Today’s topic:

- Inner classes
- Event driven programming
Inner Classes

- We have seen that typically each class is created in its own .java code file.

- Sometimes we may want to create a secondary class within a class.

- These classes inside of classes are referred to as nested or inner classes.

Because the inner classes are defined within the outer class:

- Inner class can access all instance variables and methods of the outer class.

- Outer class can access all instance variables and methods of the inner class.

- Instances of the inner class exist inside of an instance of the outer class, and are not instantiated on their own.

- Inner classes are inherited by any class that extends the outer class, however, the inner class cannot be overridden.
Inner Classes

Why use inner classes?

- They can provide helper functionality for implementing the outer class methods. In a similar way to how helper methods can be utilized.
  - If the class is only going to be used by the outer class, this is a logical way to organize it.
  - This can also make the code more readable as the code is combined in a single file.

Inner Classes

Why use inner classes?

- Increases encapsulation and abstraction.
  - Instead of changing the outer classes instance variables from private to public, so the other class can access them, we can encapsulate the second class inside the outer class.
  - This allows for encapsulating the inner class data and methods inside the outer class.
  - It also abstracts the details of the inner class because it is hidden and doesn’t need to be seen by outside code.
Event Driven Programming

- The program control centers around *events* that the user causes.

- The user *interacts* with the objects, and they respond and take control as necessary.

- In event-driven programming sometimes the user is a *person*, and other times it may be *other programs*.

---

Event Driven Programming

- When coding an event driven program, you will be writing *listeners* that wait for an event to occur, and then run when the event *occurs*.

- These listeners will most often be written as inner classes as part of our interface.
Event Driven Programming

- Types of events:
  - Button clicks
  - Mouse Events
    - mousePressed Mouse button pressed on a component.
    - mouseReleased Mouse button released on a component.
    - mouseClicked Mouse button clicked on a component.
    - mouseEntered When mouse enters a component.
    - mouseExited When mouse exits component.
  - Timers

Inner Classes

An example:

- Inner classes are used extensively in GUI applications.

- Let’s look at our DiePanel class:
  - There is an inner class named RollListener
  - An instance of this object is attached to the roll button, it’s job is to wait for the button to be clicked.
  - There would be no use for this class outside of the DiePanel class so we make it an inner class.