# Callable + Future

Callable + Future: Necessita de gerenciador de tarefas, faz sentido isso para o trabalho? Como definir qual gerenciador usar?

- Callable é uma interface funcional que cria uma tarefa
- Future é um objeto que representa o resultado de uma tarefa
- Normalmente utiliza ExecutorService ou Thread Pool. Thread pool é um mecanismo de gerenciamento de threads
- Pode ser utilizado também com FutureTask + Thread
- FutureTask é uma classe que implementa Runnable e Future. Ela serve como uma ponte entre Callable e Thread

# Comparativo: Thread (herança) vs Runnable

| Aspecto | Thread (herança) | Runnable |
|---|---|---|
| Objeto de thread | Cada thread é um objeto distinto | Várias threads podem compartilhar o mesmo objeto Runnable |
| Estado compartilhado | Cada thread tem seu próprio conjunto de variáveis (estado isolado). Não há interferência entre threads | Se o mesmo Runnable for usado em várias threads, o estado interno do objeto é compartilhado. Variáveis da classe Runnable podem ser alteradas por várias threads ao mesmo tempo |
| Flexibilidade | Menos flexível (não pode estender outra classe) | Mais flexível, pode implementar Runnable e estender outra classe |
| Gerenciamento | Cada thread é autônoma, menos risco de conflito de estado interno, mas menos controle centralizado | Necessário gerenciar acesso ao estado compartilhado, garantindo sincronização quando necessário |

# Quando utilizar

## Thread

- Cada thread precisa de um objeto independente
- A classe não precisa estender outra
- Estado compartilhado não é necessário
- Mais simples de implementar

## Runnable

- Várias threads compartilham o mesmo comportamento
- A classe já estende outra
- Para separar a lógica da thread do próprio objeto Thread
- Facilita o reuso