String

Programowanie nie skupia się tylko i wyłącznie na liczbach. Programiści wykorzystują wiele różnych typów danych. Jednym ze złożonych typów danych jest właśnie klasa *String*. W przybliżeniu to ciąg znaków, inaczej typów prostych. Podobnie jak wyrazy w języku polskim też składają się z ciągu liter, tak w przybliżeniu *String* składa się z ciągu znaków *char*. *String* jest obiektem w Javie, który posiada wiele metod. Aby utworzyć nowy *String,* można użyć konstrukcji:
String word = "word";
String sentence = new String("sentence");

Operacje na Stringu
Chyba najprostszą operacją, od której każdy początkujący programista powinien zacząć, jest konkatenacja, czyli dodawanie *String*, inaczej łączenie wyrazów.
String welcome = "Hello";
String world = "World";
String greetings = welcome + " " + world;
System.out.println(greetings); // "Hello World"
W powyższym przykładzie widzimy, że znak '+' dodawania łączy wyrazy ze sobą. Można przemiennie łączyć już zdefiniowane Stringi jak na przykład 'welcome' oraz 'world' z aktualnie tworzonymi czyli w tak zwanych „ciapkach" („"). W naszym przykładzie utworzyliśmy dynamicznie odstępnik – znak spacji, by oddzielić wyrazy.
*niezapominajmyoodstępnikach =]
Aby odejmować części literek, bądź wyrazów ze zdań niestety nie można używać znaku '-' odejmowania. Musimy odwołać się do **metod** zawartych już w naszej klasie *String*. O metodach będzie innym razem bo to obszerny temat. Teraz wystarczy wiedzieć, że **klasa String posiada szereg zdefiniowanych metod**. Jedną z takich metod jest metoda '**substring**' oznaczająca odejmowanie.
Aby wywołać metodę danej klasy trzeba postawić '.' kropkę na końcu nazwy zmiennej (więcej o metodach znajdziecie tutaj). Metodę substring można użyć na dwa sposoby. Sposób krótszy to w nawiasie umieścić jedną cyfrę oznaczającą miejsce w ciągu znaków w którym dojdzie do „odcięcia" pozostałych znaków:
String greetings = "Hello World";
System.out.println(greetings.substring(5)); // " World"
System.out.println(greetings.substring(0, 5)); // "Hello"
Na powyższym przykładzie widzimy zapis *greetings.substring(5)*, który oznacza : „odetnij po piątym znaku licząc od początku". Odcina nam początek „0H-1e-2l-3l-4o-5(wycina)". Pozostawiając nam „ World". Kolejny sposób to napisanie w nawiasie dwóch cyfr oznaczających początek cięcia i koniec cięcia. Przykład *greetings.substring(0, 5)* znaczy „ zacznij wycinać od samego początku a zakończ na 5 literze" co da nam: „ Hello". Inna wariacja to np:
System.out.println(greetings.substring(2, 8)); // "llo Wo"

Omawiany wcześniej znak '+' konkatenacji to również metoda w klasie *String*. A wywołać ją można poprzez zapis:
String greetings = "Hello World";

System.out.println(greetings.koncat(" again!")); // "Hello World again!"

Zapis *greetings.koncat(" again!")* oznacza tyle co *greetings* + *" again!"*. Java interpretuje znak '+' w przypadku *String* jak konkatenacja a w przypadku liczbowych typów danych jako dodawanie (było to omawiane na wpisie odnośnie Zmiennych).
*koncat jest niepoprawne i celowo zaznaczone jest na czerwono by zwrócić waszą uwagę, taki *mind trick*. Prawidłowo powinno być: '**concat**'. Mam nadzieję, że zapamiętacie =]

Kolejną bardzo pomocną operacją jest określenie długości Stringa poprzez wywołanie metody ***.length()***:

```
String greetings = "Hello World";
System.out.println(greetings.length()); // 11
```

Bardzo pomocna rzecz w przypadku stosowania iteracji jest metoda ***.charAt()***, pokazująca nam jaka litera znajduje się na którym miejscu w ciągu znaków:

```
String greetings = "Hello World";
System.out.println(greetings.charAt(1)); // 'e'
```

Metody *.charAt()* i *.length()* są użyteczne w operacjach logicznych gdy chcemy uzyskać na przykład odpowiednie hasło:

```
String password = "tajne";
if (password.length()>8 && password.charAt(0)=='A'){
   System.out.println("Good password");
}
System.out.println("Bad password");
```

W powyższym przykładzie uzyskamy wiadomość "Bad password", ponieważ długość hasła nie jest większa od 8 oraz pierwsza literka to nie 'A'.
Przydatne przy hasłach jest metoda ***.equals()***, która porównuje do siebie Stringi. **W typach prostych liczbowych operatorem porównania był znak '==', niestety w Stringu on nie zadziała**. Musimy użyć zapisu jak w przykładzie poniżej:

```
String password = "tajne";
String newPassword = "superTajneHaslo";
if (password.equals(newPassword)){
   // hasła są takie same
}
   // hasła są różne
```

Warto również wspomnieć o ciekawej metodzie ***.toLowerCase()***, która zamienia wszystkie duże litery na małe. O zastosowaniu nie muszę chyba wspominać, wystarczy spojrzeć na przykład poniżej:

```
String toxicEmail = "KRZYCZE NA CIEBIE DUZYMI LITERAM";
System.out.println(toxicEmail.toLowerCase()); //krzycze na ciebie duzymi literam
```

Na koniec zachęcam do poćwiczenia z wszystkimi metodami zawartymi w Stringu. Większość IDE po wpisaniu '.' kropki na końcu Stringa wyświetli wszystkie dostępne metody. Warto przetestować je, bo większość z nich dość szybko i często się przydaje.

String Builder czy String Buffer?
W programowaniu jak już zauważyliście stara się zaoszczędzić jak najwięcej pamięci systemowej poprzez używanie wielu typów danych. W pierwszych małych programach użycie *int* czy *byte* wiele nie zmieni, ale w dużych programach korzystających z naprawdę wielu danych, każdy bajt obliczany miliony razy potrafi wygenerować duże zapotrzebowanie na moc obliczeniową.
W przypadku 'budowania' Stringa, który wiemy, że będzie dobudowywany wiele razy używamy ***StringBuildera***. *StringBuilder* ma wiele zalet, jedną z nich jest oszczędność mocy obliczeniowej a druga to dynamika. Wyobraźmy sobie, że piszemy książkę i nie wiemy ile wyrazów będziemy w niej pisać, zaczynamy:

```
String book = "W";
book = book + " ciemnym";
book = book + " pomieszczeniu";
book = book + " stała";
book = book + " paproć";
book = book + " a koło niej";
book = book + " szkielet człowieka,";
book = book + " to zapewne";
book = book + " programista ubiegłego wieku.";
System.out.println(book); //W ciemnym pomieszczeniu stała paproć a koło niej szkielet człowieka, to zapewne programista ubiegłego wieku.
```

Teraz użyjemy StringBuildera by uzyskać ten sam efekt:

```
StringBuilder sb = new StringBuilder();
sb.append("W");
sb.append(" ciemnym");
sb.append(" pomieszczeniu");
sb.append(" stała");
sb.append(" paproć");
sb.append(" a koło niej");
sb.append(" szkielet człowieka,");
sb.append(" to zapewne");
sb.append(" programista ubiegłego wieku.");
String finalBook = sb.toString();
System.out.println(finalBook); // W ciemnym pomieszczeniu stała paproć a koło niej szkielet człowieka, to zapewne programista ubiegłego wieku.
```

Metoda ***.append()*** działa podobnie jak metoda *concat(=]).* A metoda ***.toString()*** powoduje przekształcenie naszego dynamicznego StringBuildera do postaci Stringa. Aby pokazać zróbmy prawdziwy przykład:
Test 1:

```
String test = "test";
long start = System.nanoTime();
for(int i=0; i<10000; i++) {
   test += test;
}
System.out.println("Konkatenacja zajęła:  " + (System.nanoTime()-start)); // Konkatenacja zajęła: 92880433
```

Test 2:

```
String test = "test";
StringBuilder sb = new StringBuilder();
long start2 = System.nanoTime();
for(int i=0; i<10000; i++) {
   sb.append(test);
}
System.out.println("String Builder zajął:  " + (System.nanoTime()-start2)); // String Builder zajął: 4342630
```

Widzimy, że w prostym dodawaniu kolejnego wyrazu „test” po 10 000 operacjach *StringBuilder* w przybliżeniu okazał się być dwukrotnie szybszy.
W Javie również można dynamicznie modyfikować ciąg znaków używając ***StringBuffera***. Jest on prawie identyczny jak *StringBuilder*. Różni się tym, że jest **klasą synchronizowaną** a taki*StringBuffer* nie. Oznacza to, że pisząc programy wielowątkowe będziemy musieli stosować*StringBuffera* by nie dopuścić do sytuacji, gdzie dwa wątki dobierają się do niedokończonej operacji, ale o tym kiedy indziej. Dla początkującego programisty zaleca się stosować *StringBuildera* z racji, że jest on najlepszym rozwiązaniem dla programów jednowątkowych, jest po prostu szybszy.

Immutable

Przy omawianiu Stringa nie można nie wspomnieć o magicznym słówku „**immutable**”. Swoją drogą zastanawia mnie dlaczego nie wspomina się o tym na przykład przy złożonej postaci *Integer*? W każdym razie, **w Javie String został stworzony jako niemodyfikowalny (Immutable).** Oznacza, że nie możemy dynamicznie dodawać do tego samego Stringa czegoś nowego. Musimy stworzyć nowy obiekt *String* albo przypisać mu po prostu nowe referencje. Spójrzmy poniżej:

```
String s = "Hello";
s.substring(1);
System.out.println(s); // Hello
s = s.substring(1);
System.out.println(s); // ello
```

Jak widzimy w powyższym przykładzie jeżeli będziemy starać się dynamicznie obciąć Stringa ‘*s.substring(1);*’ to nie uda nam się tego zrobić. Aby to zrobić trzeba przypisać mu nowe referencje ‘*s = s.substring(1);*’

Przykładem obiektu **Mutable** jest omówiony wcześniej StringBuilder:

```
StringBuilder sb = new StringBuilder();
sb.append("Hello"); // dodajemy "Hello"
sb.delete(0,1); // usuwamy dynamicznie pierwsza literke
System.out.println(sb.toString()); // ello
```

Na powyższym przykładzie nie musieliśmy przypisać nowych referencji do obiektu ‘sb’, wystarczyło dynamicznie dodać nowy wyraz: ‘*sb.append("Hello");*’.

Warto wspomnieć, że obiekty **immutable** takie jak *String* są ‘**Thread-Safety**’. Ma to swoje zastosowanie we współczesnych wielordzeniowych procesorach oraz wielowątkowych aplikacjach.

Zadanie 1

Odczytaj imię i nazwisko danej osoby, na ekran wypisz imię i nazwisko razem. Dla wprowadzonych danych: Adam Kowalski Prawidłowa odpowiedź to: Twoje imię i nazwisko to Adam Kowalski

Rozwiązanie:

```
import java.util.Scanner;

public class Zad1 {

  public static void main(String[] args) {
     Scanner scanner = new Scanner(System.in);
     System.out.println("Podaj imię");
     String firstName = scanner.next();
     System.out.println("Podaj nazwisko");
     String lastName = scanner.next();
     System.out.println("Twoje imię i nazwisko to "+firstName+" "+lastName);

  }
}
```

Zadanie 2

Odczytaj wyraz i wypisz długość wprowadonego wyrazu.

Rozwiązanie:

```
import java.util.Scanner;

public class Zad2 {
  public static void main(String[] args) {
```

```
    Scanner scanner = new Scanner(System.in);
    System.out.println("Podaj wyraz");
    String word = scanner.next();
    int length = word.length();
    System.out.println("Długość wyrazu to "+length);
  }
}
```

Zadanie 3

Wypisz na ekran następujący komunikat (uwzględniając cytat): “Mowa jest srebrem, a milczenie złotem”

Rozwiązanie:

```
public class Zad3 {

  public static void main(String[] args) {
    System.out.println("\"Mowa jest srebrem, a milczenie złotem\"");
  }
}
```

Zadanie 4

Napisz program, który wyświetla wprowadzony wyraz.

Rozwiązanie:

```
import java.util.Scanner;

public class Zad4 {

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Podaj wyraz");
    String word = scanner.next();
    System.out.println("Wprowadzony wyraz to "+word);
  }
}
```

Zadanie 5

Napisz program, który odczytuje wyraz. Twoim zadaniem jest wypisanie przedostatniej litery wyrazu.

Rozwiązanie:

```java
import java.util.Scanner;

public class Zad5 {

  public static void main(String[] args) {
     // 012345
     // Michał
     // 6 = length
     System.out.println("Podaj wyraz");
     Scanner scanner = new Scanner(System.in);
     String word = scanner.next();
     int length = word.length();
     System.out.println(word.charAt(length-2));
  }
}
```

Zadanie 6
Odczytaj wyraz i wypisz pierwszą literę wyrazu.

Rozwiązanie:

```java
import java.util.Scanner;

public class Zad6 {
  public static void main(String[] args) {
     System.out.println("Podaj wyraz");
     Scanner scanner = new Scanner(System.in);
     String word = scanner.next();
     int length = word.length();
     System.out.println(word.charAt(0));
  }
}
```

Zadanie 7
Odczytaj wyraz i na ekranie wypisz Hello Nazwa Wyrazu (odzielone spacją).

Rozwiązanie:

```java
import java.util.Scanner;

public class Zad7 {
  public static void main(String[] args) {
     Scanner scanner = new Scanner(System.in);
     System.out.println("Podaj wyraz");
     String word = scanner.next();
     System.out.println("Hello "+word);
```

```
  }
}
```

Zadanie 8
Odczytaj wyraz i wypisz 2 ostatnie litery wyrazu.

Rozwiązanie:

```
import java.util.Scanner;

public class Zad8 {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Podaj wyraz");
    String word = scanner.next();
    int length = word.length();
    System.out.println(word.charAt(length-2)+""+word.charAt(length-1));
  }
}
```

Zadanie 9
Odczytaj wyraz i wypisz na ekran wartość true lub false w zależności od tego czy wyraz zawiera w sobie napis kot.

Rozwiązanie:

```
import java.util.Scanner;

public class Zad9 {

  public static void main(String[] args) {
    System.out.println("Podaj wyraz");
    Scanner scanner = new Scanner(System.in);
    String word = scanner.nextLine();
    System.out.println(word.contains("kot"));
  }
}
```

Zadanie 10
Odczytaj wyraz i wypisz na ekranie 3 krotnie wartość wyrazu.

Rozwiązanie:

```
import java.util.Scanner;

public class Zad10 {
```

```
  public static void main(String[] args) {
     System.out.println("Podaj wyraz");
     Scanner scanner = new Scanner(System.in);
     String word = scanner.next();
     System.out.println(word+word+word);
  }
}
```

Zadanie 11
Odczytaj wyraz i zamień go na małe litery.

Rozwiązanie:

```
import java.util.Scanner;

public class Zad11 {

  public static void main(String[] args) {
     System.out.println("Podaj wyraz");
     Scanner scanner = new Scanner(System.in);
     String word = scanner.next();
     System.out.println(word.toLowerCase());
  }
}
```

Zadanie 12

Rozwiązanie:

Odczytaj wyraz i zamień * pustym ciągiem.
Przykładowo dla ciągu znaków a*bcd ma być wypisany wyraz abcd.

```
import java.util.Scanner;

public class Zad12 {

  public static void main(String[] args) {
     System.out.println("Podaj wyraz");
     Scanner scanner = new Scanner(System.in);
     String word = scanner.next();
     System.out.println(word.replace("*",""));
  }
}
```

Zadanie 13

Wypisz true jeśli wyraz kończy się wyrazem “tka” bądź false jeśli wyraz nie kończy się na wyrazie “tka”.

Rozwiązanie:

```
import java.util.Scanner;

public class Zad13 {

  public static void main(String[] args) {
    System.out.println("Podaj wyraz");
    Scanner scanner = new Scanner(System.in);
    String word = scanner.next();
    System.out.println(word.endsWith("tka"));
  }
}
```