Szyfr polialfabetyczny w języku Java

- Wprowadzenie
- Przeczytaj
- Animacja
- Sprawdź się
- Dla nauczyciela

Źródło: Amador Loureiro, dostępny w internecie: unsplash.com, domena publiczna.

Poznaliśmy już szyfry opierające się na jednym alfabecie szyfrującym, jak szyfr Cezara. Trudniejsze do złamania są bazujące na większej liczbie alfabetów szyfry polialfabetyczne. Przykładem jest szyfr Vigenère'a. W tym e-materiale dowiesz się, jak zaimplementować szyfr tego typu w języku Java.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych lekcjach z tej serii:

- Szyfr polialfabetyczny w języku C++,
- Szyfr polialfabetyczny w języku Python.

Więcej zadań? Sięgnij do: Szyfr polialfabetyczny – zadania maturalne.

Twoje cele

- Prześledzisz, jak szyfrować ciągi znaków za pomocą szyfru Vigenère'a.
- Przeanalizujesz implementację w języku Java szyfru Vigenère'a.
- Wykonasz kilka ćwiczeń sprawdzających wiedzę dotyczącą implementacji szyfru Vigenère'a w języku Java.

Przeczytaj

Problem 1

Napisz program szyfrujący dany komunikat za pomocą szyfru polialfabetycznego (wykorzystaj algorytm Vigenère'a).

Przetestuj działanie programu dla słowa jawnego ULEWA oraz klucza szyfrującego MODA.

Specyfikacja problemu:

Dane:

`wiadomosc` – słowo jawne; łańcuch znaków

`slowo_klucz` – klucz szyfrujący; łańcuch znaków

Wynik:

`szyfrogram` – słowo tajne; łańcuch znaków

```
public static void main(String[] args){
    // Tutaj dodaj własny kod. Użyj funkcji
    // System.out.print();
  }
```

```
```

Polecenie 1

Porównaj swoje rozwiązanie z zaprezentowanym w filmie.

Film dostępny pod adresem /preview/resource/R1Ugz7WqX453r

Film nawiązujący do treści materiału: Szyfr polialfabetyczny Vigenere'a.

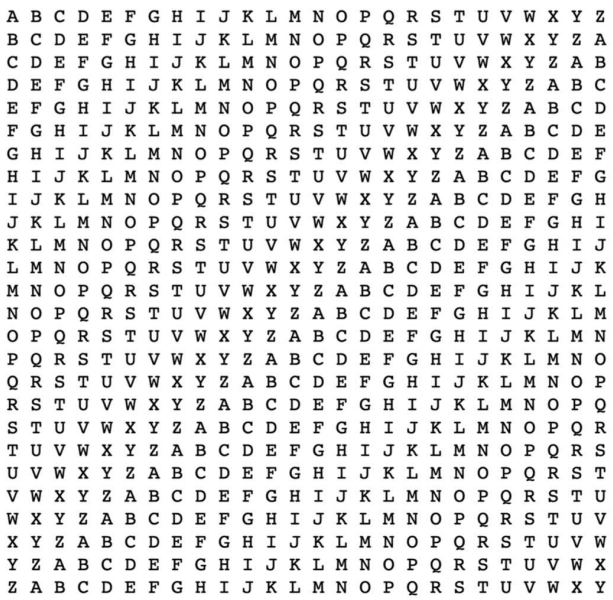
Przykład 1

> Napisz w języku Java algorytm, który korzystając z szyfru Vigenère'a, zaszyfruje wiadomość.
>
> **Łańcuch znaków do zaszyfrowania**: „PROGRAM"
>
> **Klucz**: „KLUCZ"

W oparciu o tabelę, którą konstruowaliśmy w e-materiale Szyfr polialfabetyczny, spróbujmy zakodować słowo „PROGRAM" kluczem „KLUCZ".

1. Weź poziomy rząd w tabeli, którego pierwszą literą jest „P" (pierwsza litera słowa do zaszyfrowania),
2. Weź pionowy rząd w tabeli, którego pierwszą litera jest „K" (pierwsza litera klucza).
3. Literą zaszyfrowana jest litera znajdująca się na przecięciu obu rzędów.
4. Weź kolejną literę słowa do zaszyfrowania oraz kolejną literę klucza i wróć do kroku nr 1 (w przypadku, gdy nie ma kolejnych liter klucza, należy wziąć pierwszą literę klucza).

Poniżej zostało przestawione graficzne rozwiązanie zadanego przykładu:



Implementacja w języku Java

Przejdźmy teraz do implementacji powyższego algorytmu, w postaci programu, w języku Java.

Do odwzorowania omówionej wcześniej tablicy użyjemy znanej już struktury danych – **tablicy dwuwymiarowej**.

```
1 static char[][] tabela = new char[26][26];
```

Zadeklarujemy również tablicę jednowymiarową `alfabetLacinski` o stałym rozmiarze 26, która będzie przechowywała kolejne litery alfabetu łacińskiego.

```
static char[] alfabetLacinski = new char[26];
```

Przejdźmy teraz do zadeklarowania funkcji odpowiedzialnej za utworzenie tablicy szyfrowania (dokładnie takiej, jak na powyższej grafice) oraz wyświetlenie jej na ekranie konsoli.

Funkcja przyjmie nazwę `wypelnienieTablicy()`, będzie typu `void` (nie zwracająca parametrów).

Funkcja nie będzie przyjmować żadnych parametrów.

```
static void wypelnienieTablicy() {

}
```

Następnym krokiem jest wypełnienie naszej tablicy jednowymiarowej `alfabetLacinski` kolejnymi literami alfabetu łacińskiego. Użyjemy do tego pętli oraz tablicy ASCII.

```
static void wypelnienieTablicy() {
    for (int i = 97, z = 0; i < 123; i++, z++) {
        alfabetLacinski[z] = (char) i;
    }
}
```

Kolejnym krokiem jest wypełnienie tablicy dwuwymiarowej `tabela` tak, aby zgadzała się z tabelą przedstawioną na wcześniejszych grafikach. Będzie ona służyć do szyfrowania kolejnych liter.

```
static void wypelnienieTablicy() {
    for (int i = 97, z = 0; i < 123; i++, z++) {
        alfabetLacinski[z] = (char) i;
    }

    for (int j = 0; j < 26; j++) {

```

```
7
8       }
9   }
```

W pierwszej pętli, której zmienną sterującą jest `j`, iterujemy po kolejnych wierszach tabeli, natomiast w drugiej pętli, której zmienną sterującą jest `k`, odbywa się iteracjach po wszystkich komórkach w danym wierszu – nadawane są kolejne wartości, zgodne z kolejnością w tabeli alfabet.

```
1  static void wypelnienieTablicy() {
2      for (int i = 97, z = 0; i < 123; i++, z++) {
3          alfabetLacinski[z] = (char) i;
4      }
5
6      for (int j = 0; j < 26; j++) {
7          for (int k = 0; k < 26; k++) {
8              tabela[j][k] = alfabetLacinski[k];
9          }
10     }
11 }
```

W trzeciej pętli odbywa się zamiana kolejności liter w tabeli alfabet o jedną pozycję.

```
1  static void wypelnienieTablicy() {
2      for (int i = 97, z = 0; i < 123; i++, z++) {
3          alfabetLacinski[z] = (char) i;
4      }
5
6      for (int j = 0; j < 26; j++) {
7          for (int k = 0; k < 26; k++) {
8              tabela[j][k] = alfabetLacinski[k];
9          }
10
11         for (int l = 0; l < 26 - 1; l++) {
12             char temp = alfabetLacinski[l];
13             alfabetLacinski[l] = alfabetLacinski[l + 1];
14             alfabetLacinski[l + 1] = temp;
15         }
16     }
17 }
```

Spróbujmy wypisać zawartość tabeli dwuwymiarowej, aby sprawdzić, czy jest ona zgodna z założeniami.

```
static void wypelnienieTablicy() {
    for (int i = 97, z = 0; i < 123; i++, z++) {
        alfabetLacinski[z] = (char) i;
    }

    for (int j = 0; j < 26; j++) {
        for (int k = 0; k < 26; k++) {
            tabela[j][k] = alfabetLacinski[k];
        }

        for (int l = 0; l < 26 - 1; l++) {
            char temp = alfabetLacinski[l];
            alfabetLacinski[l] = alfabetLacinski[l + 1];
            alfabetLacinski[l + 1] = temp;
        }
    }

    for (int m = 0; m < 26; m++) {
        for (int n = 0; n < 26; n++) {
            System.out.print(tabela[m][n]);
        }

        System.out.println();
    }
}
```

Wynik konsoli:

```
abcdefghijklmnopqrstuvwxyz
bcdefghijklmnopqrstuvwxyza
cdefghijklmnopqrstuvwxyzab
defghijklmnopqrstuvwxyzabc
efghijklmnopqrstuvwxyzabcd
fghijklmnopqrstuvwxyzabcde
ghijklmnopqrstuvwxyzabcdef
hijklmnopqrstuvwxyzabcdefg
ijklmnopqrstuvwxyzabcdefgh
```

```
jklmnopqrstuvwxyzabcdefghi
klmnopqrstuvwxyzabcdefghij
lmnopqrstuvwxyzabcdefghijk
mnopqrstuvwxyzabcdefghijkl
nopqrstuvwxyzabcdefghijklm
opqrstuvwxyzabcdefghijklmn
pqrstuvwxyzabcdefghijklmno
qrstuvwxyzabcdefghijklmnop
rstuvwxyzabcdefghijklmnopq
stuvwxyzabcdefghijklmnopqr
tuvwxyzabcdefghijklmnopqrs
uvwxyzabcdefghijklmnopqrst
vwxyzabcdefghijklmnopqrstu
wxyzabcdefghijklmnopqrstuv
xyzabcdefghijklmnopqrstuvw
yzabcdefghijklmnopqrstuvwx
zabcdefghijklmnopqrstuvwxy
```

Jak widać (po wywołaniu funkcji `wypelnienieTablicy` w głównej funkcji programu `main`) powstała tabela jest zgodna z wcześniej omówioną tabelą, a więc możemy przejść do kolejnej części implementacji – szyfrowania za pomocą utworzonej tabeli.

W tym celu utwórzmy kolejną funkcję, nadajmy jej nazwę `zaszyfruj()` oraz nadajmy jej typ i przyjmowane parametry.

Funkcja powinna zwracać zaszyfrowany ciąg znaków, a więc powinna być łańcuchem znaków `String`. Przyjmowanymi parametrami będą dwa łańcuchy znaków `String`:

- `String doZaszyfrowania` – podany ciąg znaków, który należy zaszyfrować,
- `String klucz` – ciąg znaków określający klucz wedle jakiego będziemy szyfrować.

```
static String zaszyfruj(String doZaszyfrowania, String klucz) {
    String wynikSzyfrowania = "";

    for (int i = 0, j = 0; i < doZaszyfrowania.length(); i++) {
        wynikSzyfrowania += tabela[(int)(doZaszyfrowania.charAt(i

        if (j == klucz.length() - 1) {
            j = 0;
        } else {
            j++;
        }
```

```
    }

    return wynikSzyfrowania;
}
```

Następnie należy wykonać iterację po słowie do zaszyfrowania i w każdej kolejnej iteracji nadpisywać zmienną `wynikSzyfrowania` o odpowiednia literę. Następujący fragment kodu:

```
[(int)(doZaszyfrowania.charAt(i) - 97)][(int)(klucz.charAt(j) - 9
```

określa komórkę w tabeli, która leży na przecięciu określonego rzędu i wiersza.

```
static String zaszyfruj(String doZaszyfrowania, String klucz) {
    String wynikSzyfrowania = "";

    for (int i = 0, j = 0; i < doZaszyfrowania.length(); i++) {
            wynikSzyfrowania += tabela[(int)(doZaszyfrowania.char

            j++;
    }

    return wynikSzyfrowania;
}
```

Może się zdarzyć, że klucz, wedle jakiego szyfrujemy, będzie krótszy niż słowo do zaszyfrowania. W takim przypadku tworzymy warunek – gdy do szyfrowania użyjemy ostatniego znaku klucza, zerujemy zmienną iterującą po kolejnych znakach klucza.

```
static String zaszyfruj(String doZaszyfrowania, String klucz) {
    String wynikSzyfrowania = "";

    for (int i = 0, j = 0; i < doZaszyfrowania.length(); i++) {
            wynikSzyfrowania += tabela[(int)(doZaszyfrowania.char

        if (j == klucz.length() - 1) {
            j = 0;
        } else {
            j++;
```

```
        }
    }

    return wynikSzyfrowania;
}
```

Wywołajmy teraz obie funkcje. Jako słowo do zaszyfrowania podajmy ciąg znaków „PROGRAM", natomiast jako wartość klucza przyjmijmy „KLUCZ".

```
public static void main(String[] args) {
    wypelnienieTablicy();
    System.out.println(zaszyfruj("program", "klucz"));
}
```

Wynik działania programu:

```
abcdefghijklmnopqrstuvwxyz
bcdefghijklmnopqrstuvwxyza
cdefghijklmnopqrstuvwxyzab
defghijklmnopqrstuvwxyzabc
efghijklmnopqrstuvwxyzabcd
fghijklmnopqrstuvwxyzabcde
ghijklmnopqrstuvwxyzabcdef
hijklmnopqrstuvwxyzabcdefg
ijklmnopqrstuvwxyzabcdefgh
jklmnopqrstuvwxyzabcdefghi
klmnopqrstuvwxyzabcdefghij
lmnopqrstuvwxyzabcdefghijk
mnopqrstuvwxyzabcdefghijkl
nopqrstuvwxyzabcdefghijklm
opqrstuvwxyzabcdefghijklmn
pqrstuvwxyzabcdefghijklmno
qrstuvwxyzabcdefghijklmnop
rstuvwxyzabcdefghijklmnopq
stuvwxyzabcdefghijklmnopqr
tuvwxyzabcdefghijklmnopqrs
uvwxyzabcdefghijklmnopqrst
vwxyzabcdefghijklmnopqrstu
wxyzabcdefghijklmnopqrstuv
xyzabcdefghijklmnopqrstuvw
```

```
yzabcdefghijklmnopqrstuvwx
zabcdefghijklmnopqrstuvwxy
zciiqkx
```

Po porównaniu z wcześniej omówionym przykładem teoretycznym, zauważamy, że program szyfruje w sposób prawidłowy.

Oto kod całego programu:

```
public class VigenereCipher {

    static char[] alfabetLacinski = new char[26];
    static char[][] tabela = new char[26][26];


    static void wypelnienieTablicy() {
        for(int i = 97, z = 0; i < 123; i++, z++) {
            alfabetLacinski[z] = (char) i;
        }

        for(int j = 0; j < 26; j++) {
            for(int k = 0; k < 26; k++) {
                tabela[j][k] = alfabetLacinski[k];
            }

            for(int l = 0; l < 26 - 1; l++) {
                char temp = alfabetLacinski[l];
                alfabetLacinski[l] = alfabetLacinski[l + 1];
                alfabetLacinski[l + 1] = temp;
            }
        }

        for(int m = 0; m < 26; m++) {
            for(int n = 0; n < 26; n++) {
                System.out.print(tabela[m][n]);
            }
            System.out.println();
        }

    }

```

```
33      static String zaszyfruj(String doZaszyfrowania, String klucz)
34          String wynikSzyfrowania = "";
35
36          for (int i = 0, j = 0; i < doZaszyfrowania.length(); i++)
37              wynikSzyfrowania += tabela[(int)(doZaszyfrowania.char
38
39              if (j == klucz.length() - 1) {
40                  j = 0;
41              } else {
42                  j++;
43              }
44          }
45
46          return wynikSzyfrowania;
47      }
48
49      public static void main(String[] args) {
50          wypelnienieTablicy();
51          System.out.println(zaszyfruj("program", "klucz"));
52      }
53
54  }
```

Słownik

iteracja

słowo pochodzące od łacińskiego *iteratio* (powtarzanie); oznacza powtarzanie w pętli tych samych instrukcji, aż do spełnienia pewnego warunku

zmienna sterująca

zmienna przechowująca numer cyklu wykonywanego przez pętlę

Animacja

Polecenie 1

Zapoznaj się z animacją. Poszukaj informacji na temat historycznego i współczesnego wykorzystania omawianego szyfru.

Film dostępny pod adresem /preview/resource/RiRgKSzli7bDK

Film nawiązujący do treści materiału: Szyfr polialfabetyczny Vigenere'a.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Znasz już oryginalną wersję szyfru Vigenère'a – dotyczy ona określonych zasad przy tworzeniu klucza szyfrującego. Na podstawie przedstawionego kodu napisz funkcję, która przyjmie jeden parametr – łańcuch znaków do zaszyfrowania, a następnie utworzy odpowiedni klucz (pierwszy znak klucza ma być równy „J") i go zwróci. Przetestuj działanie programu dla łańcucha znaków do zaszyfrowania *tajnytekst.*

Specyfikacja problemu:

Dane:

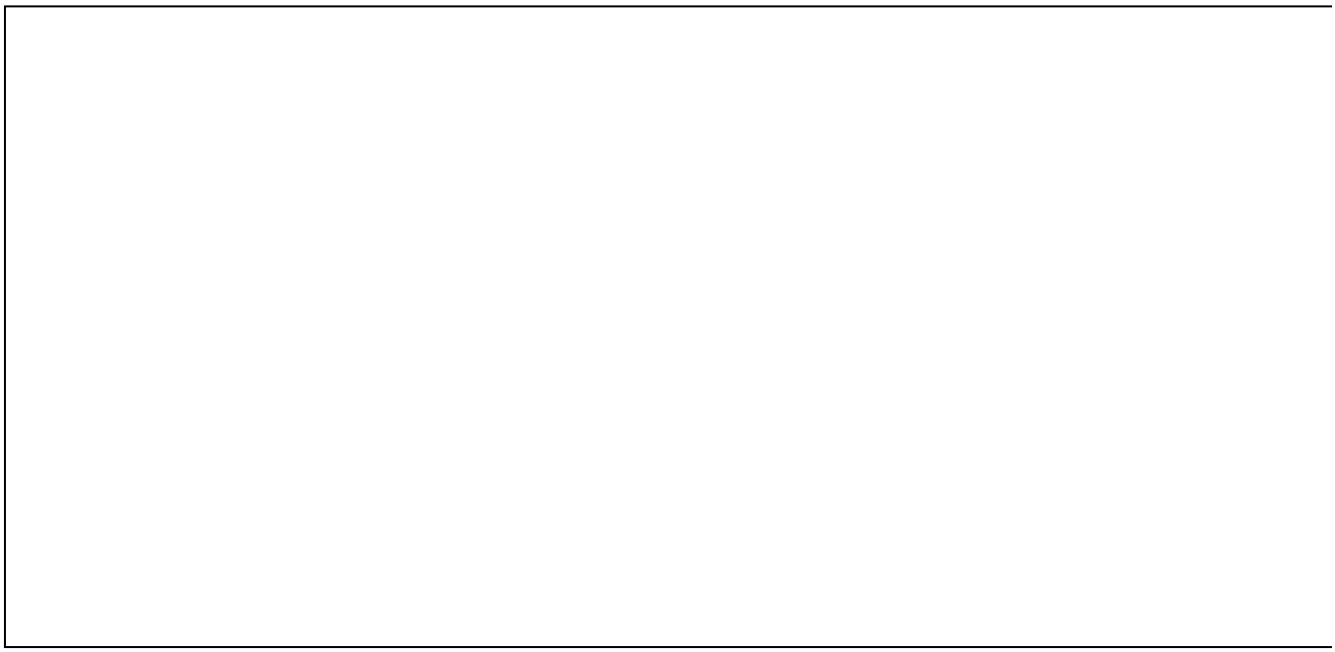
- *doZaszyfrowania* – łańcuch znaków

Wynik:

- *klucz* – łańcuch znaków

Twoje zadania

1. Program ma utworzyć klucz szyfrujący (według oryginalnej wersji szyfru Vigenère'a) dla zadanego ciągu znaków „tajnytekst", a następnie wypisać go na ekran konsoli. Za pierwszy znak klucza szyfrującego należy przyjąć znak „J".



Ćwiczenie 2

W tym zadaniu musimy rozszerzyć program z **Ćwiczenia 1** o generowanie tablicy szyfrującej oraz wykonanie szyfrowania. Klucz ma być tworzony tak, jak w poprzednim zadaniu (według oryginalnej wersji szyfru Vigenère'a).

Specyfikacja problemu:

Dane:

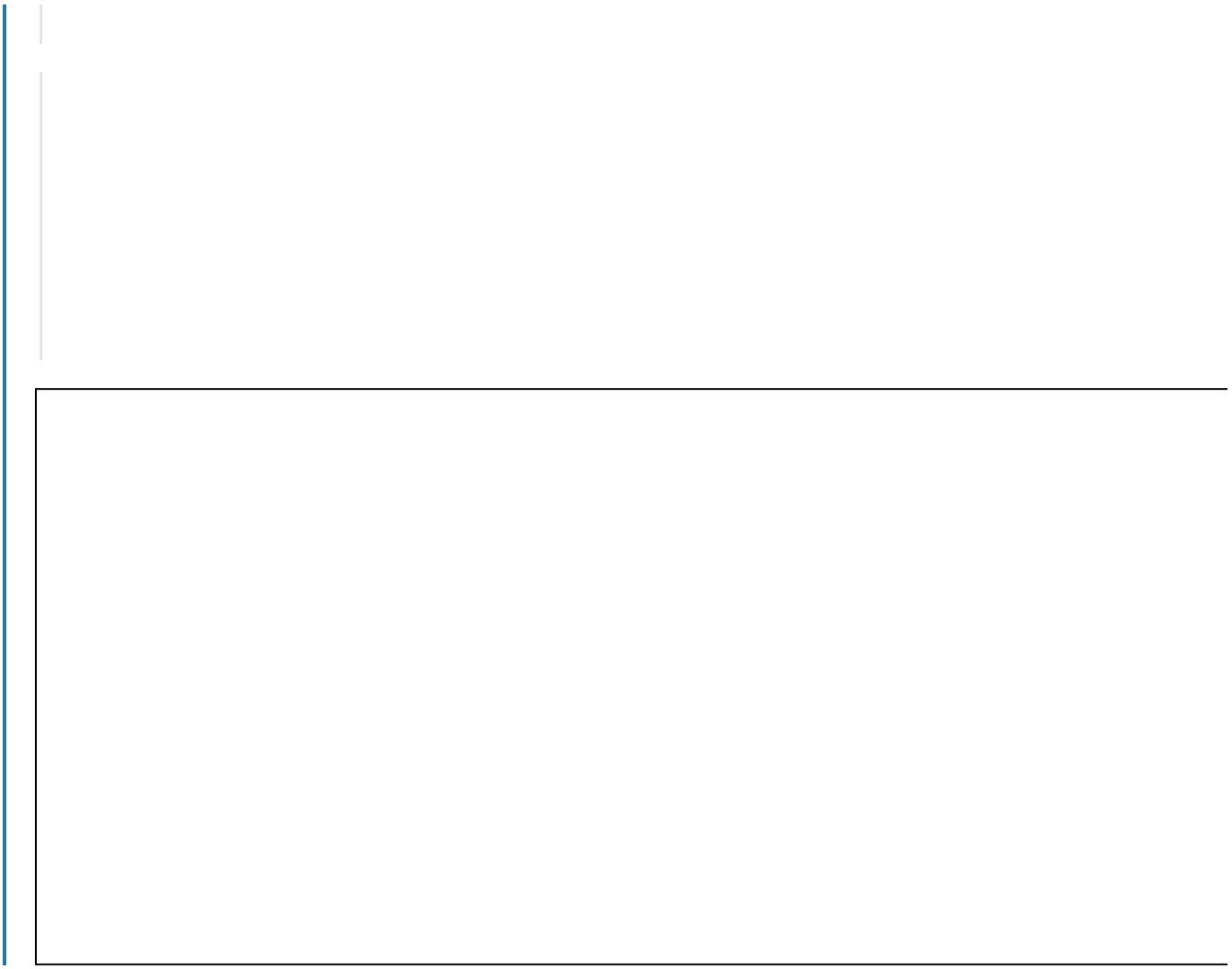
- *doZaszyfrowania* – łańcuch znaków
- *klucz* – łańcuch znaków

Wynik:

- Program wyświetla utworzoną tablicę szyfrującą, a w nowej linii wynik szyfrowania.

Twoje zadania

1. Program ma utworzyć tablicę szyfrowania (tablica 26 x 26 zawierająca znaki alfabetu łacińskiego, jak w poprzednich sekcjach e-materiału) oraz zaszyfrować (według oryginalnej wersji szyfru Vigenère'a) ciąg znaków „tajnytekst”. Za pierwszą literę klucza szyfrującego należy przyjąć literę „J”. **Należy wyświetlić utworzoną tablicę, a następnie w nowej linii wynik szyfrowania.**



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Szyfr polialfabetyczny w języku Java

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

b) na tekstach: porównywania tekstów, wyszukiwania wzorca w tekście metodą naiwną, szyfrowania tekstu metodą Cezara i przestawieniową,

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

V. Przestrzeganie prawa i zasad bezpieczeństwa.

Zakres podstawowy. Uczeń:

3) stosuje dobre praktyki w zakresie ochrony informacji wrażliwych (np. hasła, pin), danych i bezpieczeństwa systemu operacyjnego, objaśnia rolę szyfrowania informacji;

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) omawia znaczenie algorytmów szyfrowania i składania podpisu elektronicznego.

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Prześledzisz, jak szyfrować ciągi znaków za pomocą szyfru Vigenère'a.
- Przeanalizujesz implementację w języku Java szyfru Vigenère'a.
- Wykonasz kilka ćwiczeń sprawdzających wiedzę dotyczącą implementacji szyfru Vigenère'a w języku Java.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;

- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji).

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Szyfr polialfabetyczny w języku Java". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Prosi uczniów, by na podstawie wiadomości zdobytych przed lekcją zaproponowali kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
   Następnie uczniowie w parach analizują Problem 1 z sekcji „Przeczytaj" i piszą program, swoje rozwiązania porównują z zaprezentowanym na filmie.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Animacja". Uczniowie wspólnie zapoznają się z jego treścią. Zapisują ewentualne problemy i pytania. Po czym następuje dyskusja, w trakcie której nauczyciel wyjaśnia niezrozumiałe treści.
3. **Ćwiczenie umiejętności.** Uczniowie, pracując w parach, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność pisanych kodów, porównuje je i omawia wraz z uczniami. Wskazuje najbardziej efektywne rozwiązanie.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia z programowania w Javie, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 2 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).

- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.