

Zajęcia dla grup początkujących, lecz uczestnicy znają już typy zmiennych, potrafią tworzyć klasy oraz metody w Javie. Zajęcia mogłyby być wprowadzeniem do instrukcji warunkowych, wtedy należałoby się na nich bardziej skupić. Jeśli grupa byłaby już bardziej zaawansowana można rozwinąć wątek wyjątków. Nasz kalkulator można rozwijać i dodawać kolejne funkcjonalności i obsługę kolejnych operacji matematycznych.

Kalkulator

Wykonywanie prostych operacji arytmetycznych

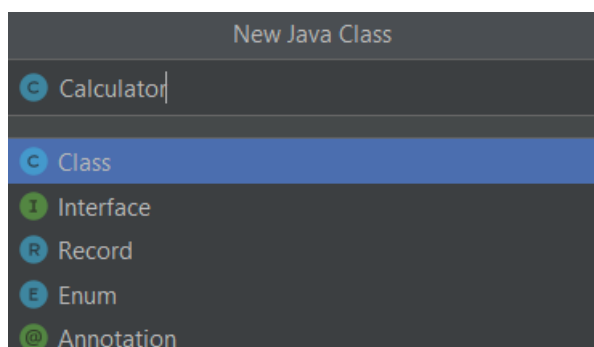
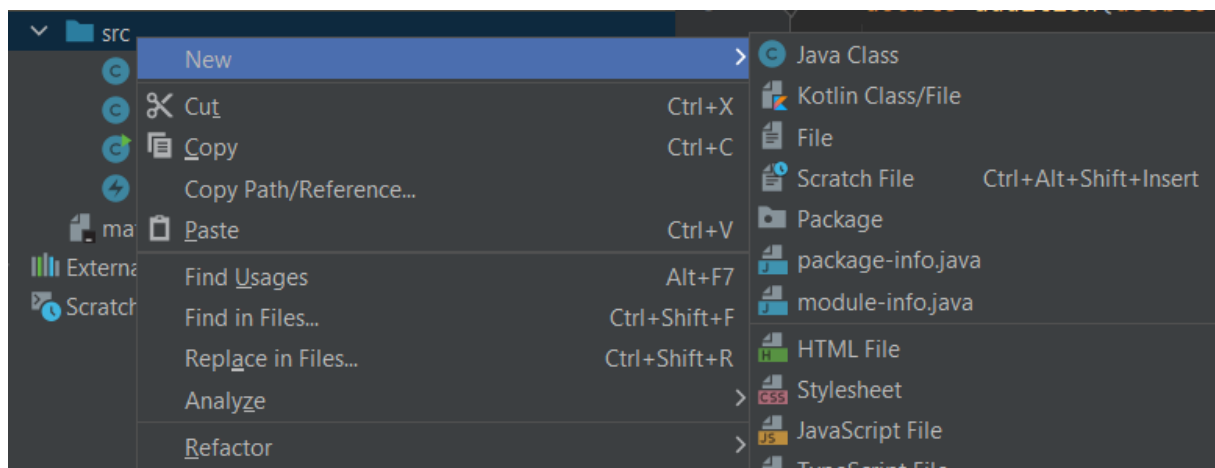
Na początku:

Omawiamy na czym będzie polegała nasza aplikacja: jakie działania chcemy być w stanie wykonywać, czy pobieramy dane od użytkownika, jakie wyjątki chcemy uwzględnić itd.

Stworzymy dwa różne kalkulatory, aby pokazać jak najwięcej funkcjonalności. Skorzystamy z instrukcji warunkowych: if i switch, poznamy klasę Scanner oraz stworzymy własną klasę obsługującą błąd.

Kalkulator #1:

Tworzymy nowy projekt oraz nową klasę „Calculator”



Można wspomnieć uczestnikom o tym, że warto od początku używać angielskiego nazewnictwa, aczkolwiek jeśli ktoś chce nazwać po polsku to nie jest to błąd.

Będziemy teraz dodawać metody, słuchamy propozycji na nazwę, parametry jakie będzie nasza metoda przyjmować oraz co będzie zwracać.

```
double addition(double a, double b){
    return a + b;
}
```

Zachęcamy aby na podobnej zasadzie utworzyć resztę potrzebnych metod.

```
double subtraction(double a, double b){
    return a - b;
}

double multiplication(double a, double b){
    return a * b;
}

double division(double a, double b){
    return a/b;
}
```

Przechodzimy do głównej klasy, jeśli korzystamy z szablonu mamy już gotową metodę main, jeśli nie musimy ją stworzyć. Jeśli będziemy pobierać dane od użytkownika, potrzebujemy Scannera.

```
import java.util.Scanner;

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

        System.out.print("Podaj pierwszą liczbę: ");
        double num1 = scanner.nextDouble();

        System.out.print("Podaj drugą liczbę: ");
        double num2 = scanner.nextDouble();
    }
}
```

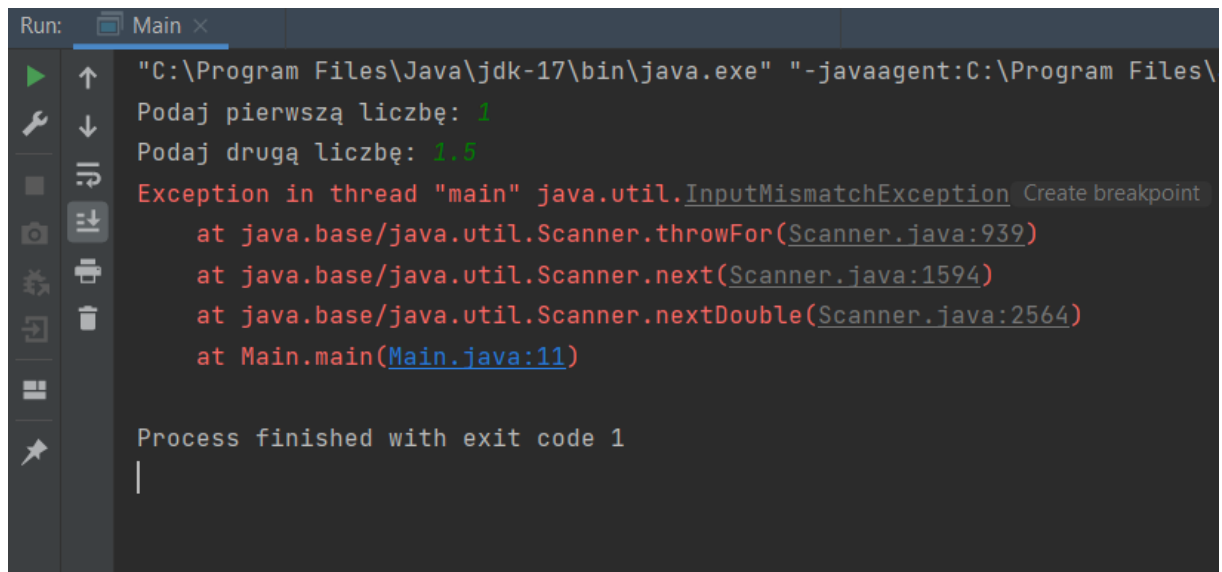
Omawiamy konstrukcje, będzie trzeba go również zaimportować. Można to zrobić najeżdżając kursorem na „Scanner” i wybrać opcję „Import class”.

```
Scanner scanner = new Scanner(System.in);
```

Cannot resolve symbol 'Scanner'

Import class Alt+Shift+Enter More actions... Alt+Enter

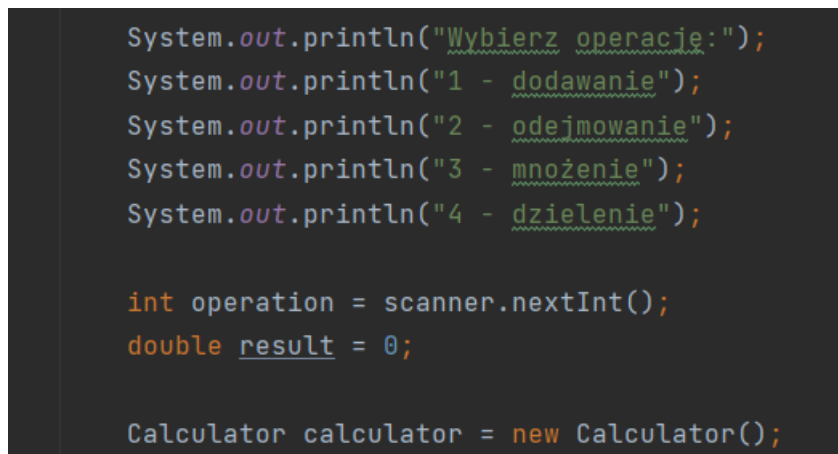
Gdy to już mamy, można pokazać jak działa Scanner w konsoli. Uczestnicy mogą przetestować na dowolnych liczbach. Warto zwrócić uwagę na liczby zmiennoprzecinkowe, w zależności od ustawień będziemy używać przecinka lub kropki. „1.5” będzie powodowało błędy, gdy „1,5” już nie.



```
Run: Main x
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\
Podaj pierwszą liczbę: 1
Podaj drugą liczbę: 1.5
Exception in thread "main" java.util.InputMismatchException Create breakpoint
    at java.base/java.util.Scanner.throwFor(Scanner.java:939)
    at java.base/java.util.Scanner.next(Scanner.java:1594)
    at java.base/java.util.Scanner.nextDouble(Scanner.java:2564)
    at Main.main(Main.java:11)

Process finished with exit code 1
```

Zastanawiamy się wspólnie o co jeszcze musimy zapytać użytkownika oraz jak to zrealizujemy. Tworzymy obiekt klasy Calculator. Zwracamy uwagę na typy i metody (int - „nextInt()”)

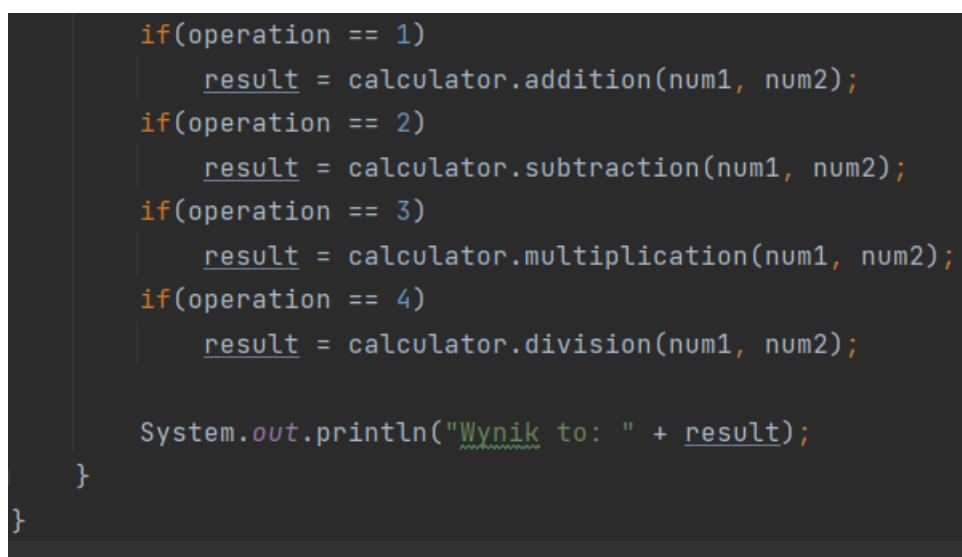


```
System.out.println("Wybierz operację:");
System.out.println("1 - dodawanie");
System.out.println("2 - odejmowanie");
System.out.println("3 - mnożenie");
System.out.println("4 - dzielenie");

int operation = scanner.nextInt();
double result = 0;

Calculator calculator = new Calculator();
```

Będziemy potrzebowali instrukcji warunkowych do użycia konkretnego operatora oraz używamy odpowiednich metod z klasy Calculator.



```
if(operation == 1)
    result = calculator.addition(num1, num2);
if(operation == 2)
    result = calculator.subtraction(num1, num2);
if(operation == 3)
    result = calculator.multiplication(num1, num2);
if(operation == 4)
    result = calculator.division(num1, num2);

System.out.println("Wynik to: " + result);
}
}
```

Testujemy aplikację:

```
0: (Program Files (x86)\jdk-17\bin\java
Podaj pierwszą liczbę: 10
Podaj drugą liczbę: 3
Wybierz operację:
1 - dodawanie
2 - odejmowanie
3 - mnożenie
4 - dzielenie
4
Wynik to: 3.3333333333333335

Process finished with exit code 0
```

Kalkulator #2

Tworzymy nową klasę oraz metodę, tym razem będzie to jedna funkcja do obsługi wszystkich operatorów (+,-,*,/). Musimy go uwzględnić w sygnaturze metody.

```
public class Calc {
    public double result(double num1, String operator, double num2) {
        double res;
        switch (operator) {
            case "+" -> res = num1 + num2;
            case "-" -> res = num1 - num2;
            case "*" -> res = num1 * num2;
            case "/" -> {
```

Omawiamy konstrukcję i działanie instrukcji warunkowej switch. Przy operacji dzielenia chcemy uwzględnić obsługę wyjątków: dzielenia przez zero oraz podania nieprawidłowego operatora.

Robimy wprowadzenie do obsługi wyjątków, ich hierarchii.

Chcemy stworzyć własny wyjątek – w tym celu tworzymy kolejną klasę. Bardzo prostą, dziedziczącą po Exception.

```
public class UnknownOperatorException extends Exception{
    public UnknownOperatorException(String message) {
        super(message);
    }
}
```

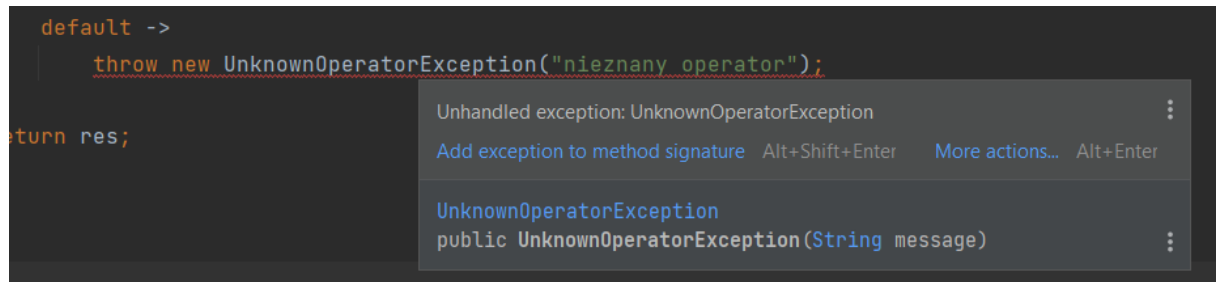
Dzielenie:

```

case "/" -> {
    if (num2 == 0)
        throw new ArithmeticException("Nie można dzielić przez 0");
    else res = num1 / num2;
}
default ->
    throw new UnknownOperatorException("nieznany operator");

```

Przy wyrzucaniu wyjątku `UnknownOperatorException` będziemy musieli dodać `throws` do sygnatury metody. Nie jest to konieczne w wypadku `ArithmeticException` ponieważ dziedziczy on po `RuntimeException`.



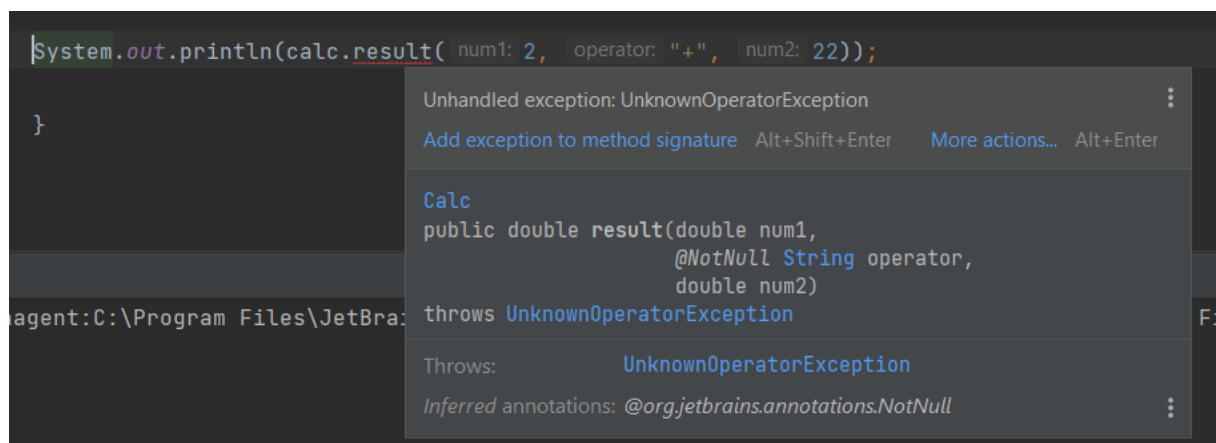
```

public double result(double num1, String operator, double num2) throws UnknownOperatorException {

```

Wracamy do klasy `Main`: tworzymy obiekt klasy `Calc` oraz chcemy użyć naszego kalkulatora (można pobierać dane z konsoli, ale tym razem już dla uproszczenia podamy je od razu w kodzie).

Znowu dowiadujemy się, że mamy nieobsłużony błąd:



Tym razem nie dodamy wyjątku do sygnatury metody, tylko użyjemy `try/catch`. Testujemy kalkulator.

```

Calc calc = new Calc();
try {
    System.out.println(calc.result( num1: 2, operator: "+", num2: 22));
} catch (UnknownOperatorException e) {
    e.printStackTrace();
}

```