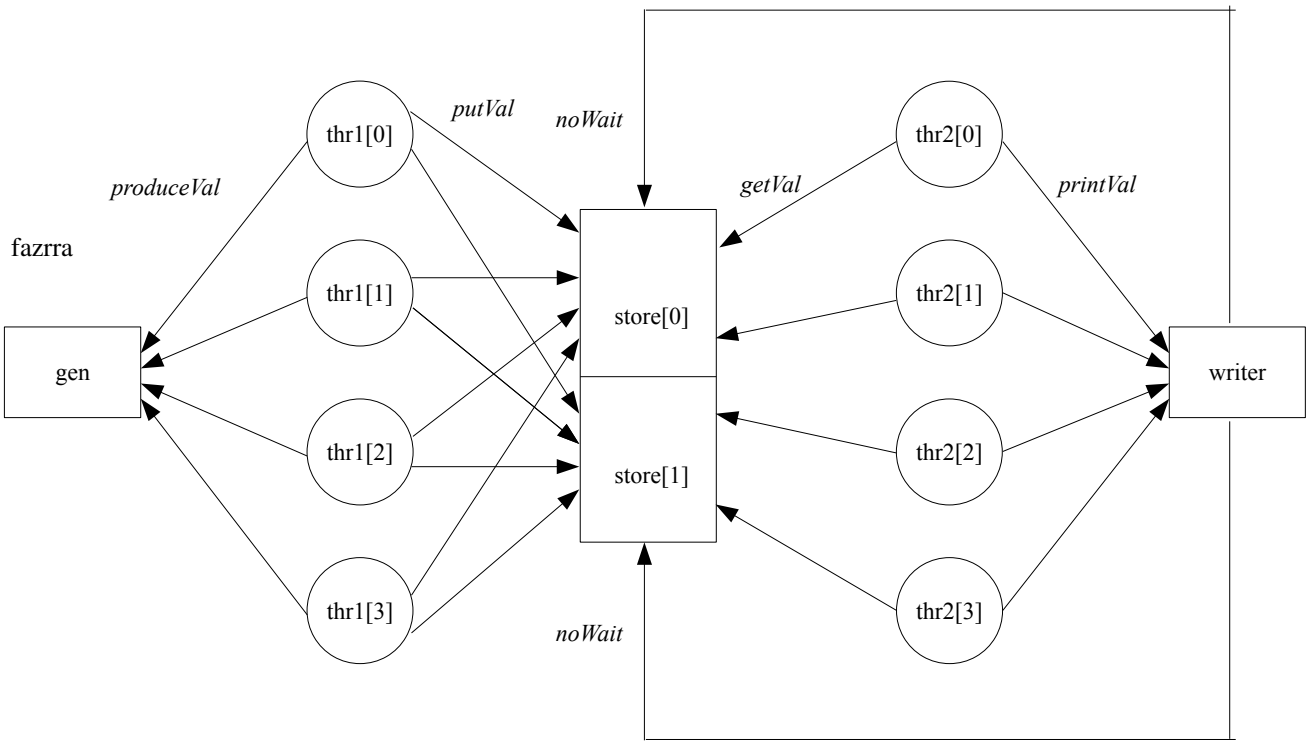
RESOLUÇÃO

1. Representing the active entities by circles and the passive entities by squares, sketch a diagram which illustrates the interaction that is taking place and describe in simple words the role played by the threads of each type (do not use more than one or two sentences in your explanation).



Os *threads* de tipo 1 lêem sucessivamente valores da memória *gen* e escrevem-nos em dobro na memória *store*[0], se forem múltiplos de três, e em singelo na memória *store*[1], caso contrário; terminam quando lerem um valor nulo. Os *threads* de tipo 2 lêem valores das memórias *store*[0], os dois primeiros, e *store*[1], os dois últimos, e imprimem-nos indicando os *ids* dos *threads* envolvidos no seu processamento; terminam quando já não há mais valores a ler.

2. Assume that, when the program is run, one gets the following printing

```
The value processed by 1 and by 2 was 24.
The value processed by 3 and by 1 was 3.
The value processed by 1 and by 2 was 12.
The value processed by 2 and by 4 was 5.
The value processed by 4 and by 3 was 4.
The value processed by 4 and by 4 was 11.
The value processed by 3 and by 2 was 36.
The value processed by 1 and by 3 was 2.
The value processed by 4 and by 4 was 7.
The value processed by 2 and by 1 was 30.
The value processed by 4 and by 4 was 16.
The value processed by 1 and by 1 was 18.
```

   Bear in mind that, because of the randomness which was introduced, this is not the only possible result. In fact, it is not even correct. There are three errors in the printing at lines 2, 6 and 9, respectively. Identify them and justify your claims carefully.

O erro na linha 2 tem a ver com o valor impresso, 3. Sendo múltiplo de 3, o valor correcto seria 6.
O erro na linha 6 tem a ver também com o valor impresso, 11. 11 não está armazenado na memória *gen*.
O erro na linha 9 é mais subtil, resulta de uma violação do princípio da causalidade. Os valores processados pelo *thread* de tipo 1, 4, e pelo *thread* de tipo 2, 4, listados nas linhas 9 e 11, 7 e 16, respectivamente, surgem na ordem inversa com que estão armazenados na memória *gen* (valores processados pelo mesmo par de *threads* têm que ser impressos na ordem de armazenamento).

3. Explain how the termination of the program is always ensured.

Tal como foi construido o ciclo de vida, os *threads* de tipo 2 não têm controlo directo sobre a sua terminação. Quando o último valor é impresso, o *thread* que o fez termina de imediato. Os restantes, porém, estarão em princípio bloqueados, ou virão a estar, nas variáveis de condição implícitas dos monitores *store*[0] e *store*[1]. Assim, a operação *noWait* é executada nesta altura sobre os dois monitores por este *thread*, acordando os restantes *threads* bloqueados ou impedindo o seu bloqueio, e permitindo portanto a sua terminação.

4. Change the code so that it becomes possible to process pairs of values (the values of each pair should be added). Start your answer by pointing out the changes that have to be made. They should be <u>minimal</u>.

As alterações mínimas a efectuar são as seguintes:

1 - Na instanciação de *writer*, substituir 12 por 6, porque agora vão ser 6 pares de valores.

2 - A instrução de impressão do método *printVal* tem que ser agora adequada ao caso em que há dois *threads* de tipo 1 e um *thread* de tipo 2 a efectuar o processamento do valor.

3 - Devido à alteração do diagrama de estados interno do tipo de dados *StoreRegion*, os métodos *putVal*, *getVal* e *noWait* sofrem ligeiras alterações.

Deixo a escrita do respectivo código para vocês.