Binôme : KOUROUMA & KERMORGANT

M1 MIAGE

Les problèmes de synchronisations :

1-Identifier les méthodes avec accès concurrent et empêcher les accès concurrents

Pour empêcher les accès concurrents à une même donnée, il est possible d'utiliser un mécanisme de verrouillage qui va bloquer l'accès à une donnée le temps qu'un autre thread accède déjà à la donnée. Ainsi si plusieurs threads tentent d'accéder à une même donnée en même temps, ces accès se feront les uns après les autres.

Les méthodes de synchronisations

1-synchronisation de la méthode recharger du responsable

doitAttendre est un Boolean initialisé a true et passe a false uniquement quand le stock dépasse un certain seuil .

```
public synchronized void recharger(int stockInit) {
		while(doitAttendre) {
			try {
				wait();
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
		stock+=stockInit;
		doitAttendre = true;
		System.out.println("le responsable vient de rajouter "+stockInit+"
billets");
		notifyAll();

	}
```

2- synchronisation de la méthode acheter du client

Le client peut effectuer un achat tant que le nombre de ticket disponible est supérieur a la quantité qu'il veut acheter.

```
public synchronized void acheterBillet(int nbBillets) throws InterruptedException
{

		while (this.stock < nbBillets) {
			doitAttendre = false;
			notifyAll();
			System.out.println("le client veut acheter   " + nbBillets +
"billet(s)");
			wait();


		}
		System.out.println("le client a achetÃ© "+ nbBillets + "billet(s)");
		Thread.sleep(tpsAchat);
		this.stock = this.stock - nbBillets;
		notifyAll();
}
```

4-synchronisation des méthodes de la classe java :

Méthode Arret : empêcher plusieurs Navette de stationner au quai

```
synchronized void arret(Navette navette) {
		while (this.navette != null) {
			try {
				System.out.println("Attend de rentrer");
				wait();
			} catch (InterruptedException e) {
			}
		}
		this.navette = navette;
		System.out.println("navette au quai");
	}
```

Méthode depart: Libérer le verrou après le départ et notifier les autres navettes

```
	synchronized void depart() {
		this.navette = null;
		notifyAll();
	}
```

Méthode traitementClient: empêcher un client de monter dans une navette qui n'est pas sur le quai ou dont la place est 0, sinon décrémenter le nombre de place dans la navette.

```
	public synchronized void TraitementClient() throws InterruptedException {
		while (navette == null ||  navette.getNbPlacesDispo() == 0) {

			System.out.println("le client attend la navette.");
			wait();
		}
		navette.setNbPlacesDispo(navette.getNbPlacesDispo()-1);
		//client.sleep(200);
		System.out.println("le client monte dans la
navette."+navette.getNbPlacesDispo()+"places restantes");
	}
```