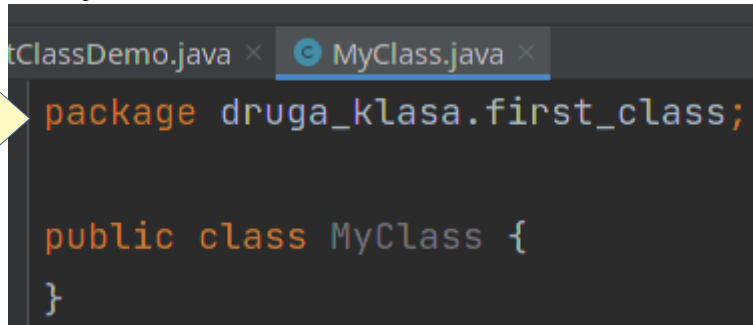


PIERWSZA KLASA

Klasę tworzymy poprzez zapisanie jej w pliku o nazwie dokładnie takiej samej jak nazwa klasy [MyClass → plik MyClass.java]

W IntelliJ tylko



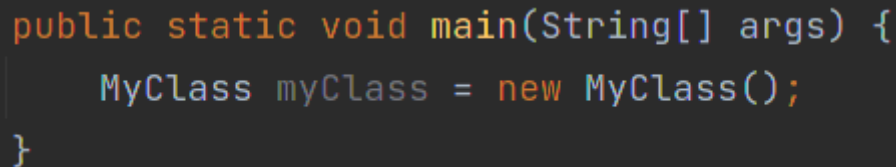
```
package druga_klasa.first_class;

public class MyClass {
}
```

Każda klasa posiada konstruktor domyślny. Konstruktory już używaliśmy np.:

new Scanner(System.in)

czyli aby teraz stworzyć obiekt na podstawie klasy należy w głównej klasie uruchamiającej wpisać:

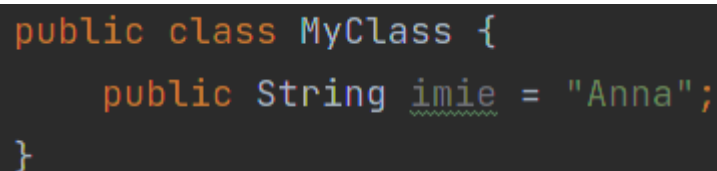


```
public static void main(String[] args) {
    MyClass myClass = new MyClass();
}
```

Nazwa obiektu dostępnego w programie to teraz **myClass** – reprezentuje on klasę **MyClass**.

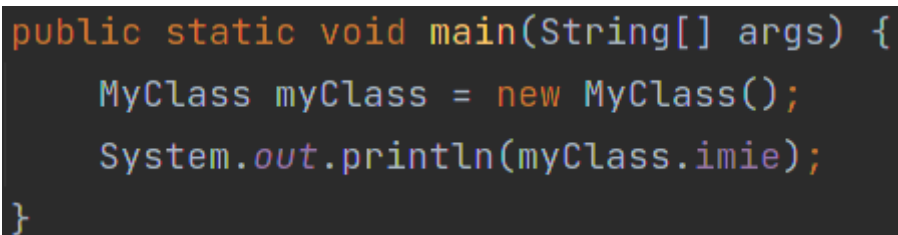
Każdy obiekt powinien posiadać jakieś parametry i metody (jak widać w powyższym przykładzie pusty obiekt też można wywołać).

Poniżej obiekt z jednym publicznym polem **imie**



```
public class MyClass {
    public String imie = "Anna";
}
```

W programie głównym można teraz odwołać się do pola imię np.: wyświetlić jego zawartość.



```
public static void main(String[] args) {
    MyClass myClass = new MyClass();
    System.out.println(myClass.imie);
}
```

MyClass.imie należy traktować jak zmienną typu String.


Każdy obiekt musi posiadać konstruktor – czyli element opisujący początkowy stan obiektu.

Chcemy aby obiekt przy utworzeniu posiadał już imię. Zmodyfikowany obiekt:

```
3 public class MyClass {
4     public String imie;
5
6     public MyClass(String imie){
7         this.imie = imie; //this.imie - odnosi się do pola w 4 linijce
8     }
9 }
```

Teraz należy w głównym programie wpisać:

```
3 ▶ public class FirstClassDemo {
4 ▶     public static void main(String[] args) {
5         MyClass myClass = new MyClass(imie: "ROMAN");
6         System.out.println(myClass.imie);
7     }
8 }
```



Każda klasa może posiadać dowolną ilość pól (preferencji) oraz dowolną ilość konstruktorów (o tym jednak później)

HERMETYZACJA oraz metody publiczne

Czym jest hermetyzacja?

Hermetyzacja, inaczej również **enkapsulacja**, polega na ukrywaniu pewnych danych. Często jest tak, że tworząc jakąś klasę, nie chcemy, żeby poszczególne jej składowe, mogły zostać zmieniane z zewnątrz, ponieważ takie sytuacje mogą doprowadzić do nieprawidłowego działania naszych aplikacji. To znaczy, im więcej udostępniamy na zewnątrz, tym bardziej może to być niebezpieczne. Wyobraź sobie, że piszysz jakąś aplikację bankową i masz klasę dla konta bankowego. A w niej, pole stan konta, jest to pole dość wrażliwe i nie chcesz go udostępniać na zewnątrz, nie chcemy, żeby na przykład ktoś z zewnątrz zmienił wartość tego pola. Dlatego taki element powinien zostać ukryty wewnątrz klasy, bez możliwości zmieniania go z zewnątrz.

Jak teoria wygląda w praktyce?

```
public class Osoba {  
    private String imie;  
    private String nazwisko;
```

Należy wszystkie składowe (parametry) klasy poprzedzić słowem kluczowym **private**.

Następnie aby mieć dostęp do tych pól w klasie **Osoba** należy stworzyć metody publiczne:

```
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; }
```

Zauważ że metody te mają PREFIX **get** lub **set**. Jest to ogólnie przyjęta zasada tworzenia nazw metod pobierających (get) i ustawiających nową wartość (set).

Teraz użycie powyższego w programie:

```
Osoba osoba = new Osoba( imie: "Janek", nazwisko: "Kolar");  
String imie = osoba.getImie();
```

Tworzymy nowy obiekt **osoba**, któremu już w konstruktorze nadajemy odpowiednie własności (czyli wprowadzamy dane). Następnie przypisujemy do zmiennej **imie** wartość pobraną z obiektu za pomocą metody **getImie()**.

```
osoba.setImie("Karol");
```

W przypadku zmiany parametru **imie** w obiekcie **osoba** używamy metody **setImie()**.