Teoria Współbieżności

1 Zadanie

1. zaimplementować semafor binarny (patrz listing 1) za pomocą metod *wait* i *notify*, użyć go do synchronizacji programu Wyścig (zajęcia 1).

2. pokazać, że do implementacji semafora za pomocą metod *wait* i *notify* nie wystarczy instrukcja *if* tylko potrzeba użyć *while*. Wyjaśnić teoretycznie dlaczego i potwierdzić eksperymentem w praktyce. (Wskazówka: rozważyć dwie kolejki: czekającą na wejscie do monitora obiektu oraz kolejkę zwiazaną z instrukcją *wait*, rozważyć kto kiedy jest budzony i kiedy następuje wyścig).

3. Zaimplementować semafor licznikowy (ogólny) (patrz sekcja 2) za pomocą semaforów binarnych. Czy semafor binarny jest szczególnym przypadkiem semafora ogólnego? Dlaczego?

2 Semafory

2.1 Semafor binarny

- Abstrakcja współbieżna
- Jeden z dwóch stanów: podniesiony, opuszczony
- Dwie metody podnieś, opuść

2.2 Semafor licznikowy

- Synchronizuje dostęp do współdzielonej puli zasobów
- Zmienna całkowita

- Metoda podnieś, zajmuje jedną jednostkę zasobu
- Metoda opuść, zwalnia jedną jednostkę zasobu
- Semafor inicjalizowany jest ilością dostepnego zasobu

Dodatki

Listing 1: Semafor binarny

```
class Semafor {
    private boolean _stan = true;
    private int _czeka = 0;

    public Semafor() {
    }

    public synchronized void P() {

    }

    public synchronized void V() {

    }
}
```