EPAM Systems, RD Dep.

Практические задания для тренинга

Task. Information handling

REVISION HISTORY					
Ver.	**Description of Change**	**Author**	**Date**	**Approved**	
				Name	**Effective Date**
<1.0>	Первая версия	Игорь Блинов Ольга Смолякова	<26.05.2014>		

Information handling

Разработать приложение, разбирающее и обрабатывающее текст.

Необходимо создать приложение, разбирающее текст из учебника по программированию из файла и позволяющее выполнять с текстом три различных операции.

Общие требования к проекту:

Разобранный текст должен быть представлен в виде объекта (текста), содержащего, например, предложения и блоки кода, предложение может содержать слова предложения. Слова предложения (части предложения), могут быть, например, простыми словами и знаками препинания. Данные классы являются классами сущностей и не должны быть перегружены методами логики.

Разобранный текст необходимо восстановить в первоначальном виде, за исключением пробелов между элементами. Пробелы и знаки табуляции при разборе могут заменяться одним пробелом.

Для деления текста на предложения и другие составляющие следует использовать регулярные выражения. Не забывать, что регулярные выражения для приложения являются литеральными константами.

Код, выполняющий разбитие текста на составляющие части, следует оформить в виде классов-парсеров.

При разработке парсеров, разбирающих текст, необходимо следить за количеством создаваемых объектов-парсеров.

При реализации задания можно использовать шаблоны Composite и Chain of Responsibility.

При обработке исключительных ситуаций приложение необходимо использовать логгер Log4j.

Созданное приложение должно позволять реализовывать группу задач по работе над текстом (задачи приведены ниже) без “переписывания” существующего кода.

Функциональные возможности, варианты для реализации.

1. Найти наибольшее количество предложений текста, в которых есть одинаковые слова.
2. Вывести все предложения заданного текста в порядке возрастания количества слов в каждом из них.
3. Найти такое слово в первом предложении, которого нет ни в одном из остальных предложений.
4. Во всех вопросительных предложениях текста найти и напечатать без повторений слова заданной длины.
5. В каждом предложении текста поменять местами первое слово с последним, не изменяя длины предложения.
6. Напечатать слова текста в алфавитном порядке по первой букве. Слова, начинающиеся с новой буквы, печатать с красной строки.
7. Рассортировать слова текста по возрастанию доли гласных букв (отношение количества гласных к общему количеству букв в слове).
8. Слова текста, начинающиеся с гласных букв, рассортировать в алфавитном порядке по первой согласной букве слова.
9. Все слова текста рассортировать по возрастанию количества заданной буквы в слове. Слова с одинаковым количеством букв расположить в алфавитном порядке.
10. Существует текст и список слов. Для каждого слова из заданного списка найти, сколько раз оно встречается в каждом предложении, и рассортировать слова по убыванию общего количества вхождений.
11. В каждом предложении текста исключить подстроку максимальной длины, начинающуюся и заканчивающуюся заданными символами.
12. Из текста удалить все слова заданной длины, начинающиеся на согласную букву.
13. Отсортировать слова в тексте по убыванию количества вхождений заданного символа, а в случае равенства – по алфавиту.
14. В заданном тексте найти подстроку максимальной длины, являющуюся палиндромом, т.е. читающуюся слева направо и справа налево одинаково.
15. Преобразовать каждое слово в тексте, удалив из него все последующие (предыдущие) вхождения первой (последней) буквы этого слова.
16. В некотором предложении текста слова заданной длины заменить указанной подстрокой, длина которой может не совпадать с длиной слова.

Пример текста для разбора.

1. The if-then and if-then-else Statements

1.1. The if-then Statement

The if-then statement is the most basic of all the control flow statements. It tells your program to execute a certain section of code only if a particular test evaluates to true. For example, the Bicycle class could allow the brakes to decrease the bicycle's speed only if the bicycle is already in motion. One possible implementation of the applyBrakes method could be as follows:

```
void applyBrakes() {
   // the "if" clause: bicycle must be moving
   if (isMoving){
      // the "then" clause: decrease current speed
      currentSpeed--;
   }
}
```

If this test evaluates to false (meaning that the bicycle is not in motion), control jumps to the end of the if-then statement.

In addition, the opening and closing braces are optional, provided that the "then" clause contains only one statement:

```
void applyBrakes() {
   // same as above, but without braces
   if (isMoving)
      currentSpeed--;
}
```

Deciding when to omit the braces is a matter of personal taste. Omitting them can make the code more brittle. If a second statement is later added to the "then" clause, a common mistake would be forgetting to add the newly required braces. The compiler cannot catch this sort of error; you'll just get the wrong results.

1.2. The if-then-else Statement

The if-then-else statement provides a secondary path of execution when an "if" clause evaluates to false. You could use an if-then-else statement in the applyBrakes method to take some action if the brakes are applied when the bicycle is not in motion. In this case, the action is to simply print an error message stating that the bicycle has already stopped.

```
void applyBrakes() {
   if (isMoving) {
      currentSpeed--;
   } else {
      System.err.println("The bicycle has already stopped!");
   }
}
```

The following program, IfElseDemo, assigns a grade based on the value of a test score: an A for a score of 90% or above, a B for a score of 80% or above, and so on.

```
class IfElseDemo {
   public static void main(String[] args) {

      int testscore = 76;
      char grade;

      if (testscore >= 90) {
         grade = 'A';
      } else if (testscore >= 80) {
         grade = 'B';
      } else if (testscore >= 70) {
         grade = 'C';
      } else if (testscore >= 60) {
         grade = 'D';
      } else {
         grade = 'F';
      }
      System.out.println("Grade = " + grade);
   }
}
```

The output from the program is:

```
   Grade = C
```

You may have noticed that the value of testscore can satisfy more than one expression in the compound statement: 76 >= 70 and 76 >= 60. However, once a condition is satisfied, the appropriate statements are executed (grade = 'C';) and the remaining conditions are not evaluated.