Vola5ile

* Keyword
* Tells compiler not to optimise access and forces next read to see the last write

Class lookMayNeverEnd {

Boolean done = false;

void work() {

While !(done}

//do work

}

Void stopWork() {

Done = true;

}

}

* In above loop the compiler may never actually read the stopWork function. This means that the done will never change to true and the loop may never end
  + Because 1 loop calls work and another calls stopWork
* Using volatile stops this
* This is more expensive but not as expensive as synchronise

Publishing

* A private variable accessed by a public method is essentially public, therefore published
* Can therefore however be accessed by multiple threads

Thread confinement

* if data doesn’t need to be shared then confine it
* Confine means limit it to a signle thread

Lambda expressions

* Can be passed around between objects
  + Normally only objects or primitives can be passed