Palindromy w języku Java

Źródło: Wokandapix, domena publiczna.

Wiesz już, czym jest palindrom i znasz algorytm sprawdzający, czy dany wyraz spełnia warunki tego pojęcia – informacje te znajdziesz w e-materiale Palindromy.

Tym razem wspomniany algorytm zaimplementujemy w języku Java. Następnie porównamy algorytm napisany w języku Java ze znanym ci już zapisanym w języku C++. Dzięki temu lepiej poznasz oba języki programowania.

Implementację programu sprawdzającego, czy dane słowo jest palindromem, w pozostałych językach programowania znajdziesz w e-materiałach:

- Palindromy w języku C++,
- Palindromy w języku Python.

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

Twoje cele

- Zaimplementujesz algorytm sprawdzający, czy dany ciąg znaków jest palindromem.
- Przeanalizujesz algorytm zaimplementowany w języku Java.
- Scharakteryzujesz wybrane algorytmy tekstowe.

Film samouczek

Problem 1

Napisz program sprawdzający, czy dany ciąg znaków `test` o długości n jest palindromem. Przetestuj jego działanie dla następującego ciągu znaków:

```
Może jutro dama da tortu jeżom
```

Specyfikacja:

Dane:

- `test` – ciąg znaków o długości $n \geq 2$

Wynik:

- Na standardowym wyjściu program wyświetla komunikat `Napis jest palindromem` lub `Napis nie jest palindromem`

Twoje zadania

1. Program ma sprawdzić, czy podany ciąg znaków jest palindromem i wyświetlić komunikat "Napis jest palindromem" lub „Napis nie jest palindromem”

```java
public class Palindrom {

    // tutaj zdefiniuj funkcje

    public static void main(String[] args) {
        String test = "Może jutro dama da tortu jeżom";

        // tutaj dopisz kod

    }
}
```

```
1
```

Polecenie 1

Porównaj swoje rozwiązanie z filmem.

Film dostępny pod adresem /preview/resource/R1KBFK7rlBxOD

Film nawiązujący do treści materiału: Algorytmy tekstowe - Palindromy w języku JAVA.

Kod programu zaprezentowanego w filmie

Plik o rozmiarze 1.21 KB w języku polskim

Polecenie 2

Przeanalizuj implementację algorytmu sprawdzającego, czy podane słowo jest palindromem. Jest to przedstawienie, krok po kroku, II sposobu.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Twoim zadaniem jest napisanie programu sprawdzającego, czy podane przez użytkownika słowo jest palindromem (II sposób).

Do napisania tego programu wykorzystamy umiejętności nabyte w innych materiałach.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Zacznijmy od zbudowania szkieletu programu, czyli zadeklarowania klasy i głównej funkcji programu – `main`. To w niej będzie się znajdować część naszego programu.

```
1 public class Palindrom {
2     public static void
  main(String[] args) {
3
```

```
    }
}
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Następnie stworzymy szkielet funkcji `czyPalindrom()`. Jej typem będzie zmienna logiczna `boolean`, która zwróci wartość `true`, jeżeli podane słowo będzie palindromem, a w przeciwnym wypadku `false`.

```
public class Palindrom {
    public static boolean czyPalindrom() {

    }

    public static void main(String[] args) {

    }
}
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Nasza funkcja będzie przyjmowała na wejście jeden argument – `slowo`, czyli ciąg znaków, który będzie sprawdzany. Argument będzie typu `String`.

```
public class Palindrom {
    public static boolean czyPalindrom(String slowo)
```

```
{

    }

    public static void
main(String[] args) {

    }
}
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Zacznijmy od określenia zmiennej typu `String` `odTylu`. Będziemy ją później porównywać z argumentem po to, by sprawdzać, czy jest to palindrom.

```
public class Palindrom {
    public static boolean
czyPalindrom(String slowo)
{
        String odTylu =
"";
    }

    public static void
main(String[] args) {

    }
}
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Napiszmy teraz pętlę `for`, w której do nowo utworzonej zmiennej będziemy zapisywać, po literze, podane słowo (w argumencie) od tyłu.

```
1  public class Palindrom {
2      String odTylu = "";
3
4      public static boolean
   czyPalindrom(String slowo)
   {
5          String odTylu =
   "";
6
7          for (int i =
   slowo.length() - 1; i >=
   0; i--) {
8              odTylu =
   odTylu + slowo.charAt(i);
9          }
10     }
11
12     public static void
   main(String[] args) {
13
14     }
15 }
```

Dzięki funkcji wbudowanej `charAt()` mamy dostęp do konkretnej litery słowa.

Zwróćmy uwagę na ten fragment kodu `odTylu = odTylu + slowo.charAt(i)`. Ta linijka odpowiedzialna jest za tworzenie nowego słowa za pomocą konkatenacji kolejnych liter zmiennej `slowo`. Użycie znaku "+" w tym przypadku powoduje dodawanie, do prawej strony, do zmiennej `String odTylu` kolejnych liter zmiennej `slowo`.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Następnym krokiem będzie stworzenie warunku, który, gdy podane słowo będzie równe temu zapisanemu od tyłu, zwróci wartość `true`. W przeciwnym przypadku zwróci wartość `false`

```
1  public class Palindrom {
2      String odTylu = "";
3
4      public static boolean
   czyPalindrom(String slowo)
   {
5          String odTylu =
   "";
6
7          for (int i =
   slowo.length() - 1; i >=
   0; i--) {
8              odTylu =
   odTylu + slowo.charAt(i);
9          }
10
11         if
   (slowo.equals(odTylu)) {
12             return true;
13         } else {
14             return false;
15         }
16     }
17
18     public static void
   main(String[] args) {
19
20     }
21 }
```

Funkcja `equals()` – służy do porównania ze sobą obiektów.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Nasza funkcja `czyPalindrom()` jest gotowa. Przejdźmy teraz do funkcji głównej programu – `int main()`. Stwórzmy obiekt klasy `Scanner`, który umożliwi użytkownikowi podanie słowa do sprawdzenia.

```
1  public class Palindrom {
2      String odTylu = "";
3
4      public static boolean
   czyPalindrom(String slowo)
   {
5          String odTylu =
   "";
6
7          for (int i =
   slowo.length() - 1; i >=
   0; i--) {
8              odTylu =
   odTylu + slowo.charAt(i);
9          }
10
11         if
   (slowo.equals(odTylu)) {
12             return true;
13         } else {
14             return false;
15         }
16     }
17
18     public static void
   main(String[] args) {
19         Scanner scanner =
   new Scanner(System.in);
20
21
    System.out.println("Podaj
   słowo do sprawdzenia");

22     }
23 }
```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Utworzymy teraz zmienną typu `String` o nazwie `slowoTest`, do której za pomocą funkcji wbudowanej `nextLine()` zapiszemy to, co podał użytkownik.

```
1  public class Palindrom {
2      String odTylu = "";
3
4      public static boolean
   czyPalindrom(String slowo)
   {
5          String odTylu =
   "";
6
7          for (int i =
   slowo.length() - 1; i >=
   0; i--) {
8              odTylu =
   odTylu + slowo.charAt(i);
9          }
10
11         if
   (slowo.equals(odTylu)) {
12             return true;
13         } else {
14             return false;
15         }
16     }
17
18     public static void
   main(String[] args) {
19         Scanner scanner =
   new Scanner(System.in);
20
21
    System.out.println("Podaj
   słowo do sprawdzenia");
22
23         String slowoTest =
   scanner.nextLine();
```

```
24      }
25  }
```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph3OtUa5N

Program już prawie gotowy. Zaprogramujemy teraz interakcję aplikacji z użytkownikiem, czyli komunikaty, jakie ma wyświetlać program w przypadku gdy funkcja `czyPalindrom()` przyjmie wartość `true` i `false`.

Oto wersja finalna programu.

```
1  public class Palindrom {
2      String odTylu = "";
3
4      public static boolean
   czyPalindrom(String slowo)
   {
5          String odTylu =
   "";
6
7          for (int i =
   slowo.length() - 1; i >=
   0; i--) {
8              odTylu =
   odTylu + slowo.charAt(i);
9          }
10
11         if
   (slowo.equals(odTylu)) {
12             return true;
13         } else {
14             return false;
15         }
16     }
17
18     public static void
   main(String[] args) {
```

```
19          Scanner scanner =
    new Scanner(System.in);
20
21
     System.out.println("Podaj
    słowo do sprawdzenia");
22
23          String slowoTest =
    scanner.nextLine();
24
25          if
    (czyPalindrom(slowoTest)
    == true) {
26
     System.out.println("Wyraz
    jest palindromem");
27          } else {
28
     System.out.println("Wyraz
    nie jest palindromem");
29          }
30      }
31  }
```

Przeczytaj

Co to jest palindrom?

Palindrom to słowo lub ciąg wyrazów, które zapisane i czytane od lewej do prawej strony brzmią tak samo, jak zapisane i czytane od prawej do lewej. Przykładami palindromów są:

- łapał za kran, a kanarka złapał,
- wół utył i ma miły tułów.

Poznaliśmy już dwa sposoby implementacji algorytmu sprawdzającego, czy dane słowo jest palindromem. Zapiszemy obie implementacje i dokonamy porównania, by ocenić, która z nich jest efektywniejsza.

Komputerowa realizacja w języku Java

Przed rozpoczęciem implementacji algorytmu sprawdzającego, czy dane słowo jest palindromem, warto przypomnieć kilka aspektów związanych ze składnią języka Java.

- **pętla** – to w niej będziemy porównywać słowa indeks po indeksie,
- **funkcja wbudowana** `charAt()` umożliwia dostęp do pojedynczego znaku typu `String`, co okaże się niezwykle przydatne,
- **funkcje** – najpierw napiszemy program, wykorzystując główną funkcję programu `main()`, a następnie dokonamy modyfikacji i wykonamy osobną funkcję do rozwiązania tego problemu,
- **operacje wejścia/wyjścia** – w szczególności użycie klasy Scanner, która umożliwi użytkownikowi samodzielne wpisanie słowa do sprawdzenia w naszym programie.

I sposób

Zaimplementujemy sposób z porównywaniem znaków o skrajnych indeksach. Przypomnijmy, na czym on polegał: słowo, które jest palindromem, napisane od przodu i od tyłu wygląda tak samo. Oznacza to, że konkretne pary znaków muszą być takie same: pierwsza i ostatnia, druga i przedostatnia itp. Zacznijmy od napisania kodu, który umożliwi użytkownikowi wpisanie własnego słowa w celu sprawdzenia, czy jest ono palindromem. Zrobimy to z pomocą klasy Scanner. Przeanalizujmy następujący fragment kodu:

```
1 Scanner scanner = new Scanner(System.in);
2
3 System.out.println("Podaj słowo do sprawdzenia");
```

```
4
5 String slowoTest = scanner.nextLine();
```

Najpierw deklarujemy obiekt klasy Scanner – tu został on nazwany scanner. Standardowe wejście `System.in` pozwala na sczytywanie tego, co wpisujemy z klawiatury. Następnie tworzymy zmienną typu `String` o nazwie `slowoTest`, do której zapisujemy to, co podał użytkownik za pomocą funkcji wbudowanej `nextLine()`. Jej dokładne działanie polega na pobieraniu tego, co się wpisuje aż do wciśnięcia przycisku `Enter` na klawiaturze.

Kolejnym krokiem będzie zadeklarowanie pętli, w której będą porównywane skrajne znaki.

```
1 for (int i = 0, j = slowoTest.length() - 1; i < slowoTest.length(
2
3 }
```

Jest to nietypowa pętla `for`. Czy widzisz dlaczego?

Posiada ona dwie zmienne sterujące: `i` oraz `j`. Pozwala to na jednoczesne porównywanie liter danego słowa. Zmienna `i` startuje od pierwszego indeksu słowa i zwiększa się, natomiast zmienna `j` zaczyna od ostatniego i zmniejsza się. To w zasadzie jeden z najważniejszych aspektów w implementacji tego sposobu.

Wypełnijmy teraz blok tej pętli i dodajmy zmienną `boolean czyPalindrom`.

```
1  boolean czyPalindrom = false;
2
3  for (int i = 0, j = slowoTest.length() - 1; i < slowoTest.length(
4      if (slowoTest.charAt(i) != slowoTest.charAt(j)) {
5          czyPalindrom = false;
6          break;
7      } else {
8          czyPalindrom = true;
9      }
10 }
```

Dodaliśmy blok `if-else`, który w przypadku niewykrycia zgodności liter od razu przerywa pętlę (słowo kluczowe `break`). W przeciwnym razie, jeżeli po przejściu całej pętli, zmiennej logicznej `czyPalindrom` nie zostanie przypisana wartość `false`, to oznacza, że wprowadzone słowo na pewno jest palindromem.

Ostatnią częścią naszego programu będzie wypisanie użytkownikowi komunikatu tekstowego powiadamiającego, czy słowo jest palindromem, czy nie.

```
if (czyPalindrom == true) {
    System.out.println("Wyraz jest palindromem");
} else {
    System.out.println("Wyraz nie jest palindromem");
}
```

Zobaczmy teraz cały program.

```
import java.util.Scanner;

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

        Scanner scanner = new Scanner(System.in);

        System.out.println("Podaj słowo do sprawdzenia");

        String slowoTest = scanner.nextLine();
        boolean czyPalindrom = false;

        for (int i = 0, j = slowoTest.length() - 1; i = 0; i++, j
            if (slowoTest.charAt(i) != slowoTest.charAt(j)) {
                czyPalindrom = false;
                break;
            } else {
                czyPalindrom = true;
            }
        }

        if (czyPalindrom == true) {
            System.out.println("Wyraz jest palindromem");
        } else {
            System.out.println("Wyraz nie jest palindromes");
        }
    }
}
```

Wersja z funkcją

Zobaczmy, jak wygląda cały program, ale napisany przy pomocy funkcji.

```
import java.util.Scanner;

public class Palindrom {
    public static boolean czyPalindrom(String slowo) {
        for (int i = 0, j = slowo.length() - 1; i < slowo.length(
            if (slowo.charAt(i) != slowo.charAt(j)) {
                return false;
            }
        }

        return true;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Podaj słowo do sprawdzenia");

        String slowoTest = scanner.nextLine();

        if (czyPalindrom(slowoTest) == true) {
            System.out.println("Wyraz jest palindromem");
        } else {
            System.out.println("Wyraz nie jest palindromem");
        }
    }
}
```

Program napisany przy pomocy funkcji posiada więcej zalet w porównaniu do wcześniejszej implementacji. Niepotrzebna nam jest dodatkowa zmienna `boolean czyPalindrom`. Teraz cała funkcja zwraca typ logicznego `boolean`. Dodatkowo, gdybyśmy chcieli sprawdzić kilka słów, w poprzedniej implementacji musielibyśmy powielać ten sam kod – tu wystarczy wywołać jeszcze raz funkcję i przekazać sprawdzane słowo jako parametr.

II sposób

Drugi sposób polega na tym, że w nowej zmiennej zapisujemy słowo, które podamy do sprawdzenia – od tyłu. Jeżeli oba słowa są równe, to wyraz ten jest palindromem.

W następnej sekcji zaimplementujemy algorytm i dokładnie go omówimy.

Słownik

> **funkcja wbudowana**
>
> gotowa funkcja mająca swoje zastosowanie, należy do jednej z bibliotek lub klas; przykładami takich funkcji z języka Java są: `length()`, `charAt()`, `nextLine()`
> **zmienna sterująca**
>
> zmienna umożliwiająca sterowanie pętlą; pozwala na przejście do kolejnych iteracji (iterator)

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Przedstawiony kod to niepełna implementacja algorytmu, który sprawdza, czy podane słowo jest palindromem. Uzupełnij brakujące linijki kodu w funkcji `czyPalindrom()`.

Specyfikacja:

Dane:

- `slowo` – ciąg znaków o długości $n \geq 2$

Wynik:

Program wyświetla komunikat `Wyraz jest palindromem` lub `Wyraz nie jest palindromem`.

Twoje zadania

1. Program sprawdza, czy podany wyraz jest palindromem i zwraca komunikat "Wyraz jest palindromem"

Ćwiczenie 2

Zaimplementuj I sposób sprawdzania, czy podane `zdanie jest palindromem` (bez używania dodatkowej zmiennej `odTylu`). Spacje i znaki interpunkcyjne powinny być przez twój program pomijane. Załóż, że podane zdanie nie będzie zawierało wielkich liter. Działanie programu przetestuj dla zdania `a wart wór kota? to krów trawa!`

Specyfikacja:

Dane:

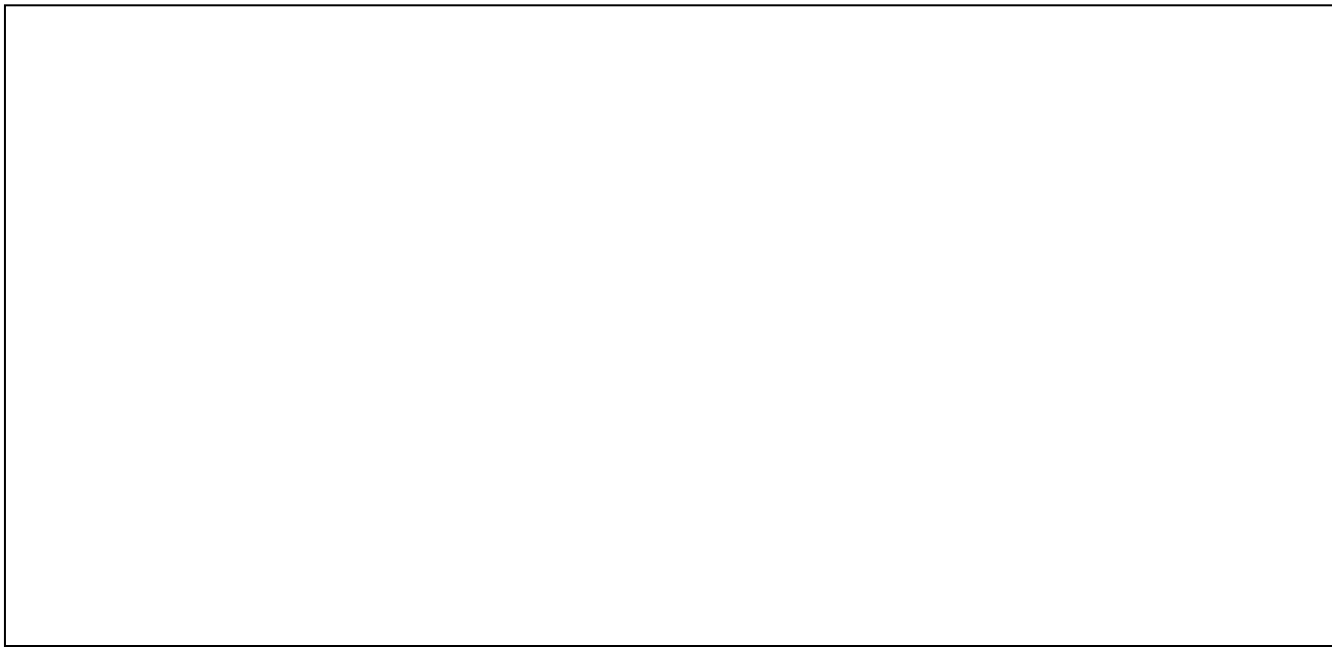
- `zdanie` – ciąg znaków o długości $n \geq 2$

Wynik:

Program wyświetla komunikat `Zdanie nie jest palindromem` lub `Zdanie jest palindromem`

Twoje zadania

1. Program ma sprawdzać, czy podane zdanie jest palindromem. Zastosuj sposób bez użycia dodatkowej zmiennej `odTylu`.



Ćwiczenie 3

Pewna firma z branży lotniczej miała problem z przekłamaniami transmisji danych – zdarzało się, że urządzenie nadawcze wysyłało bit 1, który jednak był interpretowany przez odbiornik jako 0. Uznano, że rozwiązaniem tego problemu będzie zastosowanie następującego kodu: po każdych ośmiu bitach nadawane są te same bity, ale w odwrotnej kolejności. Dzięki takiemu rozwiązaniu można określić, czy otrzymane dane są poprawne. Napisz program, który określi, czy podany ciąg zer i jedynek jest poprawnym kodem (ma poprawną strukturę oraz poprawną długość). Dla prawidłowych kodów powinien drukować wiadomość `Poprawny kod`, a dla nieprawidłowych: `Niepoprawny kod`. Przetestuj jego działanie dla ciągu bitów `1001110110111001`.

Specyfikacja:

Dane:

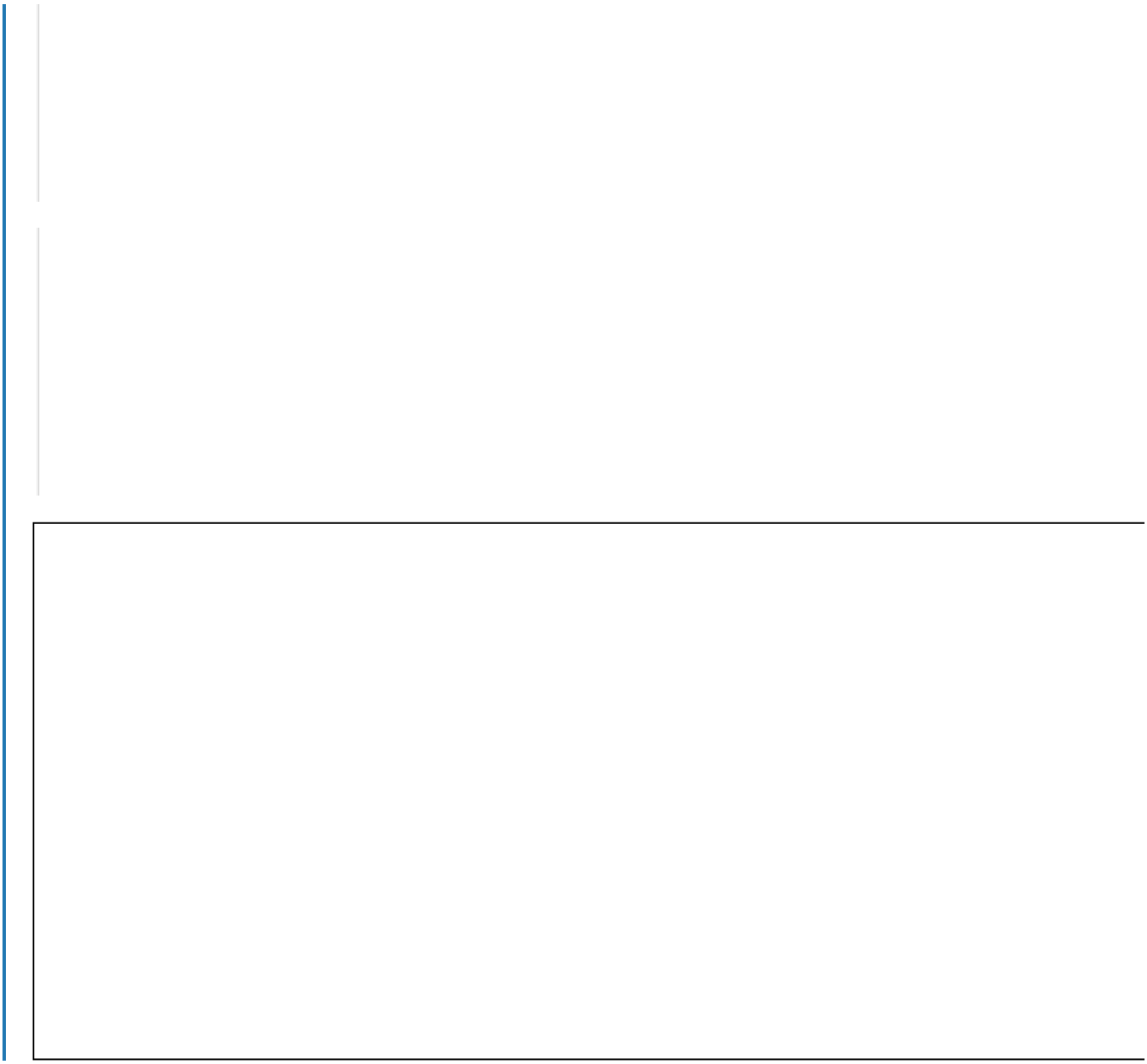
- `dane` – łańcuch znaków o długości 16; ciąg bitów składający się z dwóch połączonych ciągów bitów; pierwszy to nadane osiem bitów, a drugi to ciąg tych samych 8 bitów w odwróconej kolejności

Wynik:

Program wyświetla informację na temat tego, czy kod *dane* jest poprawny, czy nie. Wyświetla komunikat `Poprawny kod` lub `Niepoprawny kod`.

Twoje zadania

1. Program drukuje wiadomość "Poprawny kod" na standardowe wyjście dla kodu `1001110110111001`.



Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Palindromy w języku Java

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

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

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,

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

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):

- Zaimplementujesz algorytm sprawdzający, czy dany ciąg znaków jest palindromem.
- Przeanalizujesz algorytm zaimplementowany w języku Java.

- Scharakteryzujesz wybrane algorytmy tekstowe.

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: „Palindromy w języku Java". Uczniowie zapoznają się z treściami w sekcji „Film samouczek".

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z multimedium.** Uczniowie zapoznają się indywidualnie z treścią sekcji „Przeczytaj" i poleceniem nr 1, zapisują problemy i pytania z nią związane. Po czym następuje dyskusja, w trakcie której nauczyciel wyjaśnia niezrozumiałe kroki procedury.Pozostając w sekcji „Film samouczek" uczniowie w zespołach dwuosobowych zapoznają się z treścią polecenia nr 2 i wspólnie analizują kolejne kroki rozwiązania postawionego problemu.
2. **Ćwiczenie umiejętności.** Poszukiwanie najefektywniejszego rozwiązania problemu. Uczniowie wykonują w parach ćwiczenie nr 1 z sekcji „Sprawdź się", a następnie porównują swój kod omawiając go wspólnie na forum. Nauczyciel ocenia efektywność zastosowanego rozwiązania.
3. Liga zadaniowa – uczniowie pracując w parach, wykonują ćwiczenie nr 2 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku Java.

Praca domowa:

1. Uczniowie proponują alternatywny sposób rozwiązania problemu postawionego w sekcji „Przeczytaj".

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 „Film samouczek" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.