Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

**Технологии разработки качественного**

**программного обеспечения**

Курсовой проект

Игра “Ассоциации”

|  |  |
| --- | --- |
| Выполнил  студент гр.в3530904/70321 | Т.Д. Кирсанов |
| Преподаватель | Н.Г. Смирнов |

Санкт-Петербург

2020

Оглавление

[1. Введение. Задание “Ассоциации” 2](#_Toc40692396)

[2 Пакеты 3](#_Toc40692397)

[3. Описание программы 8](#_Toc40692398)

[Выводы 10](#_Toc40692399)

[Список литературы 11](#_Toc40692400)

[Приложение 1: код программы 12](#_Toc40692401)

# 1. Введение. Задание “Ассоциации”

Словесная игра – необходимо угадать слово по набору ассоциаций (слова или фразы из ассоциативного словаря – русский или английский)

Статья ассоциативного словаря состоит из СЛОВА <ассоциация1>Х,<ассоциация2>Y…, где X и Y – чамтотность ассоциации (или её вес). Разработать правила игры, предусмотреть несколько уровней сложности.

Персистенция результатов игроков в файл + 10 баллов.

# 2 Пакеты

**Java IO**

**IO API** – (Input & Output) в первую очередь это Java API, которые облегчают работу с потоками.

Очень часто приходится получать какой-то поток данных, а потом как-то их обрабатывать и отправлять дальше. Например, пользователь ввел логин и пароль, программа в свою очередь должна получить эти данные, обработать и сохранить в файл. Для этих целей можно использовать IO.

В Java библиотека IO API находится в пакете **java.io**и для того чтобы начать использовать IO достаточно импортировать данную библиотеку в ваш класс.

**Input and Output – Назначение**

В **java.io**существуют так называемые потоки ввода и вывода (**InputStream and OutputStream**).

В основном java.io предназначен для чтения и записи данных в ресурс:

1) файл;

2) при работе с сетевым подключением;

3) System.err, System.in, System.out;

4) при работе с буфером.

В этом случае **Ресурс**связан с **InputStream /Reader,**а он в свою очередь связан с **программой** и через него программа получает поток данных.

**Программа** связанна с **OutputStream / Writer**, и с его помощью записываем данные которые отдала программа в **Destination.**

**Java IO – Способности**

В Java IO много классов которые в основном работают с потоками чтения и записи, и решают различные задачи:

– получения доступа к файлам;

– получение сетевого соединения;

– работа с буфером;

– доступ к внутринему буферу памяти;

– межпоточнное общение;

– парсинг данных;

– чтение и запись текста;

– чтение и запись примитивных данных (long, int, float …);

– чтение и запись объектов.

Все эти возможности вам предоставит Java IO.

**Классы Java IO API**

**Базовые**

– InputStream / OutputStream

– Reader / Writer

– InputStreamReader / OutputStreamWriter

**Массивы**

– ByteArrayInputStream / ByteArrayOutputStream

– CharArrayReader / CharArrayWriter

**Files**

– FileInputStream / FileOutputStream

– RandomAccessFile / RandomAccessFile

– FileReader / FileWriter

**Буферизация**

– BufferedInputStream / BufferedOutputStream

– BufferedReader / BufferedWriter

Это самые распространяемые по использованию классы, но в нашей серии уроков мы на примерах рассмотрим все классы Java IO API.

**Java NIO**

Пакет Java.nio был представлен в Java 1.4. В отличие от java I / O в java NIO введен поток данных, ориентированный на буфер и канал, для операций ввода / вывода, что в результате обеспечивает более быстрое выполнение и лучшую производительность.

Также NIO API предлагает селекторы, которые вводят функциональность прослушивания нескольких каналов для событий ввода-вывода асинхронным или неблокирующим способом. В NIO наиболее трудоемкие операции ввода-вывода, включая заполнение и опустошение буферов для операционной системы, которые увеличиваются в скорости.

Основные абстракции API-интерфейсов NIO следующие –

* Буферы, которые являются контейнерами для данных, наборов символов и связанных с ними декодеров и кодеров, которые преобразуются между байтами и символами Unicode.
* Каналы различных типов, которые представляют соединения с объектами, способными выполнять операции ввода-вывода
* Селекторы и клавиши выбора, которые вместе с выбираемыми каналами определяют мультиплексированный неблокирующий механизм ввода / вывода.

Буферы, которые являются контейнерами для данных, наборов символов и связанных с ними декодеров и кодеров, которые преобразуются между байтами и символами Unicode.

Каналы различных типов, которые представляют соединения с объектами, способными выполнять операции ввода-вывода

Селекторы и клавиши выбора, которые вместе с выбираемыми каналами определяют мультиплексированный неблокирующий механизм ввода / вывода.

**Java.util**

Это один из больших пакетов. В данном пакете содержится одно из новшеств Java 2 — коллекции. Коллекцией называется некоторый набор объектов, объединяемых с целью максимально эффективного их использования. Коллекции — достаточно сложная тема, однако упомянутый пакет имеет набор классов, не относящихся к коллекциям. Ниже приведено описание некоторых таких классов.

• *Класс Calendar*. Этот класс не содержит конструкторов. Есть множество констант, среди них — константы, содержащие текущую дату и время (они, соответственно, обозначаются английскими эквивалентами компонентов— например, DAY, HOUR и др.), названия месяцев (тоже по-английски), еще некоторые константы. Вот несколько методов этого класса:  
- after— значение true, если указанный в параметре Calendar-объект более "ранний", чем объект, который этот метод вызвал;  
- before — полностью противоположен предыдущему методу;  
- clone — копирует объект;  
- get — возвращает значение какой-либо константы;  
- get Ins tance — возвращает текущую дату в виде Calendar-объекта;  
- getTime — преобразует Calendar в Date;  
- getTimeZone — преобразует Calendar в TimeZone;  
- set — устанавливает значения даты и времени для объекта;  
- setTime — преобразует в Calendar Date-объект;  
- setTimeZone — преобразует в Calendar TimeZone-объскт.

• *Класс Date*. Имеет два конструктора: первый не имеет параметров и присваивает объекту текущую дату и время, второй устанавливает их с помощью задания в параметре количества миллисекунд, прошедших с 1.1.70. Имеет основные методы after, before, clone, getTime (возвращение миллисекунд) и setTime (установление миллисекунд). О методе toString мы не упоминаем, так как он присутствует почти у всех объектов.

• *Класс GregorianCalendar*. Расширяет класс Calendar. Нужен для установки даты и времени с учетом текущих часовым поясом и регионом. В конструкторах можно указывать дату и время в разных вариациях. Можно также указывать регион и часовой пояс в виде объектов Locale и TimeZone соответственно.

• *Класс Locale*. Нужен для определения языкового региона. Содержит несколько констант с названиями различных стран, например GERMANY и ITALY. В конструкторе можно установить язык и страну. С помощью статического метода setDefault можно установить регион по умолчанию. С помощью метода getDisplayCountry () можно получить название страны, а с помощью метода getDisplayLanguage () — язык.

• *Класс Random*. Отвечает за генерацию случайных чисел. Основные методы:  
- nextBoolean — случайное значение типа boolean;  
- nextDouble — то же самое для типа данных double;  
- nextFloat —то же для типа данных float;  
- nextInt — то же для типа данных int;  
- nextLong — то же для типа данных long.

• *Класс TimeZone*. Этот класс позволяет установить отклонение от времени по Гринвичу. Есть множество методов этого класса, я приведу лишь три из них:  
- getDefault — возвращает заданный по умолчанию часовой пояс;  
- getTimeZone — возвращает объект для параметра, в котором указывается  
имя часового пояса;  
- setDefault — устанавливает задаваемый по умолчанию часовой пояс.

# 3. Описание программы

Программа построена следующим образом. В неё входит 4 класса:

ReadDictionary – класс, предназначенный для чтения словаря из текстового файла “Русский Региональный Ассоциативный СловарьТезаурус.txt”, который будет использоваться для ассоциаций.

DictEntry - класс, который представляет из себя слово и понятия, ассоциированных с ним посредством списков. Может определять, в какой из списков занести слово и с каким весом. Вес ассоциаций хранится в виде HashMap.

GameClass – класс, в котором реализована непосредственно сама игра и где применяются методы, реализованные в предыдущих двух классах.

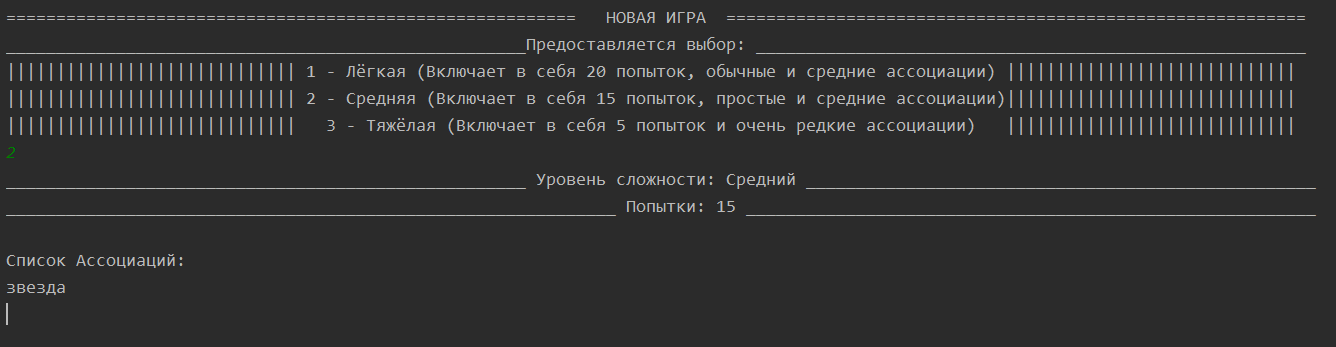
Main – основной класс с которого, собственно, и будет запускаться программа. Благодаря ему и сработают описанные выше классы и произойдёт окончательная компиляция проекта.

Работа программы:

Программа реализована в виде консольного приложения в среде разработки IntelijIDEA.

Как уже было сказано выше, через ReadDictionary будет осуществляться чтение словаря. Информация будет заносится в ArrayList через stream.

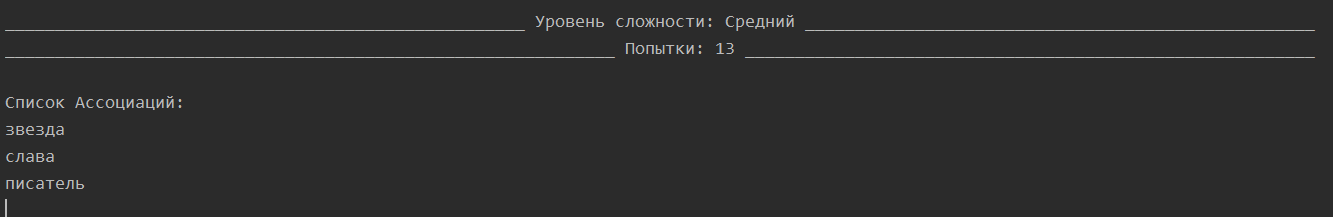
Через Main окончательно задаётся дирректория необходимого файла и запускается GameClass. Игра успешно запускается и ждёт дальнейших действий от игрока.



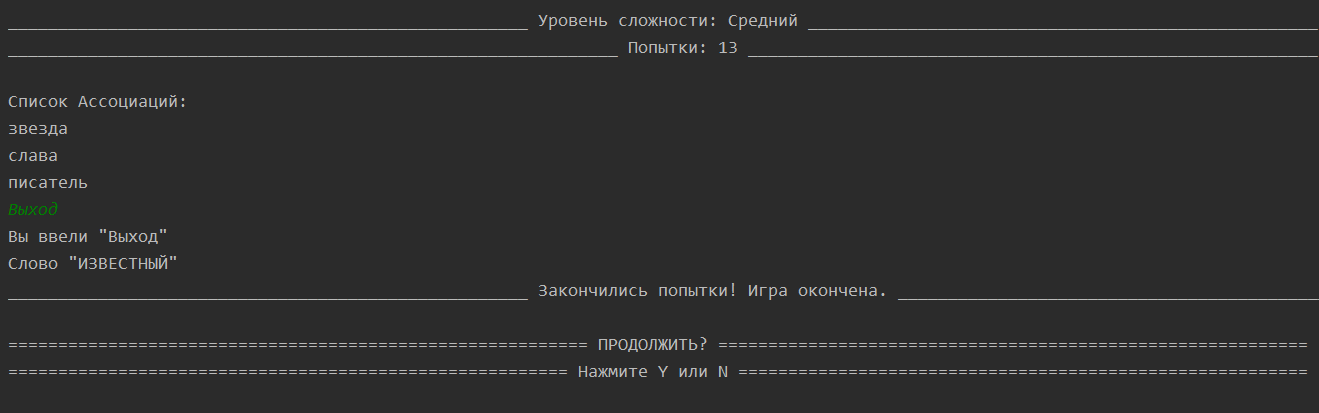
В игре реализована система сложности раундов. Система реализоавна через логические операции, выполнение которых зависит от конструкции switch во время выбора игроком сложности. В зависимости от выбора будет активироваться та или иная логическая операция, через которую будет идти обращение к методам класса DictEntry, в которой будет осуществляться необходимый вес ассоциации. Вес ассоциации будет эмулировать необходимую для раунда сложность.

Также в логических операциях системы сложности через переменную Counter\_RL была реализована система попыток. В зависимости от выбора сложности будет определённое число попыток.

На протяжении всех попыток игроку всегда будет предоставляться информация об оставшихся попытках, уровне сложности и предыдущих вариантов слов.



Также через переменную типа string Сhoice в логических операциях была реализована возможность выхода из раунда посредством ввода слова “Выход”.



# Выводы

На основании условий задания была проделана работа в создании игрового приложения “Ассоциации”, которая была выбрана мной в качестве темы курсовой работы.

# Список литературы

* 1. Devcolibri.com [электронный ресурс]– <https://devcolibri.com/%D1%83%D1%80%D0%BE%D0%BA-1-%D0%B2%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B2-java-io/>
  2. coderlessons.com [электронный ресурс]– <https://coderlessons.com/tutorials/java-tekhnologii/vyuchit-java-nio/java-nio-kratkoe-rukovodstvo>
  3. java-study.ru [электронный ресурс]– <http://java-study.ru/41-java-util.html>

# Приложение 1: код программы

package CourseWorkKirsanovTD;  
  
import CourseWorkKirsanovTD.ReadDictionary;  
import CourseWorkKirsanovTD.GameCLASS;  
  
  
import java.util.\*;  
  
import java.io.IOException;  
  
  
public class Main {  
  
 ArrayList<DictEntry> EntriesList;  
  
 public static void main(String[] args) throws IOException {  
  
 ArrayList<DictEntry> EntriesList = (ArrayList<DictEntry>) ReadDictionary.*ReadDict*("src\\CourseWorkKirsanovTD\\Русский Региональный Ассоциативный СловарьТезаурус.txt");  
  
  
 if (EntriesList == null)  
  
 {  
  
 System.*out*.println("Словарь не был прочитан!");  
  
 }  
  
  
 do {  
  
 GameCLASS IGRA\_v\_ASSOCIACII = new GameCLASS(EntriesList);  
  
 IGRA\_v\_ASSOCIACII.StartToPlay();  
  
 System.*out*.println("========================================================== ПРОДОЛЖИТЬ? ===========================================================\n" +  
 "======================================================== Нажмите Y или N =========================================================");  
  
  
  
 } while (new Scanner(System.*in*).nextLine().toUpperCase().equals("Y"));  
 }  
}

package CourseWorkKirsanovTD;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class GameCLASS {  
  
  
 private final int DIFFICULTY;  
 private String DiffStr;  
 private int Counter\_RL;  
 private DictEntry Entry\_Dict;  
 ArrayList<DictEntry> EntriesList;  
 private ArrayList<String> AGiven;  
 private ArrayList<String> AssociationEntries;  
 private boolean CONTINUE = true;  
  
  
  
 public GameCLASS(ArrayList<DictEntry> EntriesList) {  
 AGiven = new ArrayList<>();  
 this.EntriesList = EntriesList;  
 System.*out*.println("========================================================= НОВАЯ ИГРА ==========================================================\n"+  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Предоставляется выбор: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n" +  
 "||||||||||||||||||||||||||||| 1 - Лёгкая (Включает в себя 20 попыток, обычные и средние ассоциации) |||||||||||||||||||||||||||||\n" +  
 "||||||||||||||||||||||||||||| 2 - Средняя (Включает в себя 15 попыток, простые и средние ассоциации)|||||||||||||||||||||||||||||\n" +  
 "||||||||||||||||||||||||||||| 3 - Тяжёлая (Включает в себя 5 попыток и очень редкие ассоциации) |||||||||||||||||||||||||||||");  
 this.DIFFICULTY = new Scanner(System.*in*).nextInt();  
  
 switch (DIFFICULTY){  
 case (1):  
 Counter\_RL = 20;  
  
 DiffStr = "Легкий \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";  
 do  
 {  
  
 Entry\_Dict = EntriesList.get((int)Math.*floor*(Math.*random*()\*EntriesList.size()));  
  
  
 } while ( (Entry\_Dict.Get\_STRONG().size()+Entry\_Dict.Get\_MEDIUM().size() )<= Counter\_RL);  
  
 AssociationEntries = new ArrayList<String>( Entry\_Dict.Get\_STRONG().keySet() );  
  
 AssociationEntries.addAll( Entry\_Dict.Get\_MEDIUM().keySet() );  
 break;  
 case (2):  
  
 Counter\_RL = 15;  
 DiffStr = "Средний \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";  
 do {  
  
 Entry\_Dict = EntriesList.get( (int)Math.*floor*(Math.*random*()\*EntriesList.size() ) );  
  
  
 } while ( ( Entry\_Dict.Get\_STRONG().size()+Entry\_Dict.Get\_MEDIUM().size() ) <=Counter\_RL );  
  
 AssociationEntries = new ArrayList<String> (Entry\_Dict.Get\_STRONG().keySet() );  
  
 AssociationEntries.addAll(Entry\_Dict.Get\_MEDIUM().keySet());  
 break;  
 case (3):  
  
 Counter\_RL = 5;  
  
  
 DiffStr = "Сложный \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";  
  
  
 do {  
  
 Entry\_Dict = EntriesList.get((int)Math.*floor*(Math.*random*()\*EntriesList.size()));  
  
  
 } while (Entry\_Dict.Get\_EASY().size()<=Counter\_RL);  
  
 AssociationEntries = new ArrayList<String>(Entry\_Dict.Get\_EASY().keySet());  
 break;  
  
  
 }  
 }  
  
 public void StartToPlay (){  
  
 boolean WINNING = false;  
  
 while (( Counter\_RL > 0 ) && !WINNING && CONTINUE ) {  
  
  
 WINNING = GAMEPLAYRound();  
 }  
  
  
 System.*out*.println("Слово \""+Entry\_Dict.Word\_Getting()+"\"");  
 if (WINNING) {  
 System.*out*.println("========================================================= ПОЗДРАВЛЯЕМ С ПОБЕДОЙ! ==========================================================\n");  
 } else {  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Закончились попытки! Игра окончена. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
 }  
  
 }  
  
 private boolean GAMEPLAYRound(){  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Уровень сложности: "+DiffStr+"\n"+  
 "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Попытки: "+Counter\_RL+" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n"+  
 "Список Ассоциаций:"  
 );  
  
 int ListNumber = (int)(Math.*random*()\*AssociationEntries.size());  
  
 String AssociationNEW = AssociationEntries.get(ListNumber);  
 AssociationEntries.remove(ListNumber);  
 AGiven.add(AssociationNEW);  
  
 for (String ASSOCIATION : AGiven)  
 System.*out*.println(ASSOCIATION);  
  
  
 String Сhoice = new Scanner(System.*in*).nextLine();  
  
 Counter\_RL--;  
  
 if(Сhoice.equals("Выход")) {  
  
 CONTINUE = false;  
  
 System.*out*.println("Вы ввели \"Выход\"");  
  
 return false;  
  
 } else if (Сhoice.equalsIgnoreCase(Entry\_Dict.Word\_Getting())) {  
  
 System.*out*.println("ВЕРНО! \n");  
  
 return true;  
  
 }else {  
 System.*out*.println("НЕ ВЕРНО! ( \n");  
  
 return false;  
 }  
 }  
}

package CourseWorkKirsanovTD;  
  
import java.nio.file.Files;  
import java.nio.file.NoSuchFileException;  
import java.nio.file.Paths;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
import java.util.stream.Stream;  
  
public class ReadDictionary {  
  
 public static List<DictEntry> ReadDict(String DictPath){  
  
 ArrayList<DictEntry> EntriesList = new ArrayList<DictEntry>();  
  
  
  
 try (final Stream<String> STROKI = Files.*lines*(Paths.*get*(DictPath)) .flatMap(line-> Arrays.*stream*(line.split("[,;\\s+]"))))  
  
 {  
  
  
 List<String> storageList = STROKI.filter(x -> x.length() != 0).collect(Collectors.*toList*());  
  
 DictEntry Entry\_Dict = new DictEntry("Старт");  
  
 for (String say : storageList){  
 say = say.replace(",", " ").trim();  
 try {  
 if (!say.matches("[0-10]{1,40}")) {  
  
  
 Entry\_Dict.Add\_Association(say);  
  
 }else  
 {  
  
 Entry\_Dict.Adding\_Weight(Integer.*parseInt*(say));  
  
 }  
  
 } catch (NumberFormatException nfe)  
  
 {  
  
 System.*out*.println("Читатель словаря: "+nfe);  
 }  
  
 if (say.matches("[А-Я]{1,50}:"))  
 {  
  
 Entry\_Dict = new DictEntry(say.substring(0,say.length()-1));  
  
 EntriesList.add(Entry\_Dict);  
  
 }  
 }  
  
  
 } catch (NullPointerException npe)  
  
 {  
  
 System.*out*.println("NullPointerException");  
 }  
  
 catch (NoSuchFileException ex)  
  
 {  
  
 System.*out*.println("Файл не существует");  
  
 }  
  
 catch (Exception e){  
 System.*out*.println(e);  
 }  
  
 return EntriesList;  
 }  
}

package CourseWorkKirsanovTD;  
  
import java.util.HashMap;  
  
public class DictEntry {  
  
  
 private static final int *BoundMedium* = 5; //Значения для веса ассоциации (XY)  
  
 private static final int *BoundStrong* = 15;  
  
 private String latest; //  
  
 private String say;  
  
 private final HashMap<String, Integer> EASY;  
  
 private final HashMap<String, Integer> MEDIUM;  
  
 private final HashMap<String, Integer> HARD;  
  
  
 public DictEntry(String say) {  
 this.say = say;  
  
 EASY = new HashMap<String, Integer>();  
  
 MEDIUM = new HashMap<String, Integer>();  
  
 HARD = new HashMap<String, Integer>();  
 }  
  
 public void Add\_Association(String association) {  
 EASY.put(association, 1);  
  
  
 latest = association;  
 }  
  
  
 public String Word\_Getting() {  
  
 return say;  
 }  
  
 public HashMap<String, Integer> Get\_EASY() {  
  
 return EASY;  
 }  
  
 public HashMap<String, Integer> Get\_MEDIUM() {  
  
 return MEDIUM;  
 }  
  
 public HashMap<String, Integer> Get\_STRONG() {  
  
 return HARD;  
 }  
  
 @Override  
 public String toString() {  
  
 return "Слово = " + say + '\'' + "\n" +  
 ", Простые Ассоциации(" + EASY.size() + ") =" + EASY.toString() + "\n" + ", Средние Ассоциации (" + MEDIUM.size() + ") =" + MEDIUM.toString() + "\n" +  
 ", Сильные Ассоциации (" + HARD.size() + ") =" + HARD.toString();  
 }  
  
  
  
 public void Adding\_Weight(int weight) {  
  
 try {  
  
 if (weight < *BoundMedium*) {  
  
  
 EASY.replace(latest, weight);  
  
  
 } else if ((*BoundMedium* <= weight) && (weight < *BoundStrong*)) {  
  
 HARD.remove(latest);  
  
 MEDIUM.put(latest, weight);  
  
 EASY.remove(latest);  
  
  
 } else {  
  
 MEDIUM.remove(latest);  
  
 HARD.put(latest, weight);  
  
 EASY.remove(latest);  
  
  
 }  
  
 } catch (Exception e) {  
 System.*out*.println("Последний ввод: " + latest + " Ширина: " + weight);  
  
 System.*out*.println(e);  
 }  
 }  
}