Thread safeچیست؟

کد thread-safety یا thread-safe در جاوا به کدهایی اطلاق می شود که به طور ایمن می توان از آنها در محیط همزمان یا چند رشته ای استفاده کرد یا به اشتراک گذاشت و مطابق انتظار رفتار خواهند کرد.

آیا متد String یک Thread safe است؟

اشیاء غیرقابل تغییر به طور پیش‌فرض از نظر thread ایمن هستند زیرا حالت آنها پس از ایجاد نمی‌تواند تغییر یابد. از آنجایی که String در جاوا تغییر ناپذیر است، ذاتاً ایمن است.

```
public class StringThreadSafetyExample {
    public static void main(String[] args) {
        String str = "Hello, World!";

        // Thread 1
        Thread thread1 = new Thread(() -> {
            System.out.println("Thread 1: Length of string: " + str.length());
        });

        // Thread 2
        Thread thread2 = new Thread(() -> {
            System.out.println("Thread 2: Character at index 7: " + str.charAt(7));
        });

        thread1.start();
        thread2.start();

        // Wait for both threads to finish
        try {
            thread1.join();
            thread2.join();
```

```
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

مفهوم Live lock

Livelock یکی دیگر از مشکلات همزمانی است و شبیه به بن بست است. در Livelock، دو یا چند رشته به جای انتظار بی‌نهایت همانطور که در مثال بن‌بست دیدیم، به انتقال حالت‌ها بین یکدیگر ادامه می‌دهند. در نتیجه، نخ ها قادر به انجام وظایف مربوطه خود نیستند.

تفاوت آن با گرسنگی و بن بست چیست؟

اگرچه ماهیت مشابهی دارند، اما بن بست و قفل های زنده یکسان نیستند. در یک بن بست، فرآیندهای درگیر در یک بن بست به طور نامحدود گیر می کنند و هیچ تغییر حالتی ایجاد نمی کنند. با این حال، در یک سناریوی قفل زنده، فرآیندها یکدیگر را مسدود می کنند و به طور نامحدود منتظر می مانند، اما آنها به طور مداوم وضعیت منبع خود را تغییر می دهند. نکته قابل توجه این است که تغییر وضعیت منبع تأثیری ندارد و به فرآیندها کمک نمی کند تا در کار خود پیشرفت کنند.