

Flyweight DP

30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in

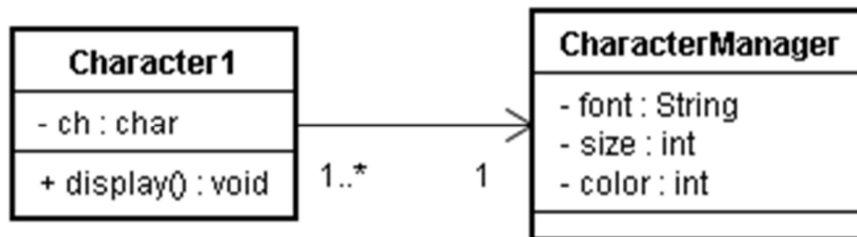
250

Problem

- We have a Character class. It stores information about one character.
 - Our document has a large number of characters.
 - Each character can have a different font and size.
 - How to reduce the memory requirements and runtime overheads?



Solution



30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in

252

Designing objects down to the lowest levels of system "granularity" provides optimal flexibility, but can be unacceptably expensive in terms of performance and memory usage.

Flyweight performs "use sharing" to support large number of fine-grained objects efficiently.

A flyweight is a shared object that can be used in multiple contexts simultaneously.

=====

The Unicode of the characters can be stored in an array.

Many of the characters have the same attributes e.g. font, font size, color, etc. We can store these common attributes for a set of characters only once.

Usage of Flyweight DP

- String in Java
- Wrapper Classes in Java
- TimerTask
- Should Dimension be Flyweight?

30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in

253

Java uses the Flyweight pattern to manage the String objects used to represent string literals.

All wrapper classes are Flyweight

TimerTask

Dimension class should have been a Flyweight

When to apply Flyweight?

- Always.
 - Unless you have a very good reason, make your class immutable.
 - Getters should normally return immutable objects.
 - Use `java.util.collections.unmodifiableList`

This rule is applicable only if the language has automatic garbage collection.

30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in

254

If you cannot make the entire class immutable, make maximum part immutable.
Limit mutability as much as you can.
The less mutable the better.
Final is the new private.

Assignment

- We have many user accounts. Permissions for each user are defined at group level.
 - A user can be in multiple groups.
 - Where can we use the flyweight pattern?

30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in



255

OOAD classes: Different classes (Immutable), Prefer Value Objects, Rules for Concurrency,

How many Flyweights?

- Java 8 has a new package `java.time`
 - What percentage of classes are Immutable?
 - If a mutable class exists, then justify its design.

Bounded Buffer Problem

- N buffers. Each can hold one immutable item.
- A set of producers produce the item and place in buffer.
- A set of consumers remove the item from buffer and process the same.
- Producers should wait till at least one buffer free.
- Consumers should wait till at least one buffer is filled.

30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in

257

Code this problem in Java. Assume a set of 3 threads produce ten million integers. A set of 2 threads consume these integers. Output should be how many evens and how many odds.

Reader-Writer Problem

- A object is shared among many threads
 - Readers – only read the collection; they do *not* perform any updates
 - Writers – can both read and write
- Problem – allow multiple readers to read at the same time
 - Only one single writer can access the shared data at the same time
- Compare performance with immutable object being shared. So no locks are needed.

30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in

258

Shared Data

Data set

Semaphore `rw_mutex` initialized to 1

Semaphore `mutex` initialized to 1

Integer `read_count` initialized to 0

The structure of a writer process

```
do {  
    wait(rw_mutex);  
  
    ...  
    /* writing is performed */  
    ...  
    signal(rw_mutex);  
} while (true);
```

The structure of a reader process

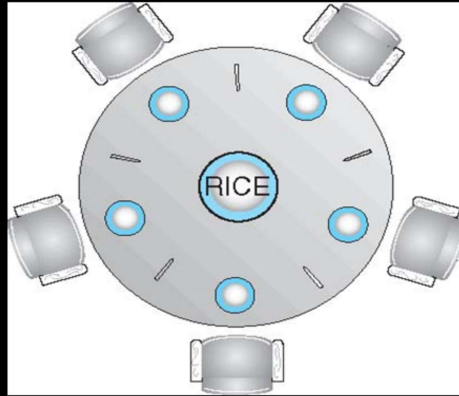
```
do {  
    wait(mutex);  
    read_count++;  
    if (read_count == 1)
```

```
        wait(rw_mutex);
    signal(mutex);

    ...
    /* reading is performed */
    ...
    wait(mutex);
    read count--;
    if (read_count == 0)
        signal(rw_mutex);
    signal(mutex);
} while (true);
```

Dining Philosophers Problem

- Simulate



30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in

259

Philosophers spend their lives thinking and eating

Don't interact with their neighbors, occasionally try to pick up 2 chopsticks (one at a time) to eat from bowl

Need both to eat, then release both when done

In the case of 5 philosophers

Shared data

Bowl of rice (data set)

Semaphore **chopstick** [5] initialized to 1

The structure of Philosopher *i*:

```
do {  
    wait (chopstick[i] );  
    wait (chopstick[ (i + 1) % 5] );  
  
    // eat  
  
    signal (chopstick[i] );  
    signal (chopstick[ (i + 1) % 5] );  
}
```

```
        // think  
  
    } while (TRUE);
```

What is the problem with this algorithm?

The last philosopher must take spoons in reverse order, otherwise there is a deadlock.

Moral: Rice is used only one thread. So no locking. Spoon is like a mutable object; only one thread can use it.

A large application will have some mutable objects shared. First try to make it immutable. If that is not possible, then ensure locks are acquired in same order.

Assignment

- We want to cache immutable objects.
- Create a LRU cache for immutable objects
 - Hint: Use `removeEldestEntry` function in `LinkedHashMap`

30-Oct-15 9:21 PM

By Vijay – caretrainings.co.in

260

Java Solution in Patterns project FlyweightCache.