Dokumentacja programu „Kasa PKP”

Wersja 1.0

Kamil Niesłuchowski,
2015

Spis treści

1. Wprowadzenie

Program „Kasa PKP” powstał na potrzeby zaliczenia przedmiotu „Programowanie matematyczne z elementami teorii gier” na Uniwersytecie Kardynała Stefana Wyszyńskiego w Warszawie. Autorem, który odpowiada za jego stworzenie jest Kamil Niesłuchowski.

1.1 Sposób działania

Pogram służy do obliczania minimalnej liczby czynnych kas. Aby tego dokonać należy uzupełnić wszystkie cztery pola interesującymi nas danymi, a następnie użyć przycisku „Oblicz” w celu wyświetlenia wyniku. Objaśnienia danych programu można sprawdzić pod przyciskiem „Legenda”.

1.2 Konfiguracja programu

Program nie wymaga konfiguracji. Po uruchomieniu gotowy jest do pracy.

1.3 Wymagania sprzętowe

Do prawidłowej pracy niezbędny jest komputer klasy PC, który spełnia następujące wymagania:

- 512 MB RAM
- PROCESOR PENTIUM 4 2,4 GHZ
- ok. 1 MB wolnego miejsca na dysku twardym
- Monitor SVGA (1024 x 768)

1.4 Wymagane oprogramowanie

Poza wymaganiami wymienionymi w punkcie 1.3., niezbędne do działania programu jest oprogramowanie wymienione poniżej:

- Dowolny system operacyjny z zainstalowaną maszyną wirtualną Javy w wersji JRE 7+

2. Instalacja programu

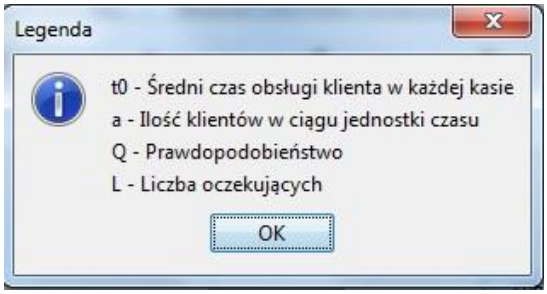
Aby zacząć używać programu wystarczy uruchomić plik o nazwie „kasaPKP.jar” bez zbędnych instalacji.

3. Wygląd oraz opis interfejsu

Rys. 1 – Wygląd interfejsu programu

Opis interfejsu:

1) Pole tekstowe „t0”, czyli średni czas obsługi klienta w każdej kasie
2) Pole tekstowe „a”, czyli ilość klientów w ciągu jednostki czasu
3) Pole tekstowe „Q”, czyli prawdopodobieństwo
4) Pole tekstowe „L”, czyli liczba oczekujących
5) Przycisk „Oblicz” służący do wykonywania operacji matematycznych w celu obliczenia wyniku dla określonych danych
6) Pole tekstowe „Wynik” przechowujące wynik ostatnich operacji matematycznych
7) Przycisk „Legenda” po wciśnięciu, którego wyświetlana jest informacja o danych programu.



Rys. 2 – Wygląd legendy

4. Opis problemu

Do sali kasowej PKP przybywa średnio a klientów w ciągu jednostki czasu. Średni czas obsługi klienta w każdej kasie jest równy t0. Zakładając, że strumień wejściowy klientów jest najprostszy, a czas obsługi ma rozkład wykładniczy, opracować algorytm wyznaczenia minimalnej liczby czynnych kas n, aby z danym z góry prawdopodobieństwem Q w dowolnej chwili czasu liczba oczekujących klientów była większa niż L osób. Zakładamy, że w każdej kasie sprzedawane są bilety na pociągi wszystkich rodzajów i we wszystkich kierunkach, tj. do wszystkich kas mamy jedną wspólną kolejkę.

5. Algorytm

5.1 Funkcja licząca silnie

```
public static double silnia(double n) {
    return (n<2) ? 1.0 : n*silnia(n-1);
}
```

5.2 Funkcja licząca P0

```
public static double obliczP0(double n, double ro) {
    double sum = 0.0;
    for (double k=0; k<=n; k++) {
        double podsum = 1;

        for (double i=1; i<=k; i++)
            podsum = podsum * (n * ro / i);
        sum += podsum;
    }

    sum += (Math.pow(n,n) * Math.pow(ro, n+1)) / (silnia(n) * (1-ro));

    return 1.0 / sum;
}
```

5.3 Funkcja licząca Pk

```
public static double obliczPk(double n, double k, double ro, double p0) {
    double p = 1;

    if (k<=n) {
        for (double i=1; i<=k; i++)
            p = p * (n * ro / i);

        return p * p0;
    }
    else {
        for (double i=1; i<=k; i++)
            p = p * (n * ro / i);

        for (double i=k+1.0; i<=n; i++)
            p = p * (n / i);

        return p*p0;
    }
}
```

5.4 Główna funkcja programu

```
public static String model() {
    double t = Double.parseDouble(input1.getText());
    double lambda = Double.parseDouble(input2.getText());
    double Q = Double.parseDouble(input3.getText());
    int L = Integer.parseInt(input4.getText());

    double n = obliczMinimalneN(t, lambda);
    double ro = (lambda / (n * (1 / t)));

    double suma = 0.0;
    n--;
    do {
        n++;
        ro = (lambda / (n * (1 / t)));
        suma = obliczPstwoCalkowite(n, ro, L);
    } while (suma < Q);

    return Integer.toString((int) n);
}
```