ЛЕКЦИЯ I

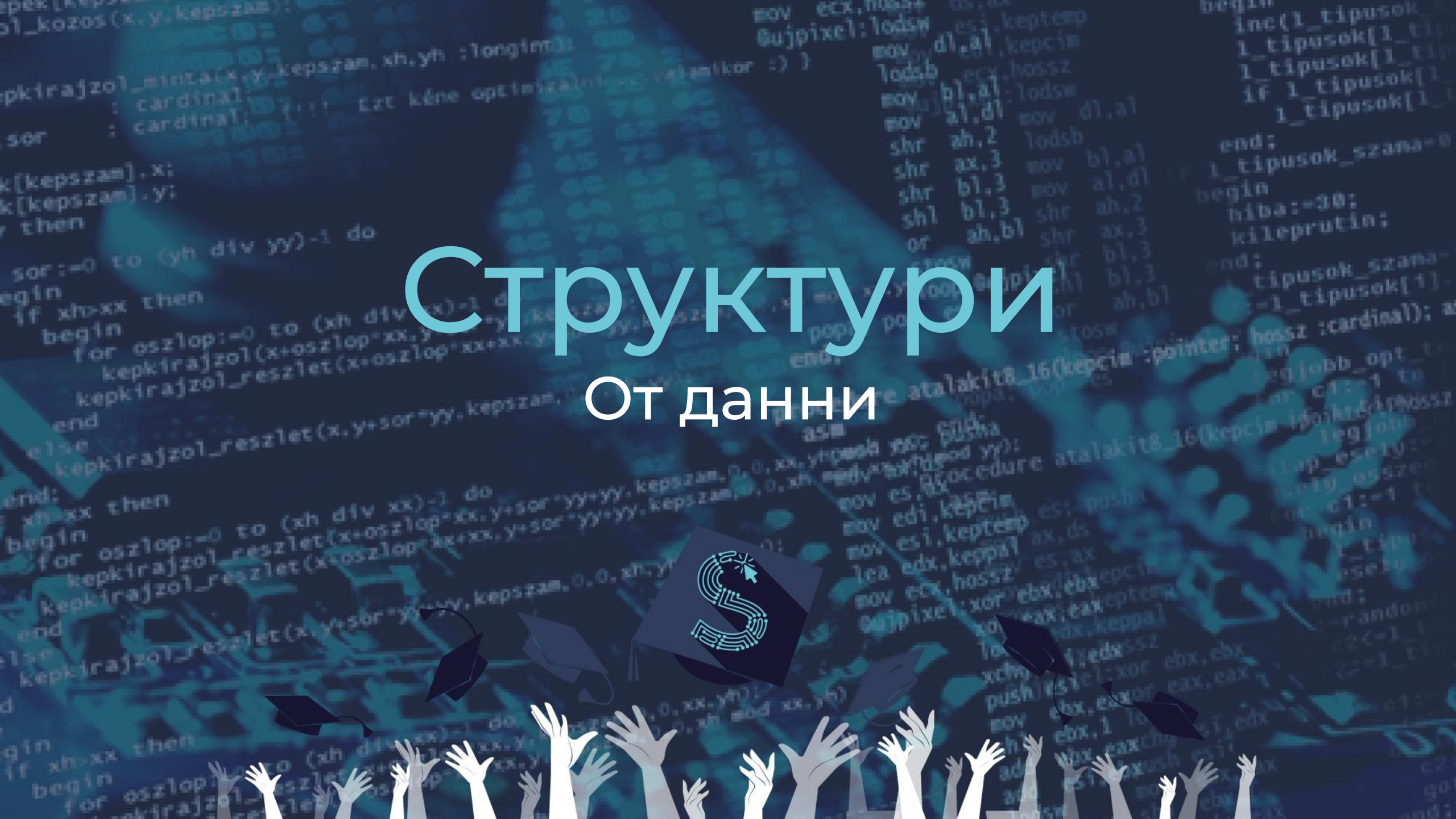
JAVA OOP:
Основни структури

Теодор Костадинов

Съдържание

Структури

От данни

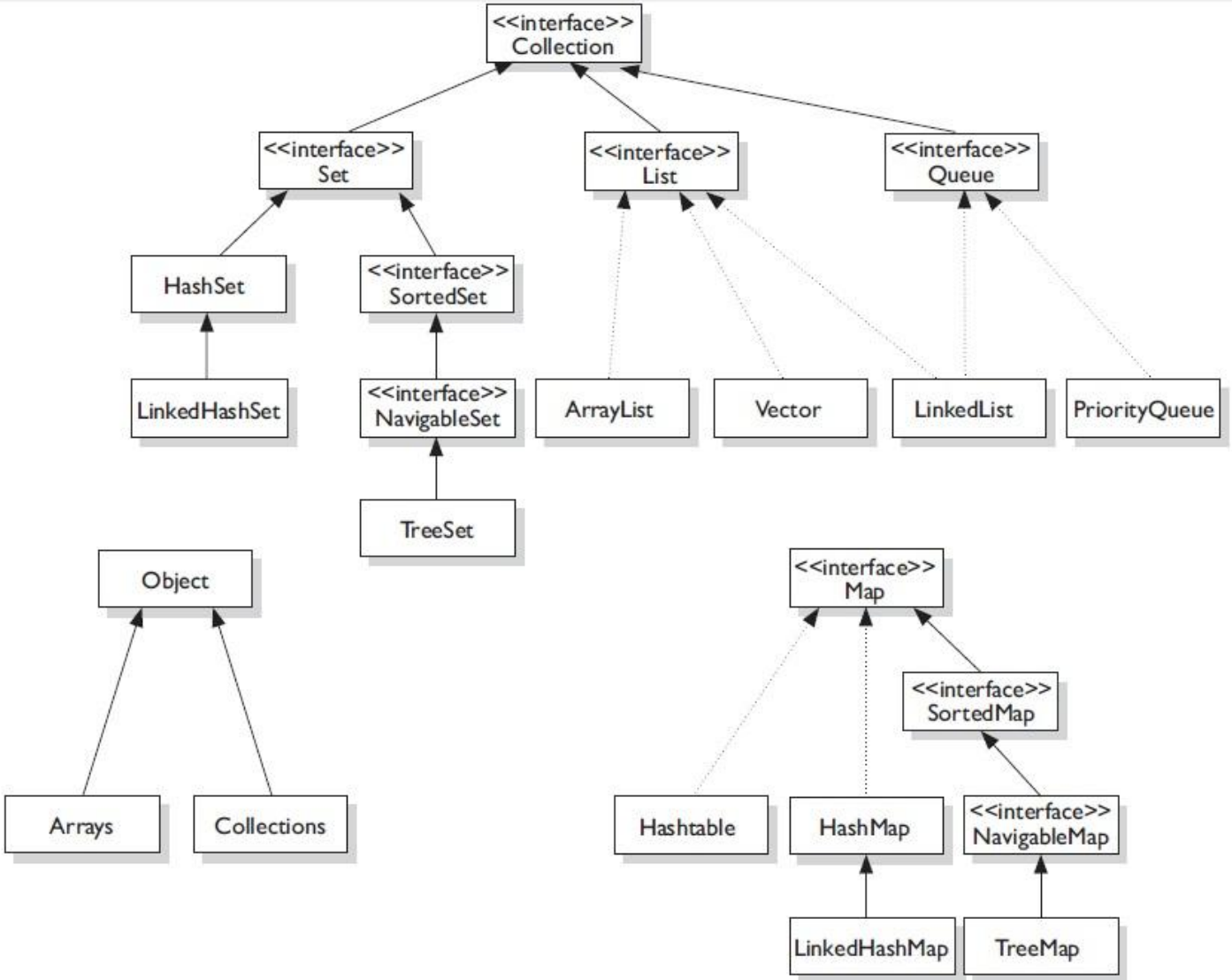


Структури от данни

- В зависимост от задачата, която трябва да решим с програмиране, се налага да организираме данните, с които работим, по различен начин (например подредба на някакви елементи или връзки между тях)
- Структурите от данни са множество от данни, организирани по определен начин.

- Има множество готови структури от данни, които можем да използваме на готово.
- При повечето структури, трябва да отбележим типа на данните, които ще съхраняваме в < > скоби
- Пример: List<String> - списък от стрингове

Йерархия на структурите



Йерархия на структурите

SOFTWARE

Interface Hierarchy

- java.util.**Comparator**<T>
- java.util.**Enumeration**<E>
- java.util.**EventListener**
- java.util.**Formattable**
- java.lang.**Iterable**<T>
  - java.util.**Collection**<E>
    - java.util.**List**<E>
    - java.util.**Queue**<E>
      - java.util.**Deque**<E>
    - java.util.**Set**<E>
      - java.util.**SortedSet**<E>
        - java.util.**NavigableSet**<E>
- java.util.**Iterator**<E>
  - java.util.**ListIterator**<E>
  - java.util.**PrimitiveIterator**<T,T_CONS>
    - java.util.**PrimitiveIterator.OfDouble**
    - java.util.**PrimitiveIterator.OfInt**
    - java.util.**PrimitiveIterator.OfLong**
- java.util.**Map**<K,V>
  - java.util.**SortedMap**<K,V>
    - java.util.**NavigableMap**<K,V>
- java.util.**Map.Entry**<K,V>
- java.util.**Observer**
- java.util.**RandomAccess**
- java.util.**Spliterator**<T>
  - java.util.**Spliterator.OfPrimitive**<T,T_CONS,T_SPLITR>
    - java.util.**Spliterator.OfDouble**
    - java.util.**Spliterator.OfInt**
    - java.util.**Spliterator.OfLong**

Class Hierarchy

- java.lang.**Object**
  - java.util.**AbstractCollection**<E> (implements java.util.Collection<E>)
    - java.util.**AbstractList**<E> (implements java.util.List<E>)
      - java.util.**AbstractSequentialList**<E>
        - java.util.**LinkedList**<E> (implements java.lang.Cloneable, java.util.Deque<E>, java.util.List<E>, java
      - java.util.**ArrayList**<E> (implements java.lang.Cloneable, java.util.List<E>, java.util.RandomAccess, java.io
      - java.util.**Vector**<E> (implements java.lang.Cloneable, java.util.List<E>, java.util.RandomAccess, java.io.Se
        - java.util.**Stack**<E>
    - java.util.**AbstractQueue**<E> (implements java.util.Queue<E>)
      - java.util.**PriorityQueue**<E> (implements java.io.Serializable)
    - java.util.**AbstractSet**<E> (implements java.util.Set<E>)
      - java.util.**EnumSet**<E> (implements java.lang.Cloneable, java.io.Serializable)
      - java.util.**HashSet**<E> (implements java.lang.Cloneable, java.io.Serializable, java.util.Set<E>)
        - java.util.**LinkedHashSet**<E> (implements java.lang.Cloneable, java.io.Serializable, java.util.Set<E>)
      - java.util.**TreeSet**<E> (implements java.lang.Cloneable, java.util.NavigableSet<E>, java.io.Serializable)
    - java.util.**ArrayDeque**<E> (implements java.lang.Cloneable, java.util.Deque<E>, java.io.Serializable)
  - java.util.**AbstractMap**<K,V> (implements java.util.Map<K,V>)
    - java.util.**EnumMap**<K,V> (implements java.lang.Cloneable, java.io.Serializable)
    - java.util.**HashMap**<K,V> (implements java.lang.Cloneable, java.util.Map<K,V>, java.io.Serializable)
      - java.util.**LinkedHashMap**<K,V> (implements java.util.Map<K,V>)
    - java.util.**IdentityHashMap**<K,V> (implements java.lang.Cloneable, java.util.Map<K,V>, java.io.Serializable)
    - java.util.**TreeMap**<K,V> (implements java.lang.Cloneable, java.util.NavigableMap<K,V>, java.io.Serializable)
    - java.util.**WeakHashMap**<K,V> (implements java.util.Map<K,V>)
  - java.util.**AbstractMap.SimpleEntry**<K,V> (implements java.util.Map.Entry<K,V>, java.io.Serializable)
  - java.util.**AbstractMap.SimpleImmutableEntry**<K,V> (implements java.util.Map.Entry<K,V>, java.io.Serializable)
  - java.util.**Arrays**
  - java.util.**Base64**
  - java.util.**Base64.Decoder**
  - java.util.**Base64.Encoder**
  - java.util.**BitSet** (implements java.lang.Cloneable, java.io.Serializable)
  - java.util.**Calendar** (implements java.lang.Cloneable, java.lang.Comparable<T>, java.io.Serializable)
    - java.util.**GregorianCalendar**
  - java.util.**Calendar.Builder**
  - java.util.**Collections**
  - java.util.**Currency** (implements java.io.Serializable)
  - java.util.**Date** (implements java.lang.Cloneable, java.lang.Comparable<T>, java.io.Serializable)
  - java.util.**Dictionary**<K,V>
    - java.util.**Hashtable**<K,V> (implements java.lang.Cloneable, java.util.Map<K,V>, java.io.Serializable)
      - java.util.**Properties**
  - java.util.**DoubleSummaryStatistics** (implements java.util.function.DoubleConsumer)
  - java.util.**EventListenerProxy**<T> (implements java.util.EventListener)
  - java.util.**EventObject** (implements java.io.Serializable)
  - java.util.**FormattableFlags**

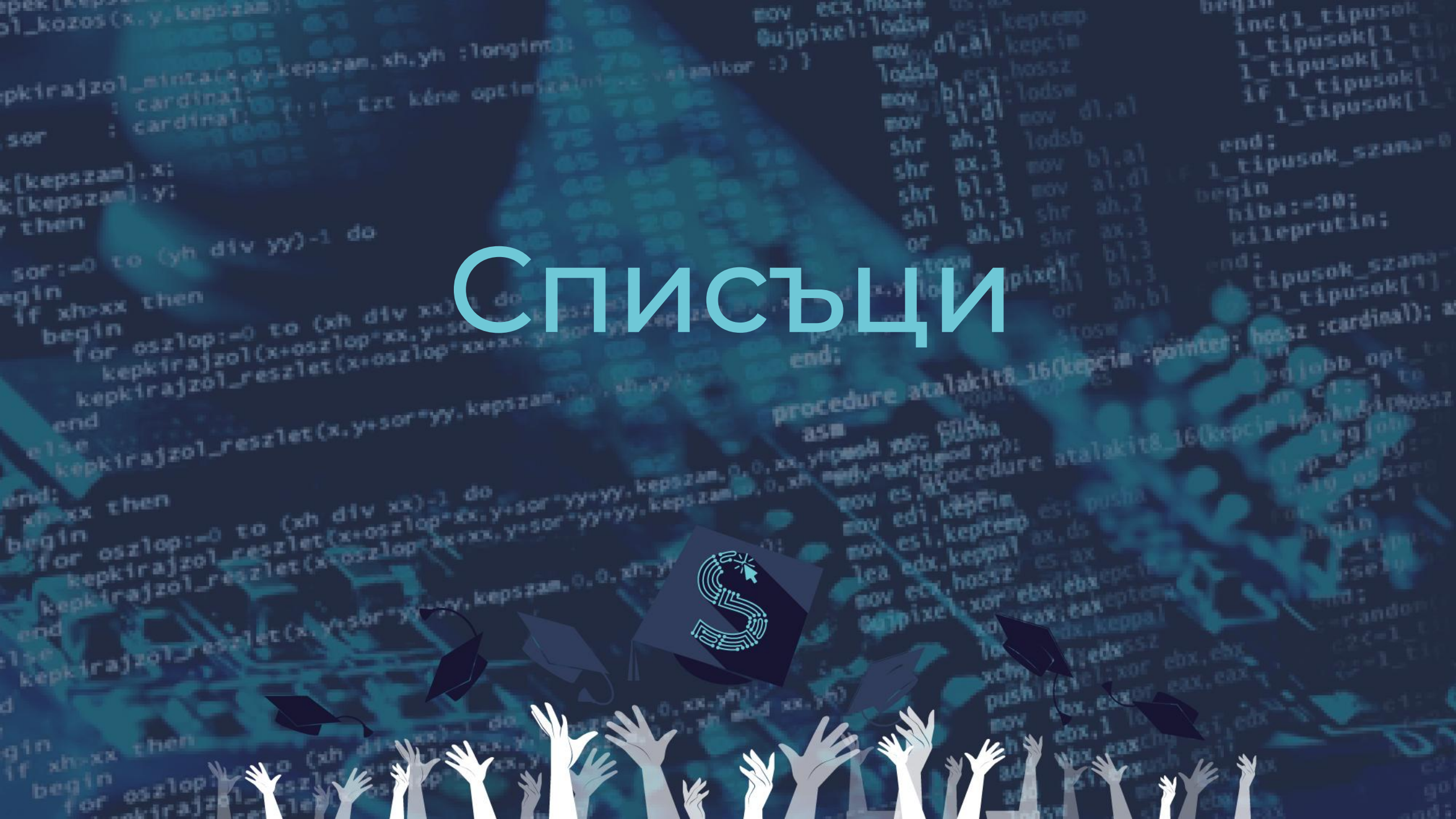
Йерархия на структурите

- От готовите структури от данни имате:
  - Интерфейси на структури - те дефинират общото поведение на различните видове структури
  - Класове на структури - те имплементират някой от интерфейсите, но самата имплементация може да е различна
- Например:
  - И двата класа HashMap и TreeMap имплементират Map интерфейса и са структури тип речник
  - Заради Map и двата класа имат методите: put, remove, clear, size..
  - НО
    - HashMap позволява ключове, които са null, TreeMap не
    - HashMap няма ред на елементите, TreeMap има
    - Добавяне и махане на елементи в HashMap има константна сложност, докато в TreeMap-а зависи от броя на елементите

Видове интерфейси на структури

- Collection - интерфейс, който представлява група от елементи. Предоставя методи add, remove, size и позволява да се извиква foreach на елементите.
  - Set - колекция без повтарящи се елементи
  - List - подредена група от елементи, при която потребителя знае номера на всеки елемент и може да го манипулира
  - Queue - подредена група от елементи, с конкретна логика за премахването на елемент (например FIFO)
- Map - група от елементи, при която всеки елемент има ключ и стойност, подобно на речник

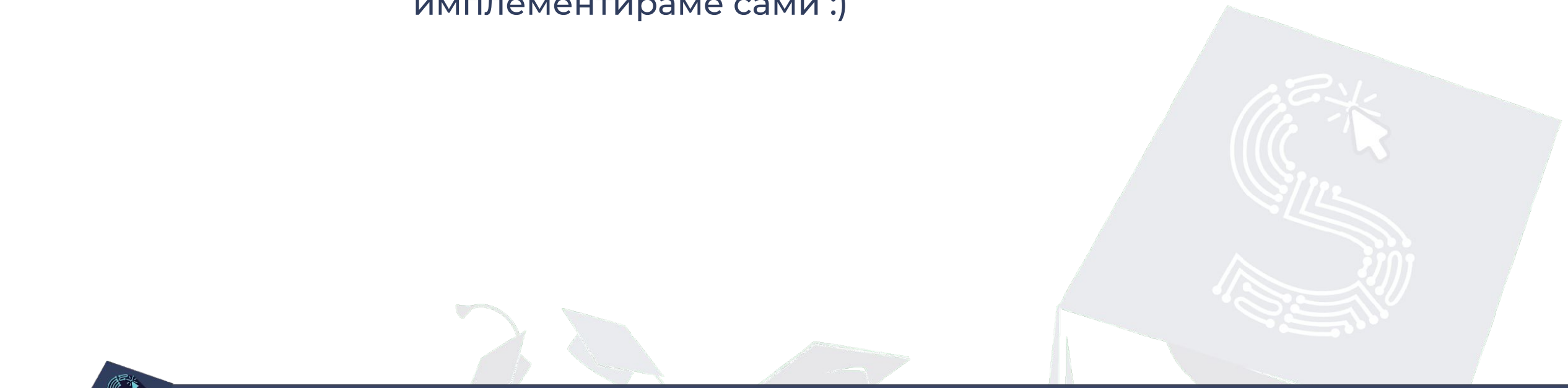
Списъци



Видове списъци

- **java.util.AbstractList<E> (implements java.util.List<E>)** - *абстрактен клас, който предоставя част от имплементацията на един списък. Тук за данните е използван масив.*
  - **java.util.AbstractSequentialList<E>** - *абстрактен клас, предоставящ част от имплементацията на списък. Тук за данните е използван "свързан списък"*
    - **java.util.LinkedList<E>** - *имплементация на двойно свързан списък*
  - **java.util.ArrayList<E>** - *модифицируем и автоматично разгъващ се списък*
  - **java.util.Vector<E>** - *синхронизирана версия на ArrayList. Друга разлика е, че при изчерпване на вътрешния масив, вектора създава нов с двен размер, докато ArrayList създава нов с 50% размера*
    - **java.util.Stack<E>** - *вектор, работещ на принципа last in, first out*

- За да разберем как точно работи един Списък, нека си го имплементираме сами :)



Ами сега?

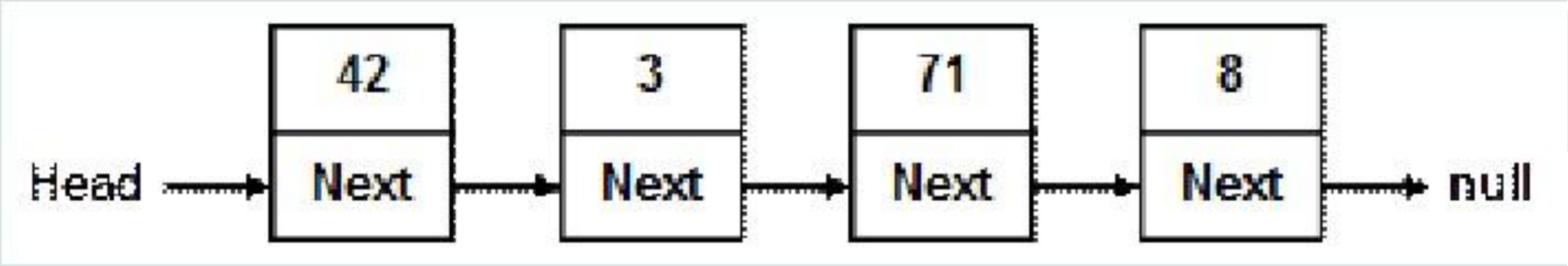
Задача за упражнение

Почивка

до 20:40

SOFTWARE ACADEMY
Educational Center Net It

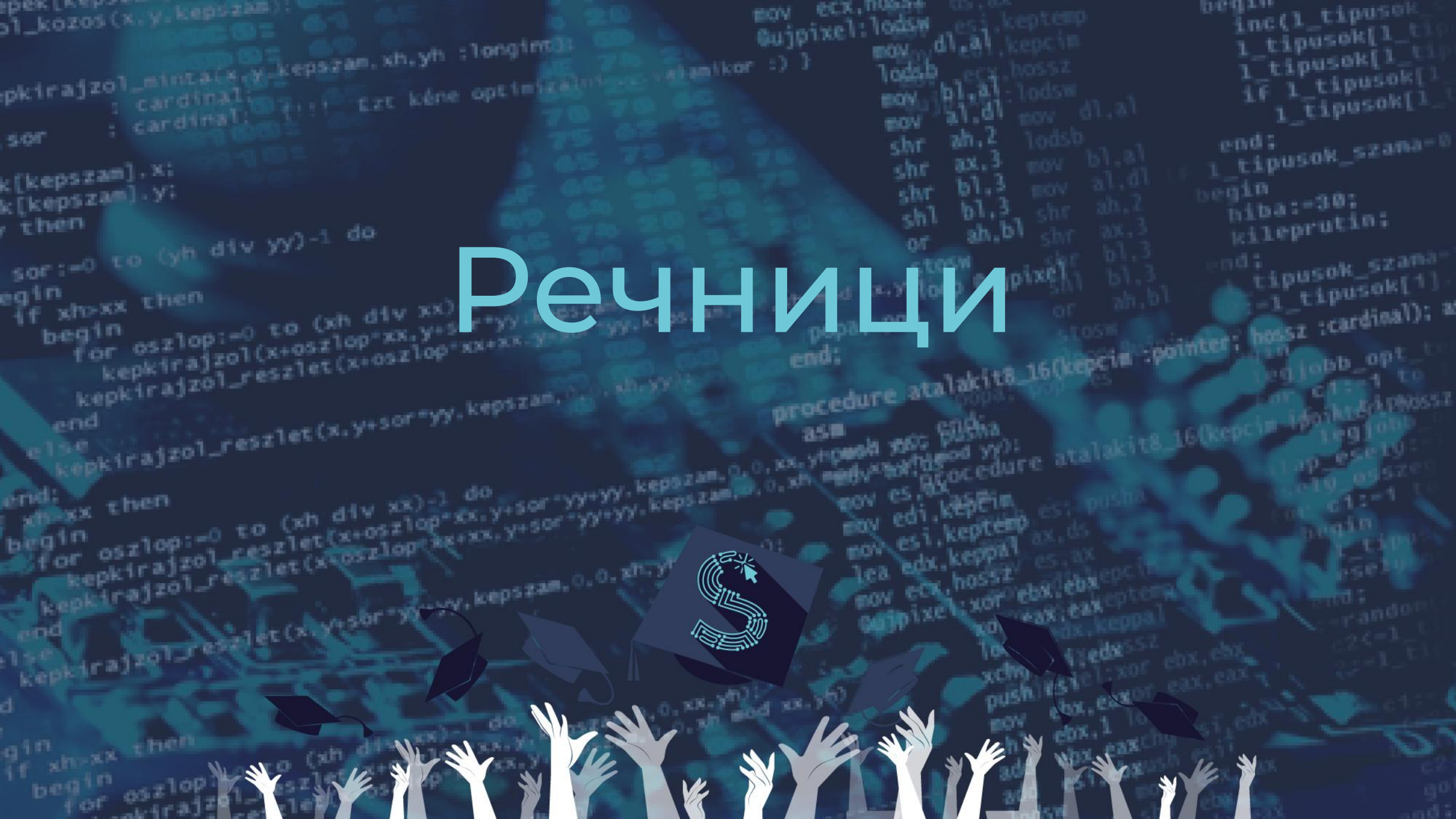
- За да разберем как точно работи един свързан списък, нека си го имплементираме сами :)



Ами сега?

Задача за упражнение

Речници



Речници

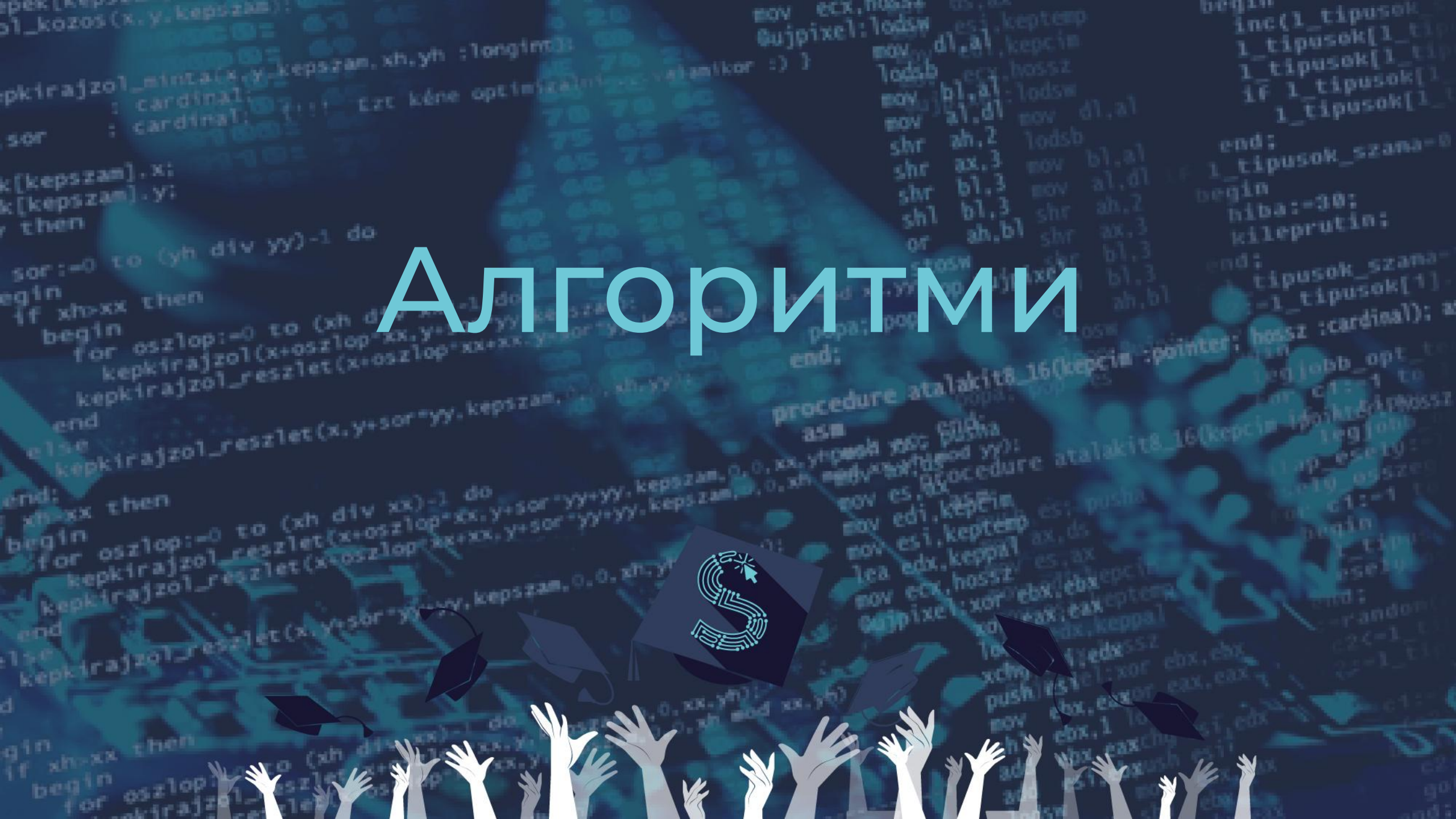
- Речниците позволяват вкарването на двойки данни
- Имплементират Map интерфейса
- Най-популярният клас за речник е HashMap
- На ключовете им се пресмятат хеш стойностите и се слагат в таблица
- Много бързо работи, ако хеш функцията е добра

- Да направим проста програма за Задачи (TODO)
  - Добавяме задача
  - Изтриваме задача
  - Достъпваме задача (за да можем да я променим, например)
  - Проверяваме дали списъкът е празен
  - Да сменяме местата на задачи
- Нека програмата да може да работи за различни потребители

Ами сега?

Задача за упражнение

Алгоритми



Готови алгоритми

- Когато искаме да извършваме операции върху колекциите и речниците, преди да реализираме наша операция, трябва да проверим дали няма готова такава
- Примери за вече реализирани алгоритми
  - сортиране
  - разбъркване
  - обръщане
  - копиране
  - размяна на елементи
  - добавяне на всички елементи
  - търсене
  - намиране на най-голяма и най-малка стойност

ВЪПРОСИ?

Резюме

- Има разнообразие от видове структури
- Трябва да знаем за какво ще ги използваме, за да изберем най-подходящата

Ресурси

SOFTWARE ACADEMY
Educational Center Net It

- Structure class hierarchy
- List Docs
- GitHub Repo with Demos
- Наков

Домашно

Качвайте домашното си в ГитХъб и слагайте линка тук:

https://forms.gle/AcvCptCbSDizr2Ay6

Задача 1

Създайте софтуер за администрация на болница.

- Болницата има лекари и пациенти.
- Лекарите си имат отделение и специализация, име, години, стаж.
- Пациентите имат лекуващ лекар, история на заболяванията, лекарствата и процедурите.
- Всяко заболяване на пациент има година на възникването.
- Всяко лекарство има име, доза, от кога до кога е приемано
- Всяка процедура има дата, име, описание и екип от лекари, които са я извършили.

Софтуерът трябва да позволява въвеждане на лекари и пациенти с всичките им данни.

Задача 2

Създайте структурата на електронен дневник.

Софтуерът може да се ползва от различни училища, всяко училище има много випуски и във всеки випуск има различни класове. Всеки клас има ученици, които получават оценки по различни предмети.

БЛАГОДАРЯ
ЗА ВНИМАНИЕТО!