PROGRAMOWANIE OBIEKTOWE JAVA – LABORATORIUM

KLASY

DEFINICJA KLASY

```
public class ClassName {

    //definicja pól
    //definicja konstruktorów
    //definicja metod
}


public class Osoba {

    //definicja pól
    String imie = "Jan", nazwisko = "Nowak";
    int wiek = 11;
    //definicja konstruktorów

    //definicja metod
    public void Info(){
        System.out.println("Imie Nazwisko: "+imie +" "+ nazwisko+"\nWiek:
"+wiek);
    }
}
```

TWORZENIE OBIEKTÓW KLASY

```
public class Main {
    public static void main(String[] args) {
        //stworzenie obiektu klasy Osoba
        Osoba obiekt1 = new Osoba();
        //wywołanie metody
        obiekt1.Info();
    }
}
```

METODY GETTER I SETTER

```
public class Osoba {

    //definicja pól
    String imie = "Jan", nazwisko = "Nowak";
    int wiek = 11;
    //definicja konstruktorów

    //definicja metod
    public void Info(){
        System.out.println("Imie Nazwisko: "+imie +" "+ nazwisko+"\nWiek:
"+wiek);
    }
    //metody getter i setter
    // Gettery i settery są metodami pozwalającymi na pobranie
    // lub zmianę wartości prywatnego pola klasy z zewnątrz.
    public String getImie() {
        return imie;
    }

    public void setImie(String imie) {
        this.imie = imie;
    }
```

```
    public String getNazwisko() {
        return nazwisko;
    }

    public void setNazwisko(String nazwisko) {
        this.nazwisko = nazwisko;
    }

    public int getWiek() {
        return wiek;
    }

    public void setWiek(int wiek) {
        this.wiek = wiek;
    }
}


public class Main {
    public static void main(String[] args) {
        Osoba obj = new Osoba();

        obj.setImie("Jan");
        obj.setNazwisko("Nowak");
        obj.setWiek(11);

        obj.Info();
    }
}
```

KONSTRUKTOR

Zadaniem konstruktora jest inicjalizacja obiektu podczas jego tworzenia. Konstruktor ma taką samą nazwę jak klasa i z punktu widzenia składni jest podobny do metody. Konstruktory nie mają jednak określonego typu zwracanego. Zwykle konstruktor nadaje wartości początkowe zmiennym składowym obiektu lub wykonuje inne czynności wymagane do nadania obiektowi ostatecznej postaci.

```
public class Osoba {

    //definicja pól
    String imie = "Jan", nazwisko = "Nowak";
    int wiek = 11;
    //definicja konstruktorów
    //konstruktor bezargumentowy
    public Osoba(){
        //ciało konstruktora
    }
    //konstruktor z agrymnetami
    public Osoba(String imie, String nazwisko, int wiek) {
        this.imie = imie;
        this.nazwisko = nazwisko;
        this.wiek = wiek;
    }

    //definicja metod
    public void Info(){
        System.out.println("Imie Nazwisko: "+imie +" "+ nazwisko+"\nWiek:
"+wiek);
    }
```

```
}

public class Kalkulator {
    double a, b;

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

JAVA STRING

```
public class JavaStringExample {

    public static void main(String[] args) {
        String str1 = "Java", str2= "Java", str3="C#";
        String str4 = "Programowanie obiektowe - ";

        System.out.println(str1);
        System.out.println(str3);

        //metody klasy String
        // length()
        System.out.println("Dlugosc tekstu: "+str1.length());

        //concat() - połaczenie dwóch String
        // str1.concat(str2);
        System.out.println(str4.concat(str1));

        //equals() - metoda porównywania ciągów znaków
        //str1.equals(str2);

        //porównanie str1 i str2
        boolean result1 = str1.equals(str2);
        System.out.println("Porownanie str1 i str2: "+result1);

        //porównanie str1 i str3
        boolean result2 = str1.equals(str3);
        System.out.println("Porownanie str1 i str2: "+result2);

    }
}
```

Znak ucieczki w Java String

W Javie stringi przechowujemy w cudzysłowie w związku tym kod `String example = "This is the "String" class";` wywoła błąd, ale użycie znaku ucieczki '\' pozwoli na następującą definicję: `String example = "This is the \"String\" class";`
Znak ucieczki powoduje ze kompilator powyższy tekst przeczyta jako jednego stringa.

Tworzenie string z użyciem operatora new

```
public class JavaStringExample {

    public static void main(String[] args) {
        StringNew();

    }

    public static void StringNew(){
```

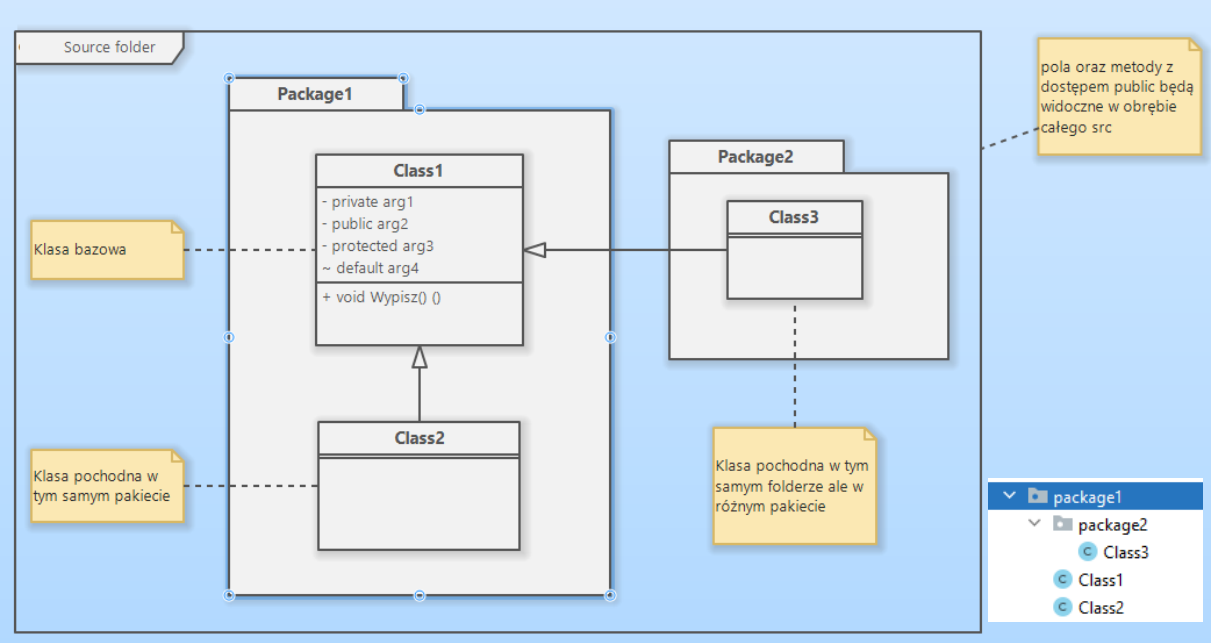
```
        String str1 = new String("Przykładowy tekst");

        System.out.println(str1);
    }
}
```

Wybrane metody klasy String

Metoda	Opis
contains()	checks whether the string contains a substring
substring()	returns the substring of the string
join()	join the given strings using the delimiter
replace()	replaces the specified old character with the specified new character
replaceAll()	replaces all substrings matching the regex pattern
replaceFirst()	replace the first matching substring
charAt()	returns the character present in the specified location
getBytes()	converts the string to an array of bytes
indexOf()	returns the position of the specified character in the string
compareTo()	compares two strings in the dictionary order
compareToIgnoreCase()	compares two strings ignoring case differences
trim()	removes any leading and trailing whitespaces
format()	returns a formatted string
split()	breaks the string into an array of strings
toLowerCase()	converts the string to lowercase
toUpperCase()	converts the string to uppercase
valueOf()	returns the string representation of the specified argument
toCharArray()	converts the string to a char array
matches()	checks whether the string matches the given regex
startsWith()	checks if the string begins with the given string
endsWith()	checks if the string ends with the given string
isEmpty()	checks whether a string is empty of not
intern()	returns the canonical representation of the string
contentEquals()	checks whether the string is equal to charSequence
hashCode()	returns a hash code for the string
subSequence()	returns a subsequence from the string

MODYFIKATORY DOSTĘPU

- default – deklaracje widoczne w pakiecie (pakiet prywatny)
- private – dostęp tylko w klasie
- protected – dostęp w pakiecie oraz w klasach pochodnych
- public – dostęp w całym projekcie



OPERATOR THIS

Słowo kluczowe this używane jest do odwoływania się do bieżącego obiektu wewnątrz metody lub konstruktora.

```
package PO_UR;

public class Osoba {

    String imie = "Jan", nazwisko = "Nowak";
    int wiek = 11;

    Osoba(String imie) {
        this.imie = imie;
        System.out.println("this reference "+this);
    }

    public static void main(String[] args) {
        Osoba obj1 = new Osoba("Jan");
        System.out.println("object reference: "+obj1);
    }

}
output
this reference PO_UR.Osoba@119d7047
object reference: PO_UR.Osoba@119d7047
Odniesienie zarówno do obiektu oraz poprzez this jest takie samo.
```

PRZEKAZANIE OBIEKTU JAKO ARGUMENT METODY:

```
public class Punkt {
    int x,y;
```

```
    public Punkt(int x, int y) {
        this.x = x;
        this.y = y;

        System.out.println("Początkowe wartości x = " + this.x + ", y = " +
this.y);

        // wywołanie metody add
        add(this);

        System.out.println("Wartość x i y po przekazaniu obiektu do metody
x= "+ this.x + ", y = " + this.y);
    }
    void add(Punkt pkt1){
        pkt1.x+=2;
        pkt1.y+=2;
    }

    public static void main(String[] args) {
        Punkt obj1 = new Punkt(1,-2);
    }
}
```

FINAL

Słowo kluczowe `final` służy do oznaczania stałych. Może być używany ze zmiennymi, metodami i klasami. W przypadku gdy zmienna, metoda lub klasa zostanie zadeklarowana jako `final`, można ją przypisać tylko raz.

```
public class ExampleFinal {

    public static void main(String[] args) {
        final int wiek = 23;
        System.out.println("wiek: "+wiek);

        wiek = 12; // error java: cannot assign a value to final variable
wiek

    }
    public final void wypisz(){
        System.out.println("Metoda final");
    }


// create a final class
final class FinalClass {
    public void display() {
        System.out.println("This is a final method.");
    }
}

// try to extend the final class
class Main extends FinalClass {
    public  void display() {
        System.out.println("The final method is overridden.");
    }

    public static void main(String[] args) {
        Main obj = new Main();
        obj.display();
```

```
    }
}
```

JAVA INSTANCEOF OPERATOR

Operator wykorzystywany do sprawdzania czy obiekt jest instancją klasy.

```
public class ExampleInstanceof {

    public static void main(String[] args) {
        String name = "Programiz";

        // checks if name is instance of String
        boolean result1 = name instanceof String;
        System.out.println("name is an instance of String: " + result1);

        // create an object of class
        ExampleInstanceof obj = new ExampleInstanceof();

        // checks if obj is an instance of Main
        boolean result2 = obj instanceof ExampleInstanceof;
        System.out.println("obj is an instance of Main: " + result2);
    }
}
```

Zadania do samodzielnego rozwiązania:

1. W nowym pakiecie utwórz klasy opisujące następujące figury geometryczne: Koło, Kwadrat, Prostokąt, Sześcian, Prostopadłościan, Kula, Stożek. Dla każdej klasy dobierz odpowiednie pola. Utwórz także metody obliczające pola figur, obwody (dla figur płaskich), oraz objętości (dla figur przestrzennych). Dla każdej klasy utwórz metodę wyświetlającą dane dotyczące figury tj. nazwa, parametry, wartość pola i obwodu lub objętości. Utwórz obiekty tych figur i pokaż wyniki obliczeń przy użyciu funkcji wyświetlającej dane. Utwórz kalkulator dla figur geometrycznych tj. odpowiednie menu pozwalające na: wybór figury geometrycznej oraz wprowadzanie parametrów dla tej figury z konsoli. Następnie wyświetl wyniki przy użyciu metody wyświetlającej dane.
2. Stwórz klasę przechowującą informacje o Budynku (nazwa, rok budowy, liczba pięter). Przygotuj metodę wyświetlającą wszystkie informacje o budynku, oraz metodę obliczającą ile lat ma budynek (rok obecny możesz ustawić na sztywno). Stwórz kilka obiektów (budynków), ustaw im wartości i wywołaj dla nich metody. Do obliczenia daty należy użyć klasy LocalData.
3. Stwórz program w którym będzie utworzonych kilka obiektów typu Gatunek. Klasa Gatunek powinna zawierać następujące pola: nazwę rodzaju, nazwę gatunkową, liczbę chromosomów 2n, podstawową liczbę chromosomów x, opis oraz metody: podającą pełną nazwę (Rodzaj + gatunek), podającą haploidalną liczbę chromosomów n, wypisującą wszystkie dane, klonującą obiekt – metoda powinna zwracać odnośnik do nowego obiektu typu Gatunek o wartościach pól takich samych jak w obiekcie, w którym została wywołana. W programie powinny być użyte wszystkie metody.