KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

DUOMENŲ STRUKTŪROS (P175B014)

I labaratorinio darbo ataskaita

Atliko:
IFF 7/12grupės stud.
Paulius Ratkevičius

Priėmė
lekt. —-

KAUNAS, 2018

TURINYS

1. PIRMO KURSO LABORATORINIO UŽDUOTIES PROGRAMA

1.1. LABORATORINIO DARBO UŽDUOTIS

Programa priima du sąrašus, vieną su gyventojais ir kiek jie suvartoją paslaugų, kitą su paslaugų įkainais. Su šiais sarašais atliekamos šios užduotys:

1. Surikiuoti vartotojuis pagal vardus.
2. Išspausdinti pąslaugas, gyventojus ir kiek jie turi sumokėti.
3. Išspausdinti gyventojus kurie sumokėjo mažiau negu vidurkis.
4. Išspausdinti kuri paslauga buvo pigiausia kurį mėnesį (neatsižvelgiant į paslaugas kurios buvo nenaudojamos).

1.2. SPRENDIMAS

```
public class Taxes //Atlieka visus skaiciavimus nurodytus uzduotyje.
{
    private UtilityContainer   utilities  = new UtilityContainer();
    private ResidentContainer residents = new ResidentContainer();

    private void run(String[] args) //Metodas kuris priima paduotus argumentus, juos nuskaito ir
        pradeda skaiciuoti.
    private void readFiles(String[] fileNames) //Nukreipia skaitymus i readUtilFile ir
        readResidentFile
    private void readUtilFile(String fileName) //Nuskaito paslaugu duomenu faila
    private void readResidentFile(String fileName) //Nuskaito gyventoju duomenu faila
    private void mapUtilities( Utility [] arr) // Sudedamos paslaugos i UtilityContainer
    private void mapResidents(Resident[] arr) // Sudedami gyventoju duomenys i ResidentContainer
    private long getAverageMoneySpent() // Grazina vidutines gyventoju islaidas
```

```
    private long getTotalMoneySpent() // Grazina visu gyventoju islaidas
    private MonthUtilityPair getCheapestUtility() // Suranda pigiausia paslauga ir issaugo kuri
        m[U+FFFD]nesi
    private static class MonthUtilityPair{ // Konteineris kuris laiko: paslauga ir jos menesi
    int month;
    Utility utilPaidFor;
    }
    private static class MonthUtilitySummator
            implements ResidentFunc, PaymentFunc{

            public MonthUtilitySummator(UtilityContainer utils)
            public MonthUtilityPair getCheapestMonthUtil()
            public void run(Resident res)
            public void run(Payment pmt)
            private static class MonthlySum{
                private int [] arr
                public void add(int month, int amount)
                public MonthAmountPaidPair getMin()
                public static class MonthAmountPaidPair
            }
        }
    }
    private static class ResidentPayPredicate implements Predicate<Resident>
    private static class ResidentSummator implements ResidentFunc
    private static class PaymentSummator implements PaymentFunc
}
class ResidentContainer{
    private HashMap<String, Resident> resByLastName = new HashMap<>();
    private HashMap<String, Resident> resByAdress = new HashMap<>();
    private ArrayList<Resident> residents = new ArrayList<>();
```

```
    public void add(Resident res)
    public Resident getByLastName(String lastName)
    public Resident getByAdress(String adress)
    public void forEach(ResidentFunc func)
    public void sort(Comparator<Resident> cmp)
    public ArrayList<Resident> filter(Predicate<Resident> pr)
    public int size()
}

class UtilityContainer{
    private HashMap<Integer, Utility> utilByCode = new HashMap<>();
    private HashMap<String, Utility> utilByName = new HashMap<>();
    private ArrayList<Utility> util = new ArrayList<>();

    public void add(Utility util)
    public Utility getByCode(int code)
    public ArrayList<Utility> getByPrice(int price)
    public Utility getByName(String name)
    public Utility [] getUniqueUtils()
}
```

1.3. PRADINIS DUOMENŲ RINKINYS

```
utils :
    69, Hookers, 50;
    220, Electricity , 74;
    1944, Gas, 84;
    47, Hitman, 1500;
    42, Therapy, 500;
```

```
res :
    "Don Morello", "Hoboken",
        1,  42,     1,
        1,  1944,   10,
        1,  47,  2,
        1,  69,  2,
        1,  220,    5,
        2,  1944,   11,
        2,  220,    4,
        2,  47,  3,
        2,  69,  1;
    "Mr. Salieri", "Little Italy",
        1,  220,    15,
        1,  1944,   5,
        1,  47,  5,
        1,  42,  1,
        2,  69,  2,
        2,  220,    50,
        2,  47,  10,
        2,  42,  0;
    Tommy, "Little Italy",
        1,  220,    1,
        1,  1944,   1,
        1,  69,  1,
        2,  1944,   2,
        2,  69,  1,
        2,  47,  1;
```

1.4. REZULTATAI

Don Morello

Therapy	1	1	500
Gas	1	10	840
Hitman	1	2	3000
Hookers	1	2	100
Electricity	1	5	370
Gas	2	11	924
Electricity	2	4	296
Hitman	2	3	4500
Hookers	2	1	50

Mr. Salieri

Electricity	1	15	1110
Gas	1	5	420
Hitman	1	5	7500
Therapy	1	1	500
Hookers	2	2	100
Electricity	2	50	3700
Hitman	2	10	15000
Therapy	2	0	0

Tommy

Electricity	1	1	74
Gas	1	1	84
Hookers	1	1	50
Gas	2	2	168
Hookers	2	1	50
Hitman	2	1	1500

Residents who paid less than average:

Don Morello

Tommy

Cheapest utility :

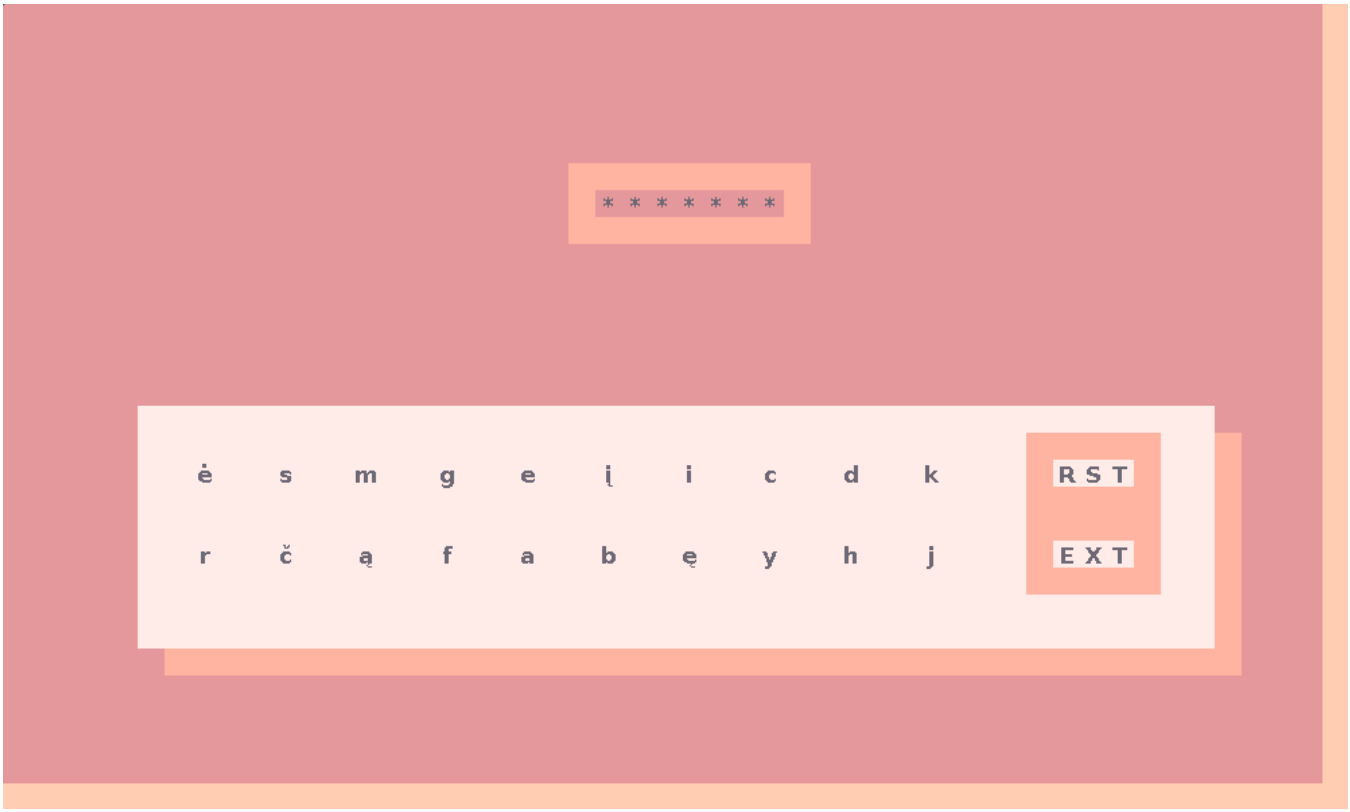
Name:Hookers

Month: 1

2. VIZUALINĖS STRUKTŪROS

2.1. UŽDUOTIS

Žaidimas kuris parenka astitiktinį žodį iš vikižodyno ir su juom sugeneruoja kartuves.



2.1 pav. Programos Nuotrauka

2.2. VARTOTOJO INSTRUKCIJA

Vartotojas pasileidžia Lab1PartB.Lab1B programą ir pamačius titulinį ekraną spaudžia pradėti. Vartotojas esantis žaidime gali rinktis raides iš duotos klaviatūros, sugeneruoti naują žodį

arba išeiti iš programos.

2.3. SPRENDIMAS

```
public class Lab1B{
    public static String RandomWord() // Grazina atsitiktini zodi is lt.vikipedija.org

}

public class graphics extends ScreenKTU{



    public void IntroScreenButton() // Sugeneruoja tituline sasaja.
    public void Start() // Pradeda zaidima.
    public String FindNewWord() // Suranda atsitiktini zodi kuris atitinka reikalavimus.
    public String GenerateKeyboard(int MaxKeys, String zodis) // Sugeneruoja klaviatura kuri turi
        raides is isrinkto zodzio ir likusias vietas uzpildo atsitiktinemis raidemis kurios
        nesikartoja.
    public void PrintKeyboard(int MaxKeys, String keyboard) // isspausdina virtualia klaviatura ant
        ekrano.
    public void CheckIfCorrect(int x) // Patikrina ar pasirinkta raide yra teisinga
    public void PrintCorrect(int x) // Isspausdina teisinga raide jeigu ji yra aptikta.
    public void Win() // Sis pranesimas ivykdomas jeigu zaidejas laimi zaidima.
    public String ShuffleString(String target) // Ismaisomas duotas string.
}
```

3. SĄRAŠINIŲ DUOMENŲ STRUKTŪRŲ KŪRIMAS

3.1. UŽDUOTIS

Realizuoti šiuos metodus:

1. add(int k, Data data)
2. set(int k, Data data)
3. remove(int k)
4. void addLast(E e)
5. boolean removeLastOccurance(Object o)
6. List<E> subList(int fromIndex, int toIndex)

Realizuoti greitaveikos testus su šiais metodais:

1. Math.sqrt(x) <-> Math.cbrt(x)
2. ArrayList<Integer> <-> LinkedList<Integer> metodas get(i)

3.2. REALIZUOTI METODAI

```
public boolean add(E e){
    if (e == null){
        return false;
    }
    if ( first == null){
        first = new Node<>(e, null);
        last = first;
    }
    else{
```

```
        Node<E> e1 = new Node(e, null);
        last.next = e1;
        last = e1;
        }
        size ++;
        return true;
    }
    public E set(int k, E e) {
        if(k < 0 || k >= size)
        return null;
        if(k == size-1)
            last = new Node<>(e, null);
        if(k == 0)
            first = new Node<>(e, first.next);
        current = first.findNode(k-1);
        E temp = current.next.element;
        current.next = new Node<>(e, current.next.next);
        return temp;

    }
    @Override
    public E remove(int k) {
        E temp;
        if(k < 0 || k >= size)
            return null;
        if(k == size - 1)
        {
            current = first.findNode(k-1);
            temp = last.element;
            current = new Node<>(current.element, null);
            last = current;
```

```
            size--;
            return temp;
        }
        if(k == 0)
        {
            temp = first.element;
            first = first.next;
            size--;
            return temp;
        }
        current = first.findNode(k-1);
        temp = current.next.element;
        current = new Node<>(current.element, current.next.next);
        size--;
        return temp;
    }
    public boolean removeLastOccurance(Object o)
    {
        if( first == null)
            return false;
        if( first.element == o || first.next == null)
        {
            first = null;
            size--;
            return true;

        }
        Node<E> temp = null;
        current = first;
        if(current.element == o)
            temp = current;
```

```
    while(current.next != null)
    {
        if (current.next.element == o)
            temp = current.next;
        current = current.next;
    }
    if (temp == first)
    {
        first = first.next;
        size--;
        return true;
    }
    if (temp == null)
        return false;
    current.next = current.next.next;
    size--;
    return true;
}
public ListKTU<E> subList(int fromIndex, int toIndex)
{
    ListKTU<E> output = new ListKTU<E>();
    current = first.findNode(fromIndex);
    for (int i=0; i<toIndex; i++)
    {
        if (current.element != null)
            output.add(current.element);
        current = current.next;
    }
    return output;
}
```

3.3. GREITAVEIKOS TIKRINIMAI

It took 15275 nanoseconds to perform 10 random retreivals from ArrayList.

It took 2205019 nanoseconds to perform the same operation with a linked list.

It took 269222 nanoseconds to pull 10 cube roots with cbrt.

It took 8540 nanoseconds to do sqrt(x)

It took 15552 nanoseconds to perform 10 random retreivals from ArrayList.

It took 977831 nanoseconds to perform the same operation with a linked list.

It took 8591 nanoseconds to pull 10 cube roots with cbrt.

It took 3493 nanoseconds to do sqrt(x)

It took 15767 nanoseconds to perform 10 random retreivals from ArrayList.

It took 1216832 nanoseconds to perform the same operation with a linked list.

It took 9302 nanoseconds to pull 10 cube roots with cbrt.

It took 3154 nanoseconds to do sqrt(x)

It took 11559 nanoseconds to perform 10 random retreivals from ArrayList.

It took 398558 nanoseconds to perform the same operation with a linked list.

It took 6112 nanoseconds to pull 10 cube roots with cbrt.

It took 2499 nanoseconds to do sqrt(x)

It took 13616 nanoseconds to perform 10 random retreivals from ArrayList.

It took 534775 nanoseconds to perform the same operation with a linked list.

It took 8931 nanoseconds to pull 10 cube roots with cbrt.

It took 3909 nanoseconds to do sqrt(x)