```
/** PCO 2015 YTA Buffer2Conso. Tampon simple, 1 producteur - 2 consommateurs.
 * --- 3eme solution - avec moniteurs/QWaitCondition ---
 * Objectif: parfaitement fonctionnel. Commentaire: parfait (avec corrections...)
template<typename T> class Buffer2Conso3 : public AbstractBuffer<T> {
private:
    T element; // contenu du buffer
    QWaitCondition* waitIsFull; // consommateur en attente que le buffer soit plein
    QWaitCondition* waitIsEmpty; // producteur en attente que le buffer soit vide
    QMutex* mutex; // protège les variables ci-dessous
    bool full; // indique l'état du tampon: plein ou vide.
    int nbConsommation; // nombre de consommation effective du tampon.
public:
    Buffer2Conso() {
        waitIsFull = new QWaitCondition();
        waitIsEmpty = new QWaitCondition();
        mutex = new QMutex();
        nbConsommation = 0;
        full = false; // vide
    }
    virtual ~Buffer2Conso() {
        delete waitIsFull;
        delete waitIsEmpty;
        delete mutex;
    }

    virtual void Buffer2Conso::put(T item) {
        mutex->lock();
        while (full) { // prélude: si c'est déjà plein: on se met en attente.
            waitIsEmpty->wait(&mutex);
        }
        element = item; // critique: poser l'element.
        full = true; // post-lude: on libère un consommateur s'il y en a en attente
        waitIsFull->wakeOne(); // réveille éventuel d'un consommateur en attente
        mutex->unlock(); // à faire APRES le wake (ici serait aussi ok avant car wh
    }

    virtual T Buffer2Conso::get(void) {
        mutex->lock();
        while(!full) { // prélude: s'il n'y a rien à consommer: on se met en attent
            waitIsFull->wait(&mutex);
        }
        T item = element; // section critique: récupérer l'élément.
        nbConsommation++; // il y a eu consommation!
        if (nbConsommation == 2) { // nombre de conso max: reset, le tampon devient
            nbConsommation = 0; // reset des consommations
            full = false;
            waitIsEmpty->wakeOne(); // réveil éventuel d'un producteur en attente.
        } else { // sinon, il peut encore y avoir consommation.
            waitIsFull->wakeOne(); // réveil éventuel d'un 2e consommateur en atter
        }
        mutex->unlock(); // à faire APRES le wake (ici serait aussi ok avant car wh
        return item; // on retourne l'élément
    }
};
```