private List<Groups> childGroup = new ArrayList<Groups>();
    String blockName;
    public Structure(String blockName) {
        this.blockName = blockName;
    }
    public void add(Groups group) {
    }
    public void remove(Groups group) {
    }
    public void assemble(int level) {
    }
}

public class ImplementComposite {
    public static void main(String[] args) {
    }
}
interface Group {
    public void assemble(int level);
}
```

Another Example

Uses the mouse to add black circles and colored rectangles onto the canvas.
Uses the mouse to drag a black circle into a rectangular container.
Once the circle is placed into the container it takes on the container’s color.
If we drag a container, all circles within it are also dragged.
Creating a Mouse Event Handler (JavaFX)

```java
@override
public void start(Stage primaryStage) throws Exception {
    root = new AnchorPane();
    scene = new Scene(root, 500, 500);
    scene.setOnMouseClicked(mouseHandler);
    scene.setOnMouseDragged(mouseHandler);
    scene.setOnMousePressed(mouseHandler);
    scene.setOnMouseReleased(mouseHandler);
    primaryStage.setTitle("Shape composer");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

```java
EventHandler<MouseEvent> mouseHandler = new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent mouseEvent) {
        // Put your mouse handling functionality here!
    }
};
```

Put your mouse handling functionality here!

Continues here!

Code inside the event handler:

Note particularly:
- `mouseEvent.getX()`
- `mouseEvent.getY()`
- `mouseEvent.getEventType.getName();`

Also note:
- `MOUSE_RELEASED`
- `MOUSE_DRAGGED`
- `MOUSE_PRESSED`

Etc.
In groups of 3-4 think about the classes that you might need to implement a snakes and ladders game. (Don’t worry about the GUI components for this exercise).

Use the CRC cards to identify classes and responsibilities. Sketch out the finished UML design on paper and we’ll do some show and tell.