USF UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

Nested Classes

CS 272 Software Development

Professor Sophie Engle
Department of Computer Science

Nested Classes

- A class defined within another class
  - Not defined in a separate *.java file
  - Only accessible through outer class
- Can improve maintainability
  - Reduces number of files (but not number of classes)
  - Groups together related classes into one file

https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

Nested Classes

- Increases **encapsulation**/**information hiding** for small classes used by only one other class
- Can combine **composition** and **inheritance** relationships
  - Outer class **is**-a other class
  - Outer class **has**-a inner class
  - Inner class **is**-a other class

https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

Types of Nested Classes

- **Inner** Class
  - A non-static nested class **dependent** on a specific instance of the outer class
- **Static Nested** Class
  - A static nested class that is **independent** of an instance of the outer class

https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

Nested Class Initialization

- Inner Class
  - Must have instance of outer class to initialize
  - `Outer outer = new Outer();`
    `Outer.Inner in = outer.new Inner();`
- Static Nested Class
  - May initialize without an instance of outer class
  - `Outer.Static in = new Outer.Static();`

https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

Outer Member Access

- Inner Classes
  - May access private instance members of outer class
  - May access private class members of outer class
- Static Nested Classes
  - May *not* access any instance members of outer class
  - May access any class members of outer class

https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

Anonymous Classes

- A nested, local class without an explicit name
  - Declared, defined, and instantiated at same time
  - Specify one class or interface to extend/implement
  - Give implementations of abstract methods
- Used for classes defined and used only once
  - Commonly used to create `Comparator` objects
  - Commonly used in multithreading

https://docs.oracle.com/javase/tutorial/java/javaOO/nested.html

Example Anonymous Class

```
Runnable r = new Runnable() {
  public void run() {
    System.out.println("Hello!");
  }
}; // note semicolon!
```

Questions?