# CS 351
# Design of Large Programs
# Creating Threads: Platform and Virtual

Brooke Chenoweth

University of New Mexico

Fall 2026

# How to create a Thread?

- Directly extend Thread
- Implement Runnable
  - Pass Runnable object to Thread constructor
  - Use a Thread.Builder or ThreadFactory

# Runnable is generally better

- Decouples task to run from Thread object
- Can use an Executor
- Can choose to use platform or virtual threads

# Executor and ExecutorService

- An Executor will somehow execute a Runnable task.
- ExecutorService extends this with ability to submit Callable objects that return a result in the Future.
- ScheduledExecutorService allows scheduling tasks to run after a delay
- Executors has methods to create many ExecutorService implementations

# Future

- A Future represents the result of an asynchonous task.
- A Future object is returned ater submitting a Callable object to an Executor service.
- The get method will block the current thread if the result isn't ready.
- isDone lets you check if the computation is finished

# Thread Pools

- I have many tasks to run.
- The system supports a limited number of Threads.
- Creating new threads can be expensive
- A thread pool creates some number of threads and has them run the tasks.

# Platform and Virtual Threads

- Until JDK 21, all threads in Java were *platform threads*
  - Instance of java.lang.Thread
  - A wrapper around an OS thread.
  - Limited to number of OS threads.
  - Limited resource
- In JDK 21, added *virtual threads*
  - Instance of java.lang.Thread
  - Still runs code on an OS thread
  - When blocked, Java runtime suspends it, freeing the OS thread to run other virtual threads.
  - Suitable for tasks that spend most time blocked (waiting for IO, for example)

# Virtual Threads

- https://docs.oracle.com/en/java/javase/25/core/vir threads.html
- Use Thread.ofVirtual() to create a new virtual thread
- Use Executor.newVirtualThreadPerTaskExecutor() for ExecutorService that uses virtual threads.