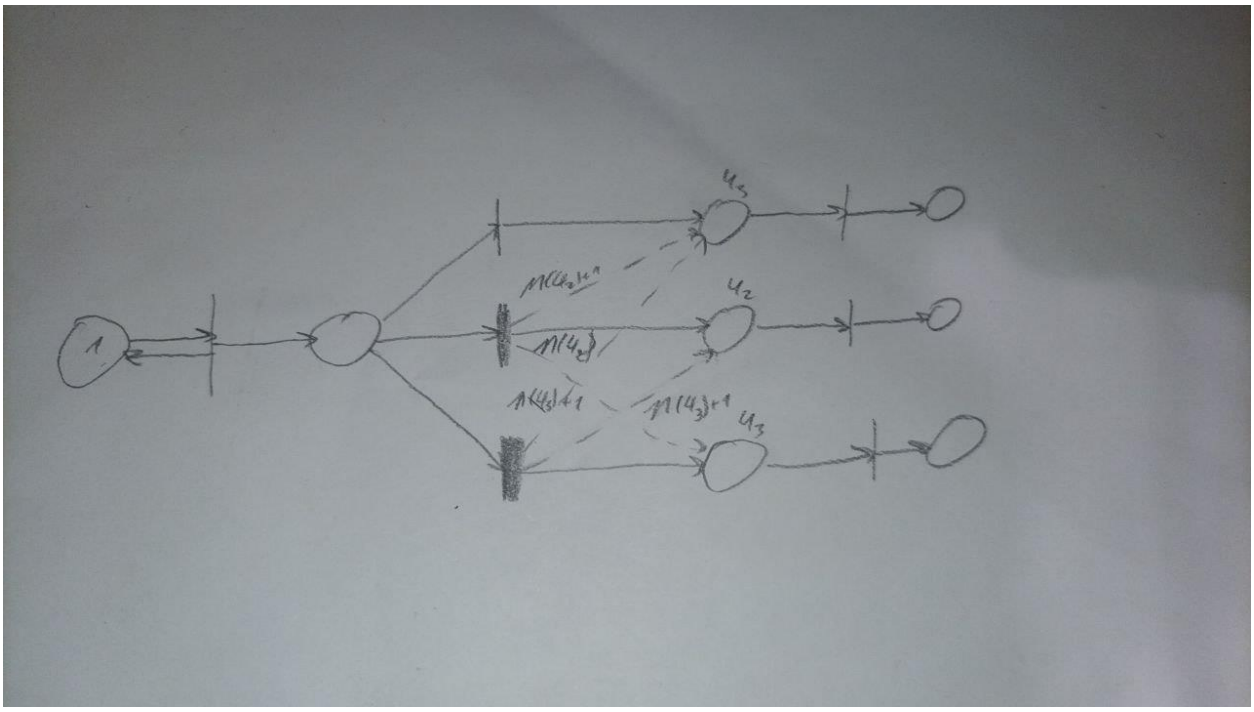
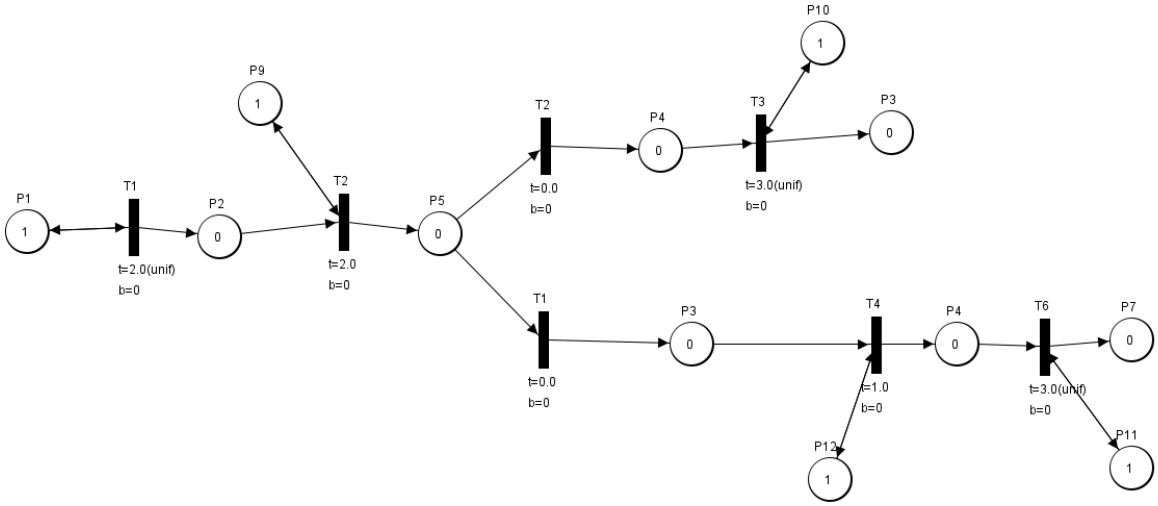
3.



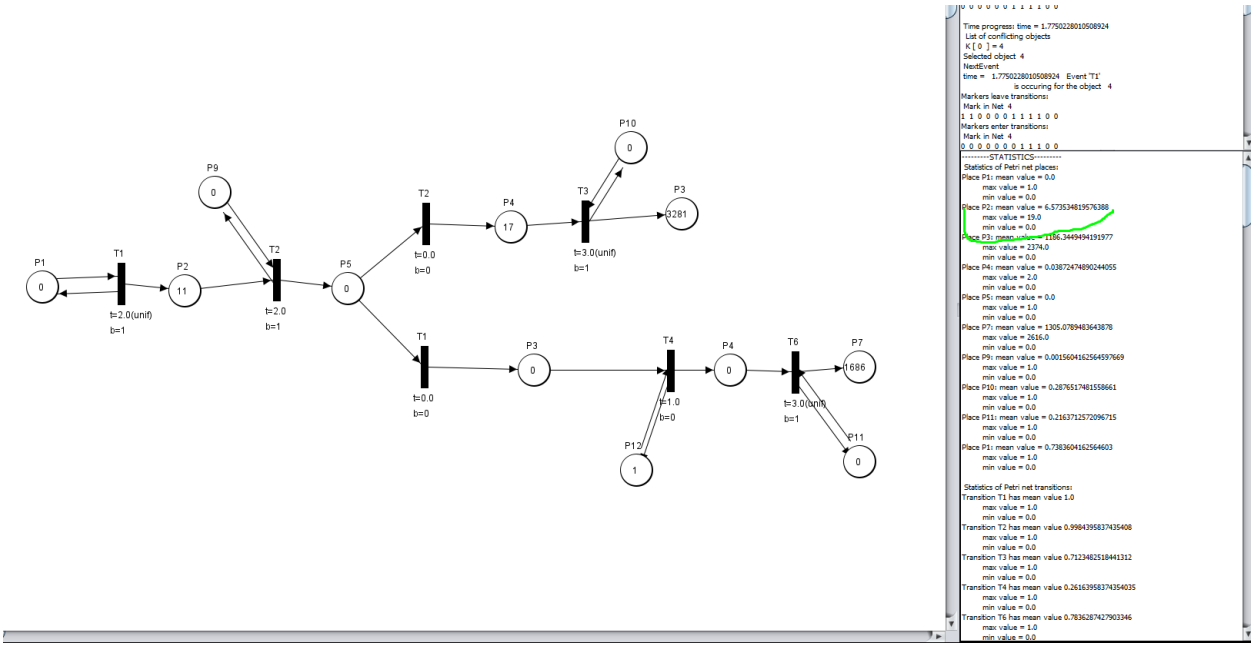
Якщо в чергах 1 і 2 більше ніж в черзі 3 ідемо в 3, якщо в 1 більше ніж 2 а в 3 стільки ж або більше ідемо в 2, інакше в 1

4.

Мережа:



Результат:



Вибір мінімального часу:

```
3 usages  2 overrides  innastetsenko
public void go(double timeModeling) {
    double min;
    this.setSimulationTime(timeModeling);
    this.setCurrentTime(0.0);

    getListObj().sort(PetriSim.getComparatorByPriority()); //edited 9.11.2015, 12.10.2017
    for (PetriSim e : getListObj()) { //edited 9.11.2015, 18.07.2018
        e.input();
    }
    if (isProtocolPrint() == true) {
        for (PetriSim e : getListObj()) {
            e.printMark();
        }
    }
    ArrayList<PetriSim> conflictObj = new ArrayList<>();
    Random r = new Random();

    while (this.getCurrentTime() < this.getSimulationTime()) { // edited 18.07.2018

        conflictObj.clear();

        min = getListObj().get(0).getTimeMin();  //пошук найближчої події

        for (PetriSim e : getListObj()) {
            if (e.getTimeMin() < min) {
                min = e.getTimeMin();
            }
        }
        /*  if(min_t<t){ // added 24.06.2013   !!!!Подумать...при отрицательных задержках висит!!!!
         JOptionPane.showMessageDialog(null, "Negative time delay was generated! Check parameters, please/");
         return;
```

Вибір об’єктів:

```
if (this.getCurrentTime() <= this.getSimulationTime()) {

    for (PetriSim sim : getListObj()) {
        if (this.getCurrentTime() == sim.getTimeMin()) // розв'язання конфлікту об'єктів рівноймовірнісним способом
        {
            conflictObj.add(sim);                          //список конфліктних обєктів
        }
    }
    int num;
    int max;
    if (isProtocolPrint() == true) {
        System.out.println(" List of conflicting objects  " + "\n");
        for (int ii = 0; ii < conflictObj.size(); ii++) {
            System.out.println(" K [ " + ii + "  ] = " + conflictObj.get(ii).getName() + "\n");
        }
    }

    if (conflictObj.size() > 1) { //вибір об'єкта, що запускається
        max = conflictObj.size();
        conflictObj.sort(PetriSim.getComparatorByPriority());
        for (int i = 1; i < conflictObj.size(); i++) { //System.out.println("  "+conflictObj.get(i).getPriority()+"  "+conflictObj.get(i-1).getPriority());
            if (conflictObj.get(i).getPriority() < conflictObj.get(i - 1).getPriority()) {
                max = i - 1;
                //System.out.println("max=  "+max);
                break;
            }

        }
        if (max == 0) {
            num = 0;
        } else {
            num = r.nextInt(max);
        }
    } else {
        num = 0;
    }
```

Виконання переходів:

```
for (PetriSim sim: getListObj()) {
    if (sim.getNumObj() == conflictObj.get(num).getNumObj()) {
        if (isProtocolPrint() == true) {
            System.out.println(" time =   " + this.getCurrentTime() + "   Event '" + sim.getEventMin().getName() + "'\n"
                    + "                        is occuring for the object   " + sim.getName() + "\n");
        }
        sim.doT();
        sim.output(); // added by Inna 11.07.2018
    }
}
if (isProtocolPrint() == true) {
    System.out.println("Markers output:");
    for (PetriSim sim : getListObj()) //ДРУК поточного маркірування
    {
        sim.printMark();
    }
}

Collections.shuffle(getListObj()); // added by Inna 11.07.2018, need for correct functioning of Petri object's shared resource

getListObj().sort(PetriSim.getComparatorByPriority());

for (PetriSim e : getListObj()) {
    //можливо змінились умови для інших обєктів
    e.input(); //вхід маркерів в переходи Петрі-об'єкта

}
if (isProtocolPrint() == true) {
    System.out.println("Markers input:");
    for (PetriSim e : getListObj()){ //ДРУК поточного маркірування
          e.printMark();
    }
}
```

Збір статистики:

```java
/**
 *
 * @param dt - the time interval
 */
6 usages  innastetsenko
public void doStatistics(double dt) {
    if (dt > 0) {
        for (PetriP position : listPositionsForStatistica) {
            position.changeMean(dt);
        }
    }
    if (dt > 0) {
        for (PetriT transition : listT) {
            transition.changeMean(dt);
        }
    }
}
```

```java
/**
 * /**
 * Recalculates the mean value
 *
 * @param a value for recalculate of mean value (value equals product of
 * marking and time divided by time modeling)
 */
6 usages  innastetsenko
public void changeMean(double a) {
    mean = mean + (mark - mean) * a;
}
```