Class Project
MET CS 665
Software Design and Patterns

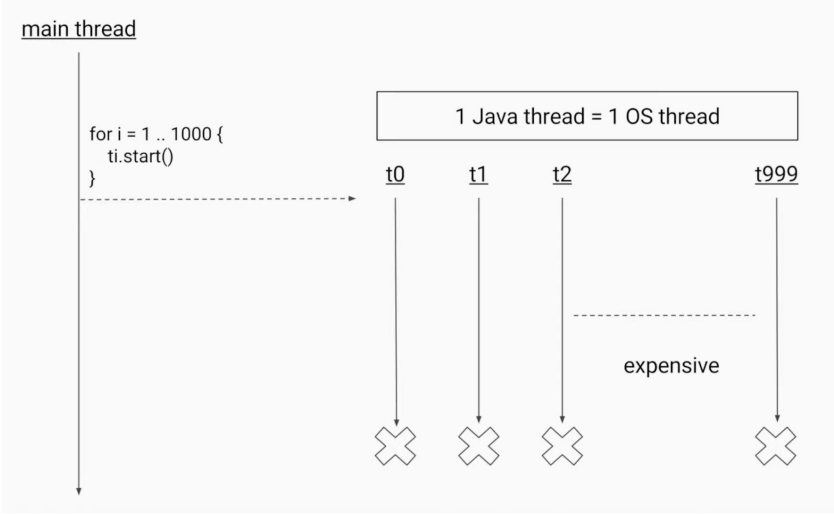
Thread Pool Pattern

Zhaowei Gu

Problem - Thread Pool

When we are dealing with a short-lived task but there is a large number of the tasks that need to be done, creating 1 Java thread for each task will make system spend more time on creating and destroying the threads other than actually executing the tasks.

Thread Pool solves this problem by reusing the thread.



Thread Pool Pattern

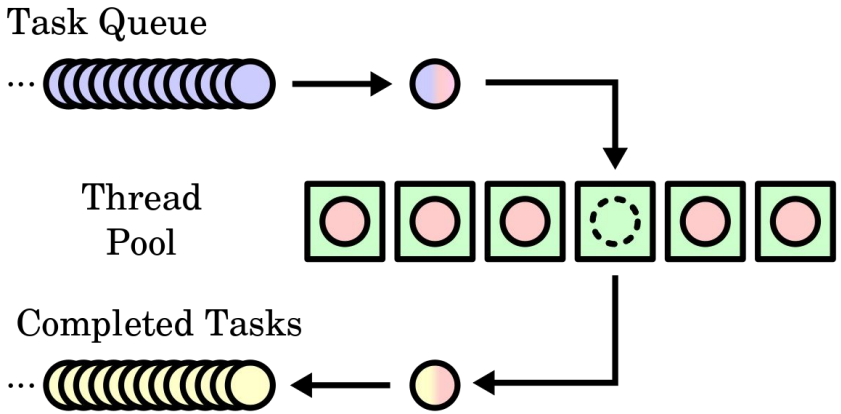
aka:

replicated workers or worker-crew model.

We could see Thread Pool as Cashiers

Task Queue are customer who is looking to checkout in a superMarket

Completed Tasks are the one completed their checkout process and got their receipt



Ideal Pool Size

CPU intensive operation

Example:

- Doing your own operation
- Calculate sum from data input
- Etc..

```
int coreCount = Runtime.getRuntime().availableProcessors();
ExecutorService service = Executors.newFixedThreadPool(coreCount);
```

Ideal pool size:

CPU Core count

i/o intensive operation

Example:

- API
- Waiting for HTTP call
- Waiting for a response

More thread could be used.

Ideal pool size:

A very High number of thread

Type of Thread Pool

1. FixedThreadPool

   ```
   Executors.newFixedThreadPool()
   ```

2. CachedThreadPool

   ```
   Executors.newCachedThreadPool()
   ```

3. ScheduledThreadPool

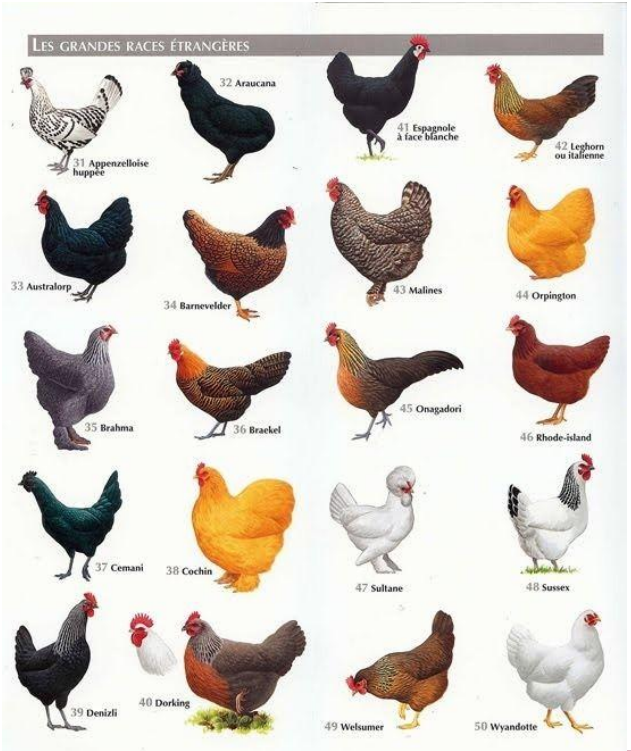
   ```
   Executors.newScheduledThreadPool()
   ```

4. SingleThreadExecutor

   ```
   Executors.newSingleThreadExecutor()
   ```

5. Single Thread Scheduled Pool

   ```
   Executors.newSingleThreadScheduledExecutor()
   ```



Supermarket Example - Thread Pool

```
//get count of available cores, fixed core system, but it changes depends on your system.
int coreCount = Runtime.getRuntime().availableProcessors();
ExecutorService service = Executors.newFixedThreadPool(coreCount);

//Create new file and empty the file
try {
  new OutputStreamWriter(new FileOutputStream( name: "shoppingList.txt",  append: false),
      StandardCharsets.UTF_8).close();
} catch (IOException e) {
  e.printStackTrace();
}

//start running the code
final long startTime = System.nanoTime();

//submit tasks for execution
for (int i = 0; i < customerNumber; i++) {
  service.execute(new Customer(
      new CreateShoppingList(arrOfItems, minBuyAmount, maxBuyAmount)));
}

// All tasks were executed, now start to shut down
service.shutdown();
while (!service.isTerminated()) {
  Thread.yield();
}
```

Supermarket Example - Thread Pool

```
1  GroundTurkey-5.99,GroundBeef-6.89,GreekYogurt-4.99,BabyWipes-12.32,
2  HeavyWhipCream-2.19,LiquidHandWash-1.00,GreenBeans-2.99,Scissor-4.22,Ga
3  Starbucks-5.39,ColoredPencil-5.99,Garlic-1.2,
4  DishwasherDetergent-12.7,LaundryDetergent-8.79,PeroxideToiletCleaner-3.
5  Disinfectingwipes-4.47,Yogurt-1.00,CoconutWater-1.88,MountainDew-8.99,C
6  PeroxideToiletCleaner-3.39,RootBeer-4.99,Bleach-5.48,Grapes-5.98,Coconu
7  AllPurposeCleaner-3.69,Watermelon-3.99,
```

Supermarket Example - Thread Pool

```
private static void supermarketRun(String @NotNull [] inputList, int discountAppliesPercentage,
    int discountPercentage) {
  //get count of available cores, fixed core system, but it changes depends on your system.
  int coreCount = Runtime.getRuntime().availableProcessors();
  ExecutorService service = Executors.newFixedThreadPool(coreCount);

  //Create new file and empty the file
  try {
    new OutputStreamWriter(new FileOutputStream( name: "supermarketReceipt.txt",  append: false),
        StandardCharsets.UTF_8).close();
  } catch (IOException e) {
    e.printStackTrace();
  }

  //start running the code
  final long startTime = System.nanoTime();

  //submit tasks for execution
  for (String line : inputList) {
    service.execute(
        new Cashier(new CheckoutCounter(line, discountAppliesPercentage, discountPercentage)));
  }

  // All tasks were executed, now start to shut down
  service.shutdown();
  while (!service.isTerminated()) {
    Thread.yield();
  }
```

Supermarket Example - Thread Pool

```
Generating testing input text

12 customer generated 10000 shopping list
they done all these in about 0 second

Supermarket day finished with 12 Cashier working together
They have helped 10000 customer in about 0 second

Process finished with exit code 0
```

Supermarket Example - Thread Pool

```
Customer 37 receipt
Item                    Qty      Price
----                    ---      -----
Vaseline                  1       2.19
OilSpray                  1       6.29
LaundryDetergent          1       8.79
PaperTowels               1       7.59
AtlanticSalmon            1       9.99
Peet'sCoffee              1      12.99
-------------------------------------
Discount:                         -4.78
Total:                            43.06
```

When use the Thread Pool Pattern

Use the Thread Pool pattern when you have a large number of short-lived tasks to be executed in parallel.

The End
Thank you for watching.