Java
Standard Edition

Cześć!

Anna Skulimowska

od zawsze lubię dzielić się wiedzą

annamskulimowska@gmail.com

a wy?

Dlaczego zaczynamy od Javy SE?

- To kurs Javy :)
- Java jest bardzo popularna w aplikacjach biznesowych
- SE to podstawowa wersja platformy Java (**S**tandard **E**dition)
- Pozwala tworzyć i uruchamiać aplikacje w języku Java
- Podstawa dla Javy EE (**E**nterprise **E**dition)

Java...

- **Java SE** - wersja standardowa (najnowsza wersja to 11)
- **Java EE** - wersja enterprise - aplikacje webowe-biznesowe
- **Java ME** - wersja micro - na urządzenia mobilne
- **Kotlin** - coraz popularniejszy język programowania oparty na JVM
- **Android** - aplikacje uruchamiane na Androidzie są napisane w Javie i coraz częściej w Kotlinie

Agenda

1. Wprowadzenie do Javy
2. Pierwsza aplikacja
3. Zmienne
4. Metody
5. Klasy i obiekty
6. Java Koans

Wprowadzenie

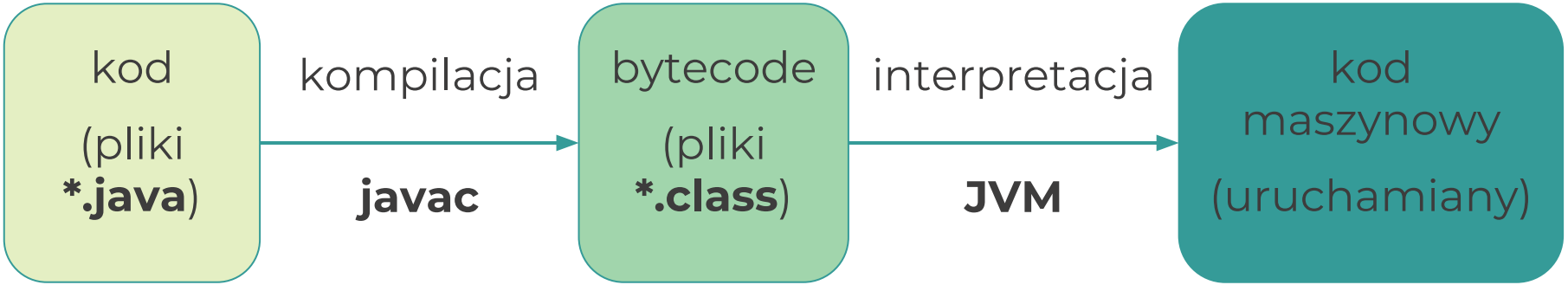
Wprowadzenie

Java Virtual Machine maszyna wirtualna oraz środowisko zdolne do wykonywania **kodu bajtowego** Javy (“procesor”).

Java Runtime Environment środowisko uruchomieniowe Javy (pozwala na uruchamianie programów). Zawiera JVM.

Java Development Kit zawiera narzędzia tworzenia aplikacji oraz JRE i JVM.

Wprowadzenie - kompilacja



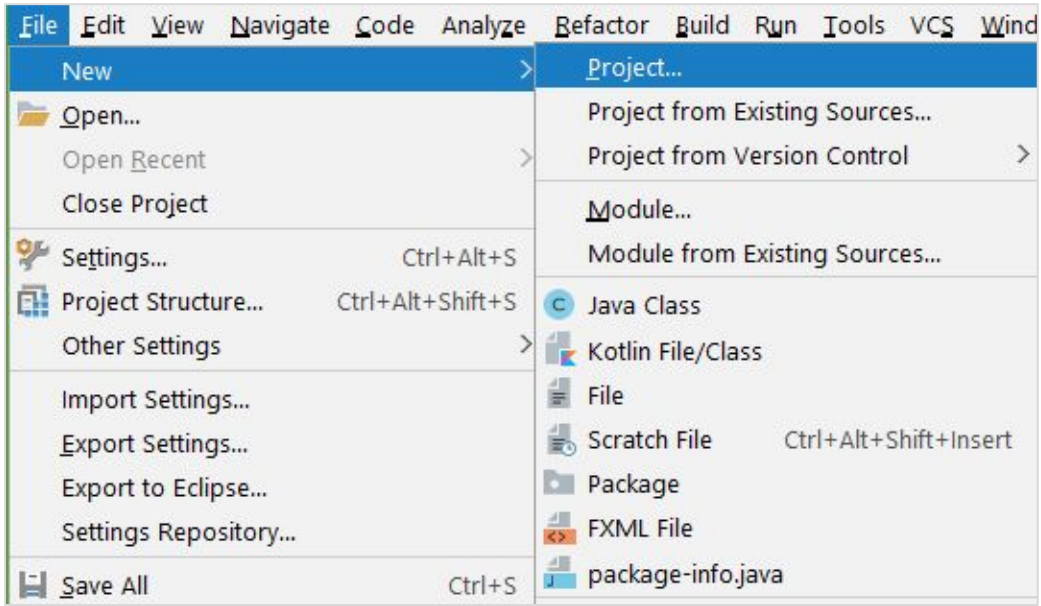
Wprowadzenie - główne cechy Javy

- **Wieloplatformowość** - kod jest niezależny od systemu operacyjnego
- **Obiektowość** - łączy stan (dane) i zachowanie (wykonywanie logiki)
- **Garbage Collection** - mechanizm automatycznego zwalniania nieużywanej pamięci

Pierwsza aplikacja

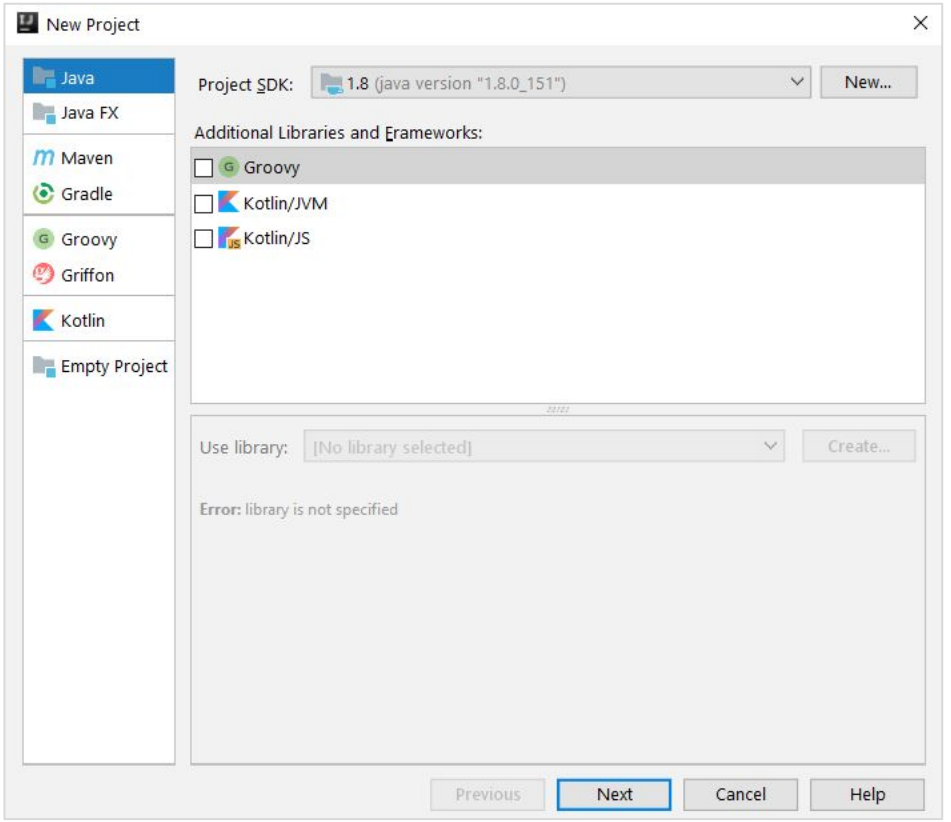
Pierwsza aplikacja - tworzenie projektu

1. File -> New -> Project...



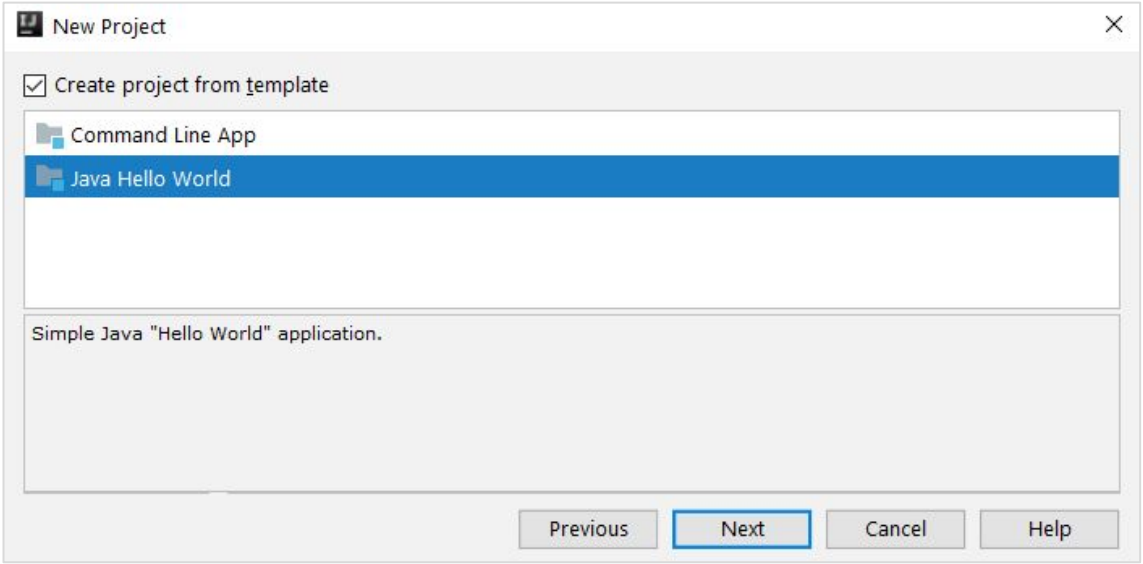
Pierwsza aplikacja - tworzenie projektu

2. Java -> Next



Pierwsza aplikacja - tworzenie projektu

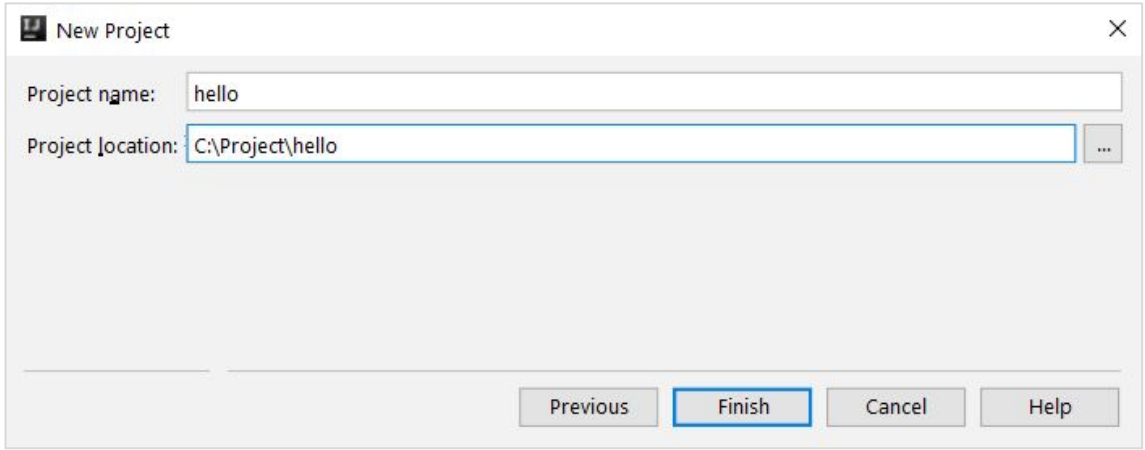
3. Create project from template -> Java Hello Word! -> Next



Pierwsza aplikacja - tworzenie projektu

4. Uzupełnij *Project name*

5. Finish

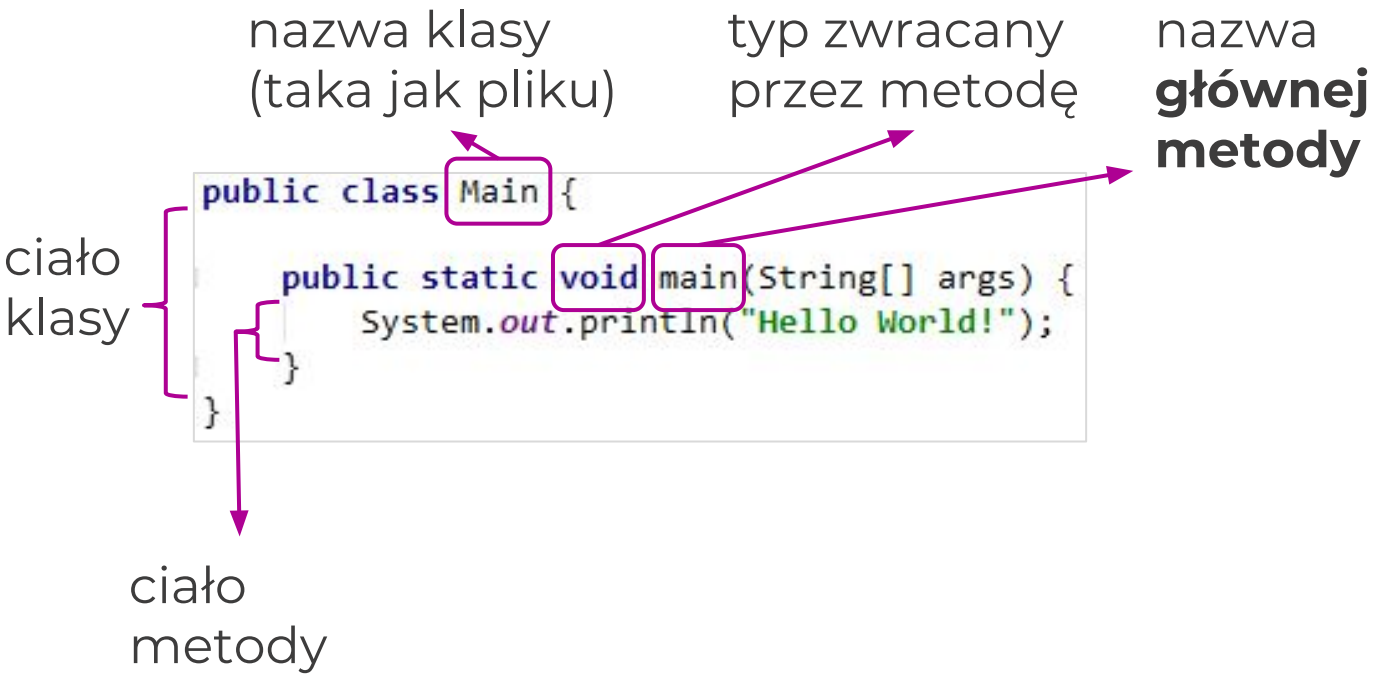


Pierwsza aplikacja - Hello world!

```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Pierwsza aplikacja - Hello world!

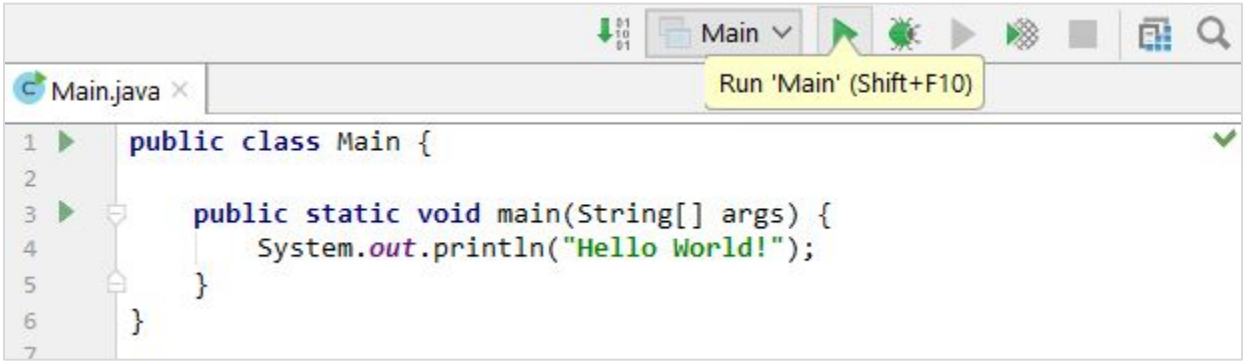


Zadanie 1

Stwórz pierwszą aplikację konsolową w Javie wypisującą na konsolę napis “Hello world!”.

Zadanie 2

Uruchom stworzoną aplikację wykorzystując przycisk **Run**.



Komentarze

```java
public class Main {

    public static void main(String[] args) {
        // comment
        System.out.println("Hello World!");

        /*
          another comment
         */
    }
}
```

Zadanie 3

1. Pobierz szkielet projektu z repozytorium Git:

```
git clone
https://github.com/infoshareacademy/jjdzl1-materialy.git
```

2. Stwórz swoją gałąź:

```
git checkout -b my-branch
git push -u origin my-branch
```

3. Otwórz pobrany projekt w IntelliJ IDEA i uruchom go

4. Po każdym zadaniu przesyłaj wyniki do repozytorium

```
git add .
git commit -m wiadomość
git push
```

Klasy

Klasy

- Podstawowy element (mechanizm) języka Java
- Typ danych
- Zawiera pola (dane) i metody (logikę)

np. klasa o nazwie **Rachunek** może zawierać pole **numerRachunku** oraz metodę **sprawdźStanRachunku**

Zmienne - przechowywanie danych

```
int number;

number = 12;

int anotherNumber = 13;
```

Zmienne - przechowywanie danych

deklaracja zmiennej

inicjalizacja zmiennej

deklaracja z inicjalizacją

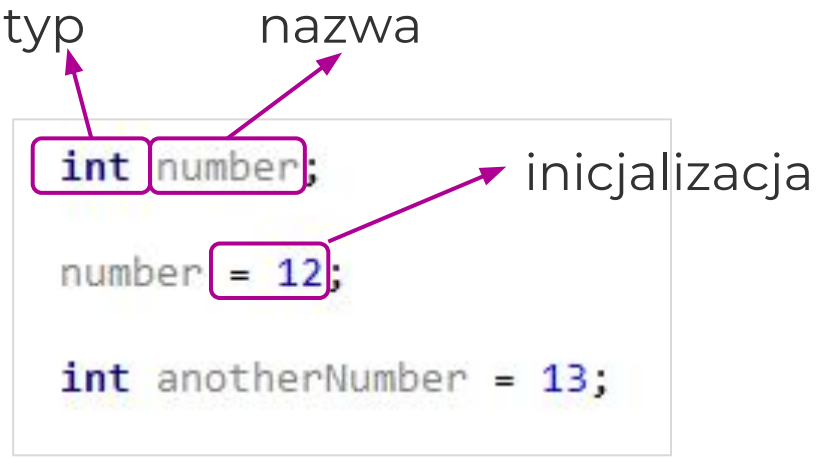
```
int number;

number = 12;

int anotherNumber = 13;
```

zmiennych używamy wewnątrz metod

Zmienne - przechowywanie danych



zmiennych używamy wewnątrz metod

Zmienne - typy

Oparte o typy proste
typ pisany z małej litery

```
int number = 0;
double numberWithComma = 1.2;
```

Oparte o klasy
typ pisany z dużej litery

```
String appleName = "Jabłka Ligol";
```

Zmienne - typy proste (primitive)

- **boolean** true/false
- **byte** liczba od -128 do 127
- **short** liczba do 32,768 do 32,767
- **int** liczba od -2^{31} do $2^{31}-1$
- **long** liczba od -2^{63} do $2^{63}-1$
- **float** liczba zmiennoprzecinkowa
- **double** liczba zmiennoprzecinkowa o większej precyzji
- **char** pojedynczy znak Unicode

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html warto przeczytać

Metody - wykonują logikę

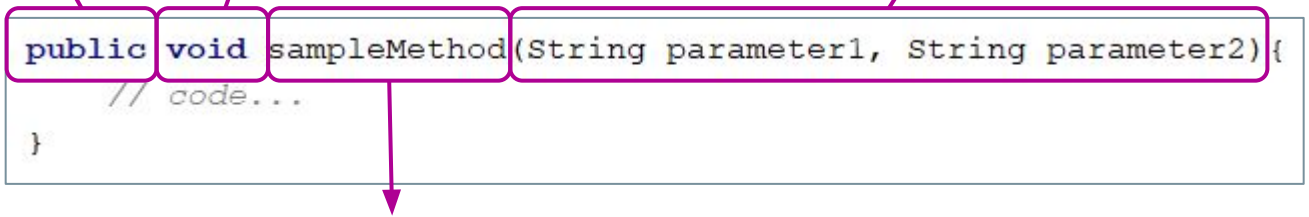
```
public void sampleMethod(String parameter1, String parameter2){
    // code...
}
```

Metody - wykonują logikę

modyfikator dostępu, czyli gdzie “widać” metodę

co zwraca metoda (typ)

lista parametrów



nazwa metody

Metody - zwracanie danych

```
public static String returnString(){
    return "sample text";
}
```

Zadanie 4 

1. Stwórz metodę `sum` zwracającą wynik dodawania dwóch liczb całkowitych podanych jako argumenty metody. Metoda powinna wypisywać na konsolę informacje o wykonywanym działaniu. Przetestuj działanie.

Klasy zawierają

Pola

// przechowują dane

// “zmienna na poziomie klasy”

Metody

// wykonują logikę

// powinny odpowiadać za jedno konkretne zachowanie

// metody statyczne są związane z klasą, metody niestatyczne są związane z obiektem

```
public class Card {

    private String name;

    public String getName() {
        return name;
    }
}
```

Klasy zawierają

klasa	obiekt

// typ danych “szablon”, “foremka”, definicja obiektów
// opisuje w jaki sposób będzie zbudowany obiekt, ale nie zawiera konkretnych danych
// mechanizm języka, pozwalający tworzyć obiekty zachowujące się w ten sam sposób
	// instancja klasy
	// “wypełnienie szablonu, foremki
	// zbudowany w oparciu o klasę, zawiera konkretne dane
	// reprezentacja klasy

Konstruktor

// definiuje w jaki sposób tworzony jest nowy obiekt

// wywoływany przy użyciu operatora `new`

// przy braku deklaracji konstruktora Java automatycznie tworzy domyślny konstruktor (bezparametrowy)

```
public Card(String name) {
    this.name = name;
}
```

this - słowo kluczowe

// odniesienie do “bieżącego obiektu” stosowane wewnątrz klasy którą reprezentuje obiekt

// umożliwia rozwiązanie konfliktu nazw

```
public Card(String name) {
    this.name = name;
}
```

static - słowo kluczowe

Znaczenie zależy od miejsca w którym występuje:

// **przed typem pola** - pole ma taką samą wartość niezależnie od obiektu (nazywane czasem zmienną globalną - bad practice)

// **przed typem metody** - powiązana z klasą, nie z obiektem. Nie można w niej używać niestatycznych pól i metod.
Np. `Assert.assertTrue();`

Argument vs parametr

Argument vs parametr

```
public class CreditCard {

    private String name;

    public CreditCard(String name) {
        this.name = name;
    }
}
```

parametr

```
String cardName = "Visa";
CreditCard visa = new CreditCard(cardName);
```

argument

Zadanie 5

1. Stwórz klasę `Card`, która zawiera:
    a. dwa pola tekstowe `name` i `number`
    b. konstruktor inicjalizujący oba pola
    c. niestatyczną metodę `printData` wypisującą na konsolę nazwę i numer karty
2. W głównej metodzie programu stwórz dwie karty i wywołaj na nich metodę `printData`

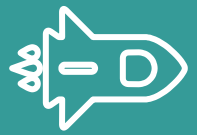
Zadanie 6

1. W utworzonej klasie `Card` dodaj statyczną metodę wypisującą na konsolę napis *“To jest karta”*.
2. Przetestuj działanie metody.
3. Spróbuj użyć w metodzie pól klasy. Co się stało? Dlaczego?

Zadanie 6

1. W utworzonej klasie `Card` dodaj statyczną metodę wypisującą na konsolę napis *“To jest karta”*.
2. Przetestuj działanie metody.
3. Spróbuj użyć w metodzie pól klasy. Co się stało? Dlaczego?

Java Koans



Seria ćwiczeń służąca do nauki Javy:

`https://github.com/infoshareacademy/java-koans`

Pobierz projekt i na swojej gałęzi uzupełnij wskazane w terminalu koany.

Java Koans

Wskazówki:

- metoda `assertEquals` porównuje wartości argumentów
- obiekty porównujemy przy pomocy metody `equals`, a typy proste przy pomocy ==
- operatory w Javie https://docs.oracle.com/javase/tutorial/java/nutsandbolts/opsummary.html

Bonus

Standardowe konwencje nazewnictwa

Nadal aktualne:

https://www.oracle.com/technetwork/java/codeconventions-135099.html

Dobre książki

1. Java: podstawy. Cay S. Horstmann (dobre na początek)
2. Thinking in Java. Bruce Eckel (dla dociekliwych)

Zadanie 7

1. Utwórz klasę `StringTest` ze statyczną metodą `testStringMethods`, która nic nie zwraca. Wywołaj ją w głównej metodzie programu.
2. Wewnątrz metody zadeklaruj dwie zmienne:
   a. `String emptyText = "";`
   b. `String text = "Lubię programować";`

Zadanie 7

3. Przetestuj działanie różnych metod klasy `String`. Przed wywołaniem powinien zostać wyświetlony na konsoli stosowny komunikat. Przykładowo:

```
public static void testStringMethods(){
    String emptyText = "";
    String text = "Lubię programować";

    System.out.println("Wynik działania metody isEmpty dla pustego stringa " + emptyText.isEmpty());
    System.out.println("Wynik działania metody isEmpty dla niepustego stringa " + text.isEmpty());

}
```

dokumentacja:

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

Zadanie 7

Argumenty metod dobierz według własnego uznania. Tak, żeby zrozumieć działanie danej metody.

Metody do przetestowania:

- `isEmpty` (dla obu zmiennych)
- `charAt` (tylko dla zmiennej text)
- `contains` (dla obu zmiennych)
- `endsWith` (dla obu zmiennych)
- `startsWith` (dla obu zmiennych)
- `indexOf` (dla obu zmiennych)

Zadanie 7

- `lastIndexOf` (dla obu zmiennych)
- `replace` (dla obu zmiennych)
- `length` (dla obu zmiennych)
- `toLowerCase` (dla obu zmiennych)
- `toUpperCase` (dla obu zmiennych)

Zadanie 8

1. Stwórz klasę `Account`, która zawiera:
    a. dwa pola tekstowe `name` i `number`
    b. jedno pole `balance` typu `BigDecimal` (precyzyjna klasa do przechowywania liczb https://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html)
    c. konstruktor inicjalizujący wszystkie pola
    d. metodę `getBalance` zwracającą wartość salda rachunku

Zadanie 8

2. W klasie `Account` dodaj metodę `pay` z parametrem `BigDecimal paymentAmount`, która pomniejsza saldo rachunku o podaną w parametrze kwotę oraz wypisuje na konsolę stosowny komunikat (metoda nie powinna nic zwracać). Zmniejszenie salda powinno wykorzystywać jedną z metod klasy `BigDecimal`.
3. W głównej metodzie programu stwórz rachunek o dowolnym numerze, nazwie i saldzie (do stworzenia salda użyj konstruktora klasy `BigDecimal` np. `new BigDecimal(10)`)
4. Używając metody `getBalance` wypisz na konsolę wysokość salda rachunku

Zadanie 8

5. Używając metody `pay` zmniejsz saldo rachunku o dowolną kwotę
6. Używając metody `getBalance` ponownie wypisz na konsolę wysokość salda rachunku

Dzięki

Pytania?

Zawsze możesz do mnie napisać:
annamskulimowska@gmail.com