package com.javarush.test.level19.lesson03.task01;  
  
/\* TableAdapter  
Измените класс TableAdapter так, чтобы он адаптировал ATable к BTable.  
Метод getHeaderText должен возвращать такую строку "[username] : tablename".  
Пример, "[Amigo] : DashboardTable"  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 //это пример вывода  
 ATable aTable = new ATable() {  
 @Override  
 public String getCurrentUserName() {  
 return "Amigo";  
 }  
  
 @Override  
 public String getTableName() {  
 return "DashboardTable";  
 }  
 };  
  
 BTable table = new TableAdapter(aTable);  
 System.*out*.println(table.getHeaderText());  
 }  
  
 public static class TableAdapter {  
  
 }  
  
 public interface ATable {  
 String getCurrentUserName();  
 String getTableName();  
 }  
  
 public interface BTable {  
 String getHeaderText();  
 }  
}

package com.javarush.test.level19.lesson03.task02;  
  
/\* Адаптер  
Используйте класс AdapterFileOutputStream, чтобы адаптировать FileOutputStream к новому интерфейсу AmigoStringWriter  
\*/  
  
public class AdapterFileOutputStream {  
  
}

package com.javarush.test.level19.lesson03.task02;  
  
import java.io.IOException;  
  
public interface AmigoStringWriter {  
 public void flush() throws IOException;  
  
 void writeString(String s) throws IOException;  
  
 void close() throws IOException;  
}

package com.javarush.test.level19.lesson03.task03;  
  
/\* Адаптация нескольких интерфейсов  
Адаптировать IncomeData к Customer и Contact.  
Классом-адаптером является IncomeDataAdapter.  
Инициализируйте countries перед началом выполнения программы. Соответствие кода страны и названия:  
UA Ukraine  
RU Russia  
CA Canada  
Дополнить телефонный номер нулями до 10 цифр при необходимости (смотри примеры)  
Обратите внимание на формат вывода фамилии и имени человека  
\*/  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class Solution {  
 public static Map<String, String> *countries* = new HashMap<String, String>();  
  
 public static class IncomeDataAdapter {  
 }  
  
 public static interface IncomeData {  
 String getCountryCode(); //example UA  
  
 String getCompany(); //example JavaRush Ltd.  
  
 String getContactFirstName(); //example Ivan  
  
 String getContactLastName(); //example Ivanov  
  
 int getCountryPhoneCode(); //example 38  
  
 int getPhoneNumber(); //example 501234567  
 }  
  
 public static interface Customer {  
 String getCompanyName(); //example JavaRush Ltd.  
  
 String getCountryName(); //example Ukraine  
 }  
  
 public static interface Contact {  
 String getName(); //example Ivanov, Ivan  
  
 String getPhoneNumber(); //example +38(050)123-45-67  
 }  
}

ackage com.javarush.task.task19.task1904;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.Calendar;  
import java.util.Scanner;  
  
/\*  
И еще один адаптер  
Адаптировать Scanner к PersonScanner.  
Классом-адаптером является PersonScannerAdapter.  
В классе адаптере создать приватное финальное поле Scanner fileScanner. Поле инициализировать в конструкторе с одним аргументом типа Scanner.  
  
Данные в файле хранятся в следующем виде:  
Иванов Иван Иванович 31 12 1950  
Петров Петр Петрович 31 12 1957  
  
В файле хранится большое количество людей, данные одного человека находятся в одной строке.  
Метод read() должен читать данные только одного человека.  
\*/  
  
public class Solution {  
  
 public static void main(String[] args) {  
  
 }  
  
 public static class PersonScannerAdapter {  
  
 }  
}

package com.javarush.task.task19.task1904;  
  
import java.io.IOException;  
  
public interface PersonScanner {  
 Person read() throws IOException;  
  
 void close() throws IOException;  
}

package com.javarush.task.task19.task1904;  
  
import java.util.Date;  
  
public class Person {  
 private String firstName;  
 private String middleName;  
 private String lastName;  
 private Date birthDate;  
  
 public Person(String firstName, String middleName, String lastName, Date birthDate) {  
 this.firstName = firstName;  
 this.middleName = middleName;  
 this.lastName = lastName;  
 this.birthDate = birthDate;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("%s %s %s %s", lastName, firstName, middleName, birthDate.toString());  
 }  
}

package com.javarush.task.task19.task1905;  
  
import java.util.HashMap;  
import java.util.Map;  
  
/\*   
Закрепляем адаптер  
Закрепляем адаптер  
Адаптировать Customer и Contact к RowItem.  
Классом-адаптером является DataAdapter.  
  
Инициализируйте countries перед началом выполнения программы. Соответствие кода страны и названия:  
UA Ukraine  
RU Russia  
CA Canada  
\*/  
  
public class Solution {  
 public static Map<String,String> *countries* = new HashMap<String,String>();  
  
 public static void main(String[] args) {  
  
 }  
  
 public static class DataAdapter {  
 public DataAdapter(Customer customer, Contact contact) {  
 }  
 }  
  
 public static interface RowItem {  
 String getCountryCode(); //example UA  
 String getCompany(); //example JavaRush Ltd.  
 String getContactFirstName(); //example Ivan  
 String getContactLastName(); //example Ivanov  
 String getDialString(); //example callto://+380501234567  
 }  
  
 public static interface Customer {  
 String getCompanyName(); //example JavaRush Ltd.  
 String getCountryName(); //example Ukraine  
 }  
  
 public static interface Contact {  
 String getName(); //example Ivanov, Ivan  
 String getPhoneNumber(); //example +38(050)123-45-67  
 }  
}

package com.javarush.task.task19.task1906;  
  
/\*   
Четные байты  
Считать с консоли 2 имени файла.  
Вывести во второй файл все байты с четным индексом.

Использовать FileReader b FileWriter  
Пример:  
второй байт, четвертый байт, шестой байт и т.д.  
Закрыть потоки ввода-вывода.  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1907;  
  
/\*   
Считаем слово  
Считать с консоли имя файла.  
Файл содержит слова, разделенные знаками препинания.  
Вывести в консоль количество слов «world«, которые встречаются в файле.  
Закрыть потоки.  
\*/  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1908;  
  
/\*   
  
Выделяем числа  
Считать с консоли 2 имени файла.  
Вывести во второй файл все числа, которые есть в первом файле.  
Числа выводить через пробел.  
Закрыть потоки.  
  
Пример тела файла:  
12 text var2 14 8v 1  
  
Результат:  
12 14 1  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1909;  
  
/\*   
Замена знаков  
Считать с консоли 2 имени файла.  
Первый Файл содержит текст.  
Считать содержимое первого файла и заменить все точки «.» на знак «!«.  
Результат вывести во второй файл.  
Закрыть потоки.  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1910;  
  
/\*   
Пунктуация  
Пунктуация  
Считать с консоли 2 имени файла.  
Первый Файл содержит текст.  
Считать содержимое первого файла, удалить все знаки пунктуации, включая символы новой строки.  
  
Результат вывести во второй файл.  
  
http://ru.wikipedia.org/wiki/%D0%9F%D1%83%D0%BD%D0%BA%D1%82%D1%83%D0%B0%D1%86%D0%B8%D1%8F  
  
Закрыть потоки.  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1911;  
  
/\*   
Ридер обертка  
В методе main подмени объект System.out написанной тобой ридер-оберткой по аналогии с лекцией.  
Твоя ридер-обертка должна преобразовывать весь текст в заглавные буквы.  
Вызови готовый метод printSomething(), воспользуйся testString.  
Верни переменной System.out первоначальный поток.  
Выведи модифицированную строку в консоль.  
\*/  
  
public class Solution {  
 public static TestString *testString* = new TestString();  
  
 public static void main(String[] args) {  
  
 }  
  
 public static class TestString {  
 public void printSomething() {  
 System.*out*.println("it's a text for testing");  
 }  
 }  
}

package com.javarush.task.task19.task1912;  
  
/\*   
Ридер обертка 2  
В методе main подмените объект System.out написанной тобой ридер-оберткой по аналогии с лекцией.  
Твоя ридер-обертка должна заменять все подстроки «te» на «??«.  
Вызови готовый метод printSomething(), воспользуйтесь testString.  
Верни переменной System.out первоначальный поток.  
Выведи модифицированную строку в консоль.  
\*/  
  
public class Solution {  
 public static TestString *testString* = new TestString();  
  
 public static void main(String[] args) {  
 }  
  
 public static class TestString {  
 public void printSomething() {  
 System.*out*.println("it's a text for testing");  
 }  
 }  
}

package com.javarush.task.task19.task1913;  
  
/\*   
Выводим только цифры  
В методе main подмени объект System.out написанной тобой ридер-оберткой по аналогии с лекцией.  
Твоя ридер-обертка должна выводить только цифры.  
Вызови готовый метод printSomething(), воспользуйтесь testString.  
Верни переменной System.out первоначальный поток.  
Выведи модифицированную строку в консоль.  
  
Пример вывода:  
12345678  
\*/  
  
public class Solution {  
 public static TestString *testString* = new TestString();  
  
 public static void main(String[] args) {  
 }  
  
 public static class TestString {  
 public void printSomething() {  
 System.*out*.println("it's 1 a 23 text 4 f5-6or7 tes8ting");  
 }  
 }  
}

package com.javarush.task.task19.task1914;  
  
/\*   
Решаем пример  
В методе main подмени объект System.out написанной тобой ридер-оберткой по аналогии с лекцией.  
Твоя ридер-обертка должна выводить на консоль решенный пример.  
Вызови готовый метод printSomething(), воспользуйтесь testString.  
Верни переменной System.out первоначальный поток.  
  
Возможные операции: + — \*  
Шаблон входных данных и вывода: a [знак] b = c  
Отрицательных и дробных чисел, унарных операторов — нет.  
  
Пример вывода:  
3 + 6 = 9  
\*/  
  
public class Solution {  
 public static TestString *testString* = new TestString();  
  
 public static void main(String[] args) {  
 }  
 public static class TestString {  
 public void printSomething() {  
  
 System.*out*.println("3 + 6 = ");  
 }  
 }  
}

package com.javarush.task.task19.task1915;  
  
/\*   
Дублируем текст  
Считай с консоли имя файла.  
В методе main подмени объект System.out написанной тобой ридер-оберткой по аналогии с лекцией.  
Твоя ридер-обертка должна дублировать вывод всего текста в файл, имя которого ты считал.  
Вызови готовый метод printSomething(), воспользуйся testString.  
Верни переменной System.out первоначальный поток.  
Закрой поток файла.  
  
Пример вывода на экран:  
it's a text for testing  
  
Пример тела файла:  
it's a text for testing  
\*/  
  
public class Solution {  
 public static TestString *testString* = new TestString();  
  
 public static void main(String[] args) {  
 }  
  
 public static class TestString {  
 public void printSomething() {  
 System.*out*.println("it's a text for testing");  
 }  
 }  
}

package com.javarush.task.task19.task1916;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
import java.util.List;  
  
/\*   
Отслеживаем изменения  
Считать с консоли 2 имени файла — file1, file2.  
Файлы содержат строки, file2 является обновленной версией file1, часть строк совпадают.  
Нужно создать объединенную версию строк, записать их в список lines.  
Операции ADDED и REMOVED не могут идти подряд, они всегда разделены SAME.  
В оригинальном и редактируемом файлах пустых строк нет.  
  
Пример:  
оригинальный редактированный общий  
file1: file2: результат:(lines)  
  
строка1 строка1 SAME строка1  
строка2 REMOVED строка2  
строка3 строка3 SAME строка3  
строка4 REMOVED строка4  
строка5 строка5 SAME строка5  
строка0 ADDED строка0  
строка1 строка1 SAME строка1  
строка2 REMOVED строка2  
строка3 строка3 SAME строка3  
строка5 ADDED строка5  
строка4 строка4 SAME строка4  
строка5 REMOVED строка5  
\*/  
  
public class Solution {  
 public static List<LineItem> *lines* = new ArrayList<LineItem>();  
  
 public static void main(String[] args) throws IOException {  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
 BufferedReader file1 = new BufferedReader(new FileReader(reader.readLine()));  
 BufferedReader file2 = new BufferedReader(new FileReader(reader.readLine()));  
 }  
  
  
 public static enum Type {  
 *ADDED*, //добавлена новая строка  
 *REMOVED*, //удалена строка  
 *SAME* //без изменений  
 }  
  
 public static class LineItem {  
 public Type type;  
 public String line;  
  
 public LineItem(Type type, String line) {  
 this.type = type;  
 this.line = line;  
 }  
 }  
}

package com.javarush.task.task19.task1917;  
  
/\*   
Свой FileWriter  
Реализовать логику FileConsoleWriter.  
Класс FileConsoleWriter должен содержать приватное поле FileWriter fileWriter.  
Класс FileConsoleWriter должен содержать все конструкторы, которые инициализируют fileWriter для записи.  
  
При записи данных в файл, должен дублировать эти данные на консоль.  
\*/  
  
public class FileConsoleWriter {  
 public static void main(String[] args) {  
  
 }  
  
}

package com.javarush.task.task19.task1918;  
  
/\*   
Знакомство с тегами  
Считайте с консоли имя файла, который имеет HTML-формат.  
  
Пример:  
Info about Leela <span xml:lang=»en» lang=»en»><b><span>Turanga Leela  
</span></b></span><span>Super</span><span>girl</span>  
  
Первым параметром в метод main приходит тег. Например, «span«.  
Вывести на консоль все теги, которые соответствуют заданному тегу.  
Каждый тег на новой строке, порядок должен соответствовать порядку следования в файле.  
Количество пробелов, n, r не влияют на результат.  
Файл не содержит тег CDATA, для всех открывающих тегов имеется отдельный закрывающий тег, одиночных тегов нет.  
Тег может содержать вложенные теги.  
  
Пример вывода:  
<span xml:lang="en" lang="en"><b><span>Turanga Leela</span></b></span>  
<span>Turanga Leela</span>  
<span>Super</span>  
<span>girl</span>  
  
Шаблон тега:  
<tag>text1</tag>  
<tag text2>text1</tag>  
<tag  
text2>text1</tag>  
  
text1, text2 могут быть пустыми  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1919;  
  
/\*   
Считаем зарплаты  
В метод main первым параметром приходит имя файла.  
В этом файле каждая строка имеет следующий вид:  
имя значение  
где [имя] — String, [значение] — double. [имя] и [значение] разделены пробелом.  
  
Для каждого имени посчитать сумму всех его значений.  
Все данные вывести в консоль, предварительно отсортировав в возрастающем порядке по имени.  
Закрыть потоки.  
  
Пример входного файла:  
Петров 2  
Сидоров 6  
Иванов 1.35  
Петров 3.1  
  
Пример вывода:  
Иванов 1.35  
Петров 5.1  
Сидоров 6.0  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1920;  
  
/\*   
Самый богатый  
В метод main первым параметром приходит имя файла.  
В этом файле каждая строка имеет следующий вид:  
имя значение  
где [имя] — String, [значение] — double. [имя] и [значение] разделены пробелом.  
  
Для каждого имени посчитать сумму всех его значений.  
Вывести в консоль имена в алфавитном порядке, у которых максимальная сумма.  
Имена разделять пробелом либо выводить с новой строки.  
Закрыть потоки.  
  
Пример входного файла:  
Петров 0.501  
Иванов 1.35  
Петров 0.85  
  
Пример вывода:  
Петров  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1921;  
  
import java.util.ArrayList;  
import java.util.List;  
  
/\*   
Хуан Хуанович  
В метод main первым параметром приходит имя файла.  
В этом файле каждая строка имеет следующий вид:  
имя день месяц год  
где [имя] — может состоять из нескольких слов, разделенных пробелами, и имеет тип String.  
[день] — int, [месяц] — int, [год] — int  
данные разделены пробелами.  
  
Заполнить список PEOPLE используя данные из файла.  
Закрыть потоки.  
  
Пример входного файла:  
Иванов Иван Иванович 31 12 1987  
Вася 15 5 2013  
\*/  
  
public class Solution {  
 public static final List<Person> *PEOPLE* = new ArrayList<Person>();  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task19.task1922;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
import java.util.List;  
  
/\*   
Ищем нужные строки  
Считать с консоли имя файла.  
Вывести в консоль все строки из файла, которые содержат всего 2 слова из списка words.  
Закрыть потоки.  
  
Пример:  
words содержит слова А, Б, В  
  
Строки:  
В Б А Д //3 слова из words, не подходит  
Д А Д //1 слово из words, не подходит  
Д А Б Д //2 слова — подходит, выводим  
\*/  
  
public class Solution {  
 public static List<String> *words* = new ArrayList<String>();  
  
 static {  
 *words*.add("файл");  
 *words*.add("вид");  
 *words*.add("В");  
 }  
  
 public static void main(String[] args) throws IOException {  
  
 }  
}

package com.javarush.task.task19.task1923;  
  
/\*   
Слова с цифрами  
В метод main первым параметром приходит имя файла1, вторым — файла2.  
Файл1 содержит строки со слов, разделенные пробелом.  
Записать через пробел в Файл2 все слова, которые содержат цифры, например, а1 или abc3d.  
Закрыть потоки.  
\*/  
  
public class Solution {  
 public static void main(String[] args) {  
 }  
}

package com.javarush.task.task19.task1924;  
  
import java.util.HashMap;  
import java.util.Map;  
  
/\*   
Замена чисел  
1. В статическом блоке инициализировать словарь map парами [число-слово] от 0 до 12 включительно.  
Например, 0 — «ноль«, 1 — «один«, 2 — «два»  
2. Считать с консоли имя файла, считать содержимое файла.  
3. Заменить все числа на слова используя словарь map.  
4. Результат вывести на экран.  
5. Закрыть потоки.  
  
Пример данных в файле:  
Это стоит 1 бакс, а вот это - 12 .  
Переменная имеет имя file1.  
110 - это число.  
  
Пример вывода в консоль:  
Это стоит один бакс, а вот это - двенадцать .  
Переменная имеет имя file1.  
110 - это число.  
\*/  
  
public class Solution {  
 public static Map<Integer, String> *map* = new HashMap<Integer, String>();  
  
 public static void main(String[] args) {  
  
 }  
}

package com.javarush.task.task19.task1925;  
  
/\*   
Длинные слова  
В метод main первым параметром приходит имя файла1, вторым — файла2.  
Файл1 содержит слова, разделенные пробелом.  
Записать через запятую в Файл2 слова, длина rr которых строго больше 6.  
В конце файла2 запятой не должно быть.  
Закрыть потоки.  
  
Пример выходных данных в файл2:  
длинное,короткое,аббревиатура  
\*/  
  
import java.io.\*;  
import java.util.ArrayList;  
  
public class Solution {  
 public static void main(String[] args) throws IOException {  
 }  
}

package com.javarush.task.task19.task1926;  
  
/\*   
Перевертыши  
1 Считать с консоли имя файла. Считать содержимое файла.  
2 Для каждой строки в файле:  
2.1 переставить все символы в обратном порядке.  
2.2 вывести на экран.  
3 Закрыть потоки.  
  
Пример тела входного файла:  
я - программист.  
Амиго  
  
Пример результата:  
.тсиммаргорп - я  
огимА  
\*/  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
  
public class Solution {  
 public static void main(String[] args) throws IOException {  
 }  
}

package com.javarush.task.task19.task1927;  
  
/\*   
Контекстная реклама  
В методе main подмени объект System.out написанной тобой реадер-оберткой.  
Твоя реадер-обертка должна выводить на консоль контекстную рекламу после каждого второго println-а.  
Вызови готовый метод printSomething(), воспользуйся testString.  
Верни переменной System.out первоначальный поток.  
  
Рекламный текст: «JavaRush — курсы Java онлайн»  
  
Пример вывода:  
first  
second  
JavaRush - курсы Java онлайн  
third  
fourth  
JavaRush - курсы Java онлайн  
fifth  
\*/  
  
public class Solution {  
 public static TestString *testString* = new TestString();  
  
 public static void main(String[] args) {  
 }  
  
 public static class TestString {  
 public void printSomething() {  
 System.*out*.println("first");  
 System.*out*.println("second");  
 System.*out*.println("third");  
 System.*out*.println("fourth");  
 System.*out*.println("fifth");  
 }  
 }  
}

package com.javarush.task.task19.task1928;  
  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStream;  
  
/\*   
Исправить ошибку. Классы и интерфейсы  
Программа содержит всего 1 логическую ошибку.  
Найди и исправь ее.  
\*/  
  
public class Solution {  
 {  
 System.*out*.println("it's Solution class");  
 }  
  
 public static void main(String... args) throws IOException {  
 try (  
 FileOutputStream outputStream = new FileOutputStream(args[0]);  
 InputStream is = Solution.class.getClassLoader().getResourceAsStream(args[1]);  
 ) {  
 ;  
 byte[] b = new byte[is.available()];  
 outputStream.write(is.read(b));  
  
 int value = 123\_456\_789;  
 System.*out*.println(value);  
  
 Example result = null;  
 String s = "a";  
 switch (s) {  
 case "a": {  
 result = new Solution().new A();  
 break;  
 }  
 case "b": {  
 result = new Solution().new B();  
 break;  
 }  
 case "c": {  
 result = new Solution().new C();  
 break;  
 }  
 }  
  
 if (result instanceof A) {  
 C p = (C) result;  
 System.*out*.println(p.getClass().getSimpleName());  
 }  
  
 } catch (IOException e) {  
 }  
 }  
  
 interface Example {  
 }  
  
 class A implements Example {  
 {  
 System.*out*.println("it's A class");  
 }  
 }  
  
 class B implements Example {  
 {  
 System.*out*.println("it's B class");  
 }  
 }  
  
 class C extends A {  
 {  
 System.*out*.println("it's C class");  
 }  
 }  
}