**The Snake Race**

**Problemas de Concorrência**

* **Acesso à celula**

O principal problema de concorrência com que nos defrontamos foi quando duas ou mais cobras tentam aceder à mesma célula simultaneamente. A célula é o recurso partilhado e se o acesso não for controlado poderá surgir problemas.

Começámos por modificar o movimento de cada cobra. Para uma cobra poder aceder a uma célula terá que a “reservar” primeiro, e para o fazer terá de verificar se alguma outra cobra ocupa essa célula. Caso esteja livre, a cobra movimenta-se para dentro da célula. Caso contrário, fica em espera – wait().

Esta verificação tem de ser feita de forma sincronizada, isto é, a Thread da cobra realiza esta operação de maneira indivisível (atomicidade) para impedir problemas gerados pela concorrência.

Após a reserva ter sido efetuada com sucesso, o estado da célula passa para cheio e todas as Threads em espera são notificadas – notifyAll().

Qualquer cobra que a tente aceder ficará em espera.

O movimento da cobra pela célula fica completo quando a ultima parte da cobra sai da célula. Nesse momento é quando libertamos a célula, também de forma sincronizada, e informamos todas as threads em espera que a célula foi libertada. Se alguma cobra estivesse a espera para entrar nessa célula, essa notificação iria “acordá-la” e permitir que ela continuasse o movimento.

* **Deadlock**

O tratamento da concorrência no acesso à célula traz-nos o seguinte problema – caso duas cobras tentem aceder a células ocupadas pela outra ficam as duas em à espera [wait()] que a outra liberte o cadeado. A maneira como resolvemos este problema foi através de um ActionListener que desbloqueia a cobra selecionada através de um interrupt() que dá um novo caminho à mesma.