Łańcuchy znaków w języku Java

Źródło: Danielle MacInnes, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Przetwarzanie tekstu należy do czynności wykonywanych przez programistów bardzo często. Do środowisk programistycznych dołączane są nierzadko biblioteki ułatwiające wykonywanie takich zadań.

Język Java oddaje do dyspozycji programisty specjalną klasę `String`. Upraszcza ona operowanie ciągami znaków i pozwala wykonywać czynności takie jak łączenie wielu wyrazów w całość, sprawdzanie długości tekstu, zapisywanie go wspak, porównywanie zdań itp. Tymi właśnie zagadnieniami zajmiemy się w tym e-materiale.

Podstawowe informacje na temat przetwarzania ciągów znaków w programach znajdziesz w e-materiale Łańcuchy znaków. Implementacje łańcuchów znaków w pozostałych językach oprogramowania znajdziesz w e-materiałach:

- Łańcuchy znaków w języku C++,
- Łańcuchy znaków w języku Python.

Więcej zadań? Przejdź do e-materiału Łańcuchy znaków – zadania maturalne.

> **Twoje cele**

- Scharakteryzujesz obiekty klasy `String` w języku Java.
- Użyjesz narzędzi służących do wykonywania operacji na ciągach znaków.
- Rozwiążesz zadania związane z przetwarzaniem tekstu za pomocą narzędzi wchodzących w skład klasy `String`.

Przeczytaj

Ciągi znaków w języku Java

Ciągiem znaków nazywamy uporządkowany ciąg elementów typu `char`. Poszczególne znaki ciągu są indeksowane; pierwszy z nich ma numer zero, drugi jeden itd.

W języku Java istnieje specjalna klasa `String` służąca do posługiwania się ciągami znaków. Ponieważ nie omawialiśmy jeszcze programowania obiektowego, poprzestańmy na stwierdzeniu, że klasa `String` jest kolejnym typem danych, do którego dołączono pewne narzędzia.

Klasa pozwala tworzyć obiekty będące odpowiednikami zwykłych zmiennych. Można powiedzieć, że obiekt jest zmienną typu klasa (podobnie jak po wydaniu polecenia `char litera` mówi się, że `litera` jest zmienną typu znakowego). Na razie obiekty będziemy traktować jak zmienne, a klasę `String` jako typ danych.

Zdefiniujmy obiekt (zmienną) klasy `String`:

```
1 String imie = "marcin";
```

W taki sposób utworzyliśmy zmienną `imie`. Jest to obiekt typu `String`; nadaliśmy mu od razu wartość `marcin`.

> **Ważne!**
>
> Pamiętaj o różnicy między znakami apostrofu i cudzysłowu. W apostrofy ujmuje się pojedyncze znaki (np. `'a'`), zaś w cudzysłów całe ich ciągi (np. `"marcin"`).

W języku Java ciągi znaków są niemodyfikowalne. Jeżeli popełnimy błąd podczas definiowania zmiennej `imie` – na przykład zapiszemy słowo Marcin z małej litery – nie możemy wymienić tylko pierwszego znaku. Konieczne jest przypisanie zmiennej nowej, tym razem poprawnej wartości:

```
1 imie = "Marcin";
```

Przypisując wartość obiektowi `imie`, możemy dołączyć do niego kolejny ciąg znaków. Nazywamy to konkatenacją, a cała operacja przypomina sumowanie liczb, ponieważ używamy znaku dodawania (**+**):

```
```

```
imie = "Marcin" + " Kowalski";
```

Zmienna `imie` ma obecnie wartość `Marcin Kowalski`.

Jak widać, klasa `String` ułatwia łączenie fragmentów tekstu. Ma ona także wiele innych zalet. Wspomnieliśmy wcześniej, że do klasy dołączono zestaw narzędzi. Są nimi funkcje, za pomocą których można przetwarzać zmienne tego typu.

Wyobraźmy sobie, że chcemy poznać długość obiektu `imie`. Użyjemy wchodzącej w skład klasy `String` funkcji `length()`:

```
imie.length();
```

W odpowiedzi funkcja `length()` zwróci wartość `15`, ponieważ fraza „Marcin Kowalski" składa się z piętnastu znaków.

Funkcje wbudowane w klasę nazywa się metodami. Aby je wywołać, należy podać nazwę obiektu, na którym ma być wykonana operacja, a następnie wpisać kropkę i nazwę funkcji.

Aby odczytać konkretny element ciągu znaków, używamy metody `charAt(indeks)`, podając indeks jako parametr funkcji:

```
imie.charAt(3);
```

Funkcja zwróci w tym przypadku wartość `c` typu `char`, czyli czwartą literę w ciągu „Marcin Kowalski".

Jak sprawdzić, czy dwa ciągi znaków są identyczne? Mogłoby się wydawać, że należy użyć operatora równości (`==`). Nic bardziej mylnego – służy on do porównywania pojedynczych znaków. Aby zbadać dwa ciągi, używa się kolejnej metody należącej do klasy `String`:

```
ciag1.equals(ciag2);
```

Funkcja `equals()` zwraca wartość `true`, jeżeli obiekty `ciag1` oraz `ciag2` są takie same. W przeciwnym wypadku zwracaną wartością jest `false`.

> **Przykład 1**
>
> Wykorzystamy zdobytą już wiedzę o klasach, obiektach i metodach do napisania programu, który poda liczbę wystąpień litery `litera` we frazie `haslo`.

Przetestujemy jego działanie dla frazy `radosny rabarbar` oraz znaku `r`.

Specyfikacja:

Dane:

- `haslo` – fraza, w której będziemy szukać; ciąg znaków
- `litera` – szukana litera; znak

Wynik:

Program wyświetla liczbę wystąpień litery `litera` we frazie `haslo`.

Zaczniemy od szkicu zapisanego za pomocą pseudokodu:

```
haslo ← "radosny rabarbar"
litera ← 'r'
liczbaWystapien ← 0

dla i = 0, 1, ..., długość(haslo) - 1 wykonuj
  jeżeli haslo[i] = litera:
    liczbaWystapien ← liczbaWystapien + 1

wypisz liczbaWystapien
```

Program ma zbadać kolejne znaki ciągu `haslo`. Jeżeli którykolwiek z nich to `litera`, zmienna `liczbaWystapien` zwiększa się o jeden. Na koniec wyświetlamy jej wartość.

Napiszemy teraz algorytm w języku Java. Najpierw inicjujemy zmienne:

```
String haslo = "radosny rabarbar";
char litera = 'r';
int liczbaWystapien = 0;
```

Następnie otwieramy pętlę `for`, która będzie sprawdzać kolejne elementy ciągu `haslo`:

```
String haslo = "radosny rabarbar";
char litera = 'r';
int liczbaWystapien = 0;

for (int i = 0; i < haslo.length(); i++) {

```

```
6
7 }
```

Długość zmiennej typu `String` poznajemy dzięki metodzie `length()`. Następnie stosujemy instrukcję warunkową, która sprawdza, czy badany znak jest zadaną literą `litera` (i w takim przypadku zwiększa wartość zmiennej `liczbaWystapien`):

```
1 String haslo = "radosny rabarbar";
2 char litera = 'r';
3 int liczbaWystapien = 0;
4
5 for (int i = 0; i < haslo.length(); i++) {
6   if (haslo.charAt(i) == litera) {
7     liczbaWystapien++;
8   }
9 }
```

Do odczytania pojedynczych znaków używamy metody `charAt()`, której parametrem są indeksy kolejnych elementów ciągu. Funkcja ta zwraca wartość typu `char`, a zatem korzystamy ze zwykłego operatora porównania (==), nie zaś z wchodzącej w skład klasy `String` metody `equals()`.

Jeżeli przetwarzana litera jest równa zliczanej literze `litera`, inkrementujemy zmienną `liczbaWystąpien`. Na koniec wyświetlamy jej wartość:

```
1 String haslo = "radosny rabarbar";
2 char litera = 'r';
3 int liczbaWystapien = 0;
4
5 for (int i = 0; i < haslo.length(); i++) {
6   if (haslo.charAt(i) == litera) {
7     liczbaWystapien++;
8   }
9 }
10
11 System.out.println(liczbaWystapien);
```

Słownik

inkrementacja

zwiększenie wartości zmiennej

metoda

funkcja należąca do klasy i służąca do operowania na jej obiektach (zmiennych); metody są wywoływane za pomocą polecenia `obiekt.nazwa_metody()`

palindrom

wyrażenie brzmiące tak samo, gdy jest czytane zarówno od strony lewej do prawej, jak i wspak (np. „a kilku tu klika”)

Schemat interaktywny

Polecenie 1

Napisz program, który sprawdzi, czy podany przez użytkownika wyraz `slowo` jest palindromem.

Przetestuj program dla słowa `owocowo`.

Specyfikacja:

Dane:

- `slowo` – sprawdzane słowo; łańcuch znaków

Wynik:

Komunikat „Słowo `slowo` jest palindromem” albo „Słowo `slowo` nie jest palindromem”.

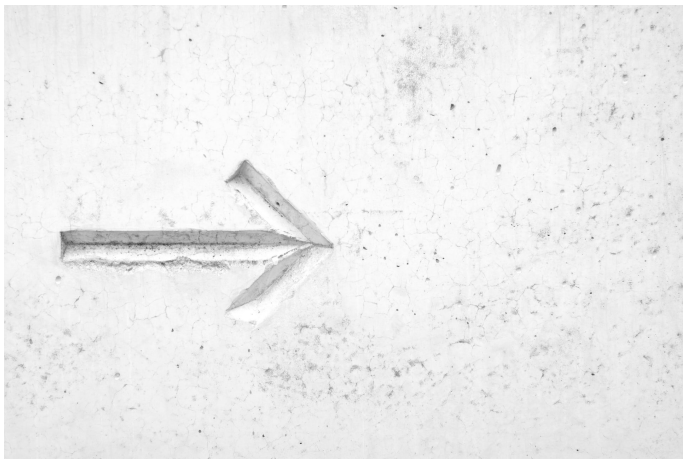
```
public class Main {
    public static void main (String [] args) {
        String slowo = "owocowo"; // dla testów zmień słowo
        boolean palindrom = true;

        // tutaj dopisz kod

        if (palindrom == true) {
            System.out.println("Słowo " + slowo + " jest palindromem.");
        } else {
            System.out.println("Słowo " + slowo + " nie jest palindromem.");
        }
```

```
```

Polecenie 2

Porównaj swoje rozwiązanie z prezentacją.

1



Źródło: Hello I'm Nik, unsplash.com, CC0.

Napiszemy program, który sprawdzi, czy pewne słowo jest palindromem.

2

Palindrom jest frazą brzmiącą tak samo, gdy jest czytana od strony lewej do prawej i odwrotnie. Przykładowo, palindromem jest słowo „kajak”.

3

Skoro słowo czytane w obie strony jest takie samo, to identyczne są w nim pary liter: pierwsza i ostatnia, druga i przedostatnia, trzecia i trzecia od końca itd.

4

Powinniśmy zatem porównywać kolejne pary znaków składających się na słowo. Jeżeli wszystkie okażą się identyczne, wyraz jest palindromem.

5

Pisanie programu zaczniemy od zdefiniowania zmiennych:

```
1 String slowo = "owocowo";
2 boolean palindrom = true;
```

Zakładamy wstępnie, że obiekt `slowo` jest palindromem. Jeżeli znajdziemy dowód na to, że pomyliliśmy się, zmienimy wartość logiczną `palindrom` na `false`. Dowodem będzie znalezienie pary liter, które nie są takie same.

6

Kolejnym etapem jest porównywanie par znaków składających się na ciąg. Liczba tych operacji jest równa połowie długości obiektu `slowo`, ponieważ za każdym razem sprawdzamy dwa elementy:

```
1 String slowo = "owocowo";
2 boolean palindrom = true;
```

```
3
4 for (int i = 0; i <
  slowo.length() / 2; i++) {
5   // Sprawdź, czy para
  znaków jest taka sama
6 }
```

7

Jeżeli którakolwiek para będzie składać się z różnych znaków, to wyraz nie jest palindromem. Musimy wówczas przypisać zmiennej `palindrom` wartość `false`:

```
1 String slowo = "owocowo";
2 boolean palindrom = true;
3
4 for (int i = 0; i <
  slowo.length() / 2; i++) {
5   if (slowo.charAt(i) !=
  slowo.charAt(slowo.length(
  ) - i - 1)) {
6     palindrom = false;
7   }
8 }
```

8

Za każdym razem porównujemy znak o indeksie `i` z elementem oddalonym od końca słowa o `i` miejsc. Ma on indeks `slowo.length() - i - 1` (jak pamiętamy, numerowanie elementów ciągu zaczyna się od zera, dlatego musimy odjąć 1).

9

Po zakończeniu pętli sprawdzamy wartość zmiennej `palindrom`. Możemy już

poinformować, czy badany wyraz jest palindromem. Aby wyświetlić rozbudowany komunikat, zastosujemy konkatenację:

```
String slowo = "owocowo";
boolean palindrom = true;

for (int i = 0; i < slowo.length() / 2; i++) {
  if (slowo.charAt(i) != slowo.charAt(slowo.length() - i - 1)) {
    palindrom = false;
  }
}

if (palindrom == true) {
  System.out.println("Słowo " + slowo + " jest palindromem.");
} else {
  System.out.println("Słowo " + slowo + " nie jest palindromem.");
}
```

10

Źródło: Edge2Edge Media, unsplash.com, CC0.

Oto gotowy kod programu:

```
public class Main {
```

```
    public static void main (String [] args) {
        String slowo = "owocowo";
        boolean palindrom = true;

        for (int i = 0; i < slowo.length() / 2; i++) {
            if (slowo.charAt(i) != slowo.charAt(slowo.length() - i - 1)) {
                palindrom = false;
            }
        }

        if (palindrom == true) {

System.out.println("Słowo " + slowo + " jest palindromem.");
        } else {

System.out.println("Słowo " + slowo + " nie jest palindromem.");
        }
    }
}
```


Jeśli chcesz dowiedzieć się więcej na temat palindromów, przejdź do e-materiału Palindromy.

Ćwiczenie 1

Wykorzystując schemat interaktywny, zapisz program, który sprawdzi, czy dwa łańcuchy znaków są takie same. Program powinien wyświetlać komunikat `tak`, jeśli są takie same, oraz `nie` jeśli nie są takie same. Przetestuj działanie programu dla następujących danych:

```
slowo1 = aaababbababaab
slowo2 = aaabbababababaab
```

Specyfikacja problemu: *Dane:*

- `slowo1` – łańcuch znaków
- `slowo2` – łańcuch znaków

Wynik:

- `czyTakieSame` – łańcuch znaków

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który sprawdzi, czy łańcuch znaków `slowo1` jest taki sam jak `slowo2`. Przetestuj działanie programu dla łańcuchów znaków `abcdefghijkl123` oraz `abcdefghijkl123`.

Specyfikacja:

Dane:

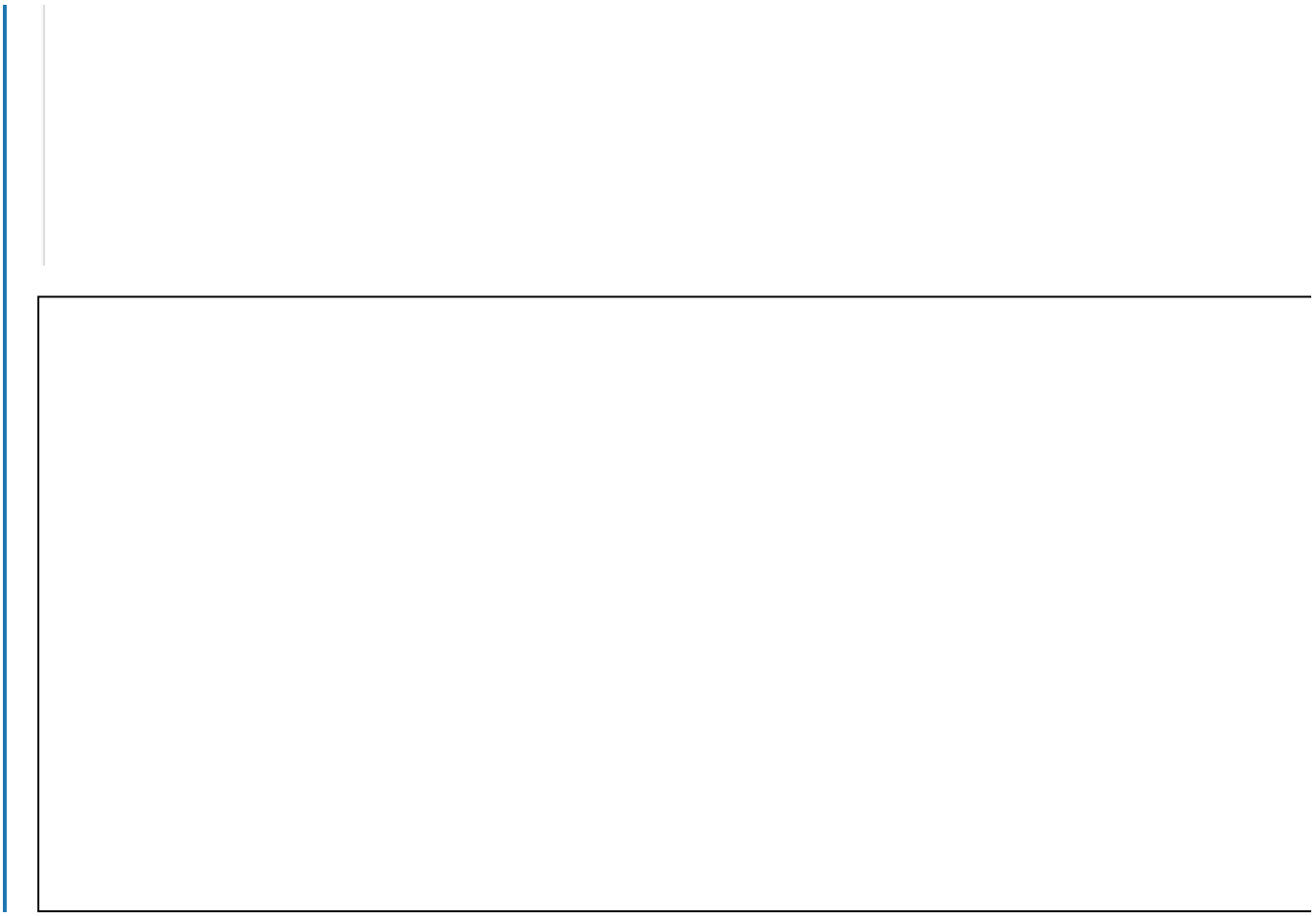
- `slowo1` – łańcuch znaków
- `slowo2` – łańcuch znaków

Wynik:

Program wyświetla komunikat `tak` lub `nie` w zależności od tego, czy `slowo1` jest takie samo jak `slowo2`.

Twoje zadania

1. Program sprawdza, czy słowa są identyczne.



Ćwiczenie 2

Napisz program, który policzy, ile razy dany znak `litera` pojawił się w łańcuchu znaków `slowo`.

Przetestuj działanie programu dla słowa `katarynka` oraz litery `k`.

Specyfikacja:

Dane:

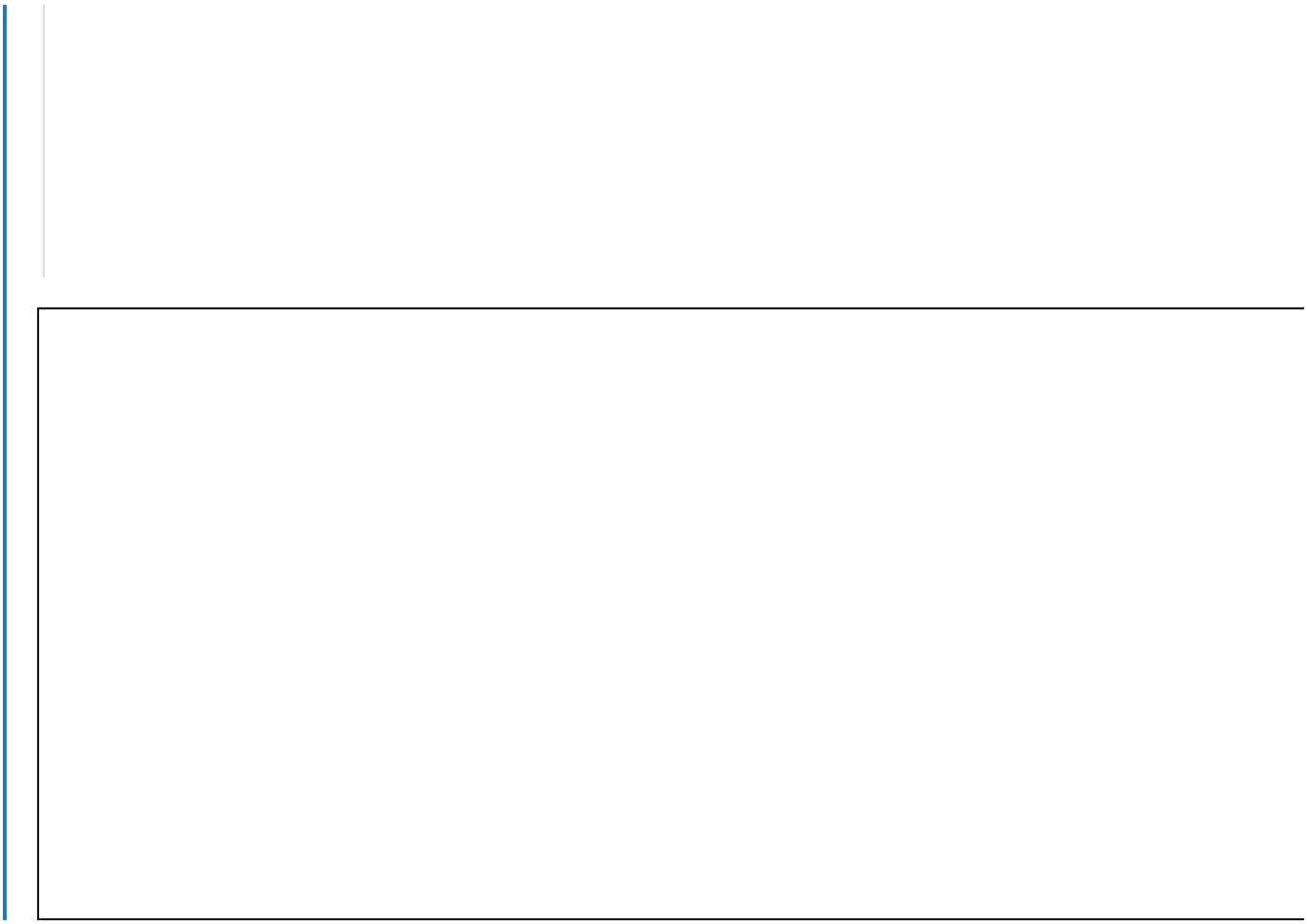
- `litera` – zliczana litera; znak
- `slowo` – łańcuch znaków

Wynik:

Program wypisuje liczbę wystąpień litery `litera` w słowie `slowo`.

Twoje zadania

1. Program zlicza wystąpienia litery w słowie.



Ćwiczenie 3

Napisz program, który odwraca kolejność liter w wyrazie `slowo` i wyświetla na ekranie słowo zapisane wspak.

Przetestuj działanie programu dla słowa `informatyka`.

Specyfikacja:

Dane:

- `slowo` – wyraz do odwrócenia; łańcuch znaków

Wynik:

Program wypisuje odwrócone wspak słowo `slowo`.

Twoje zadania

1. Program zapisuje słowo wspak.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Łańcuchy znaków w języku Java

Grupa docelowa:

Liceum ogólnokształcące i technikum, 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:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

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,

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

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

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

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

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

2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:

c) znajdowania w ciągu podciągów o różnorodnych własnościach, np. najdłuższego spójnego podciągu niemalejącego, spójnego podciągu o największej sumie,

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):

- Scharakteryzujesz obiekty klasy `String` w języku Java.
- Użyjesz narzędzi służących do wykonywania operacji na ciągach znaków.
- Rozwiążesz zadania związane z przetwarzaniem tekstu za pomocą narzędzi wchodzących w skład klasy `String`.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

- ćwiczenia praktyczne.

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: „Łańcuchy znaków w języku Java". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć 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".
   Uczniowie powtarzają przykład 1 na swoich komputerach.
2. Nauczyciel wyświetla zawartość sekcji „Schemat interaktywny". Uczniowie w parach rozwiązują polecenie 1, następnie porównują swoje wyniki z prezentacją.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1-2 z sekcji „Sprawdź się" . Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie”. W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.

Praca domowa:

1. Uczniowie wykonują ćwiczenie 3 z sekcji „Sprawdź się”.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać multimedia z sekcji: „Przeczytaj”, „Schemat interaktywny”, „Sprawdź się” do przygotowania się do lekcji powtórkowej.