Programowanie obiektowe w Javie - teoria 3

W języku Java, jeśli w klasie jednym z pól jest ArrayList i chcesz przekazywać go jako argument w konstruktorze, powinieneś zdefiniować konstruktor w następujący sposób:

```
import java.util.ArrayList;

class MyClass {
    private ArrayList<String> stringList;

    // Konstruktor
    public MyClass(ArrayList<String> stringList) {
        this.stringList = new ArrayList<>(stringList);
    }

    // Getter
    public ArrayList<String> getStringList() {
        return new ArrayList<>(stringList);
    }

    // Setter
    public void setStringList(ArrayList<String> stringList) {
        this.stringList = new ArrayList<>(stringList);
    }
}
```

W powyższym kodzie, konstruktor przyjmuje ArrayList jako argument. W konstruktorze, getterze i setterze używamy konstruktora kopiującego `new ArrayList<>(stringList)` w celu stworzenia nowej listy na podstawie przekazanej listy. Dzięki temu zapewniamy, że obiekt klasy `MyClass` operuje na swojej własnej kopii listy, co zabezpiecza go przed przypadkowym modyfikowaniem przekazanej listy przez inne części kodu. To jest ważne ze względu na bezpieczeństwo i utrzymanie poprawności danych w klasie.

Jeśli polem będzie ArrayList zamiast ArrayList, będziesz musiał zmienić typ pola, argumentu konstruktora oraz gettera i settera. Oto jak powinna wyglądać klasa z ArrayList:

```
import java.util.ArrayList;

class MyClass {
    private ArrayList<Double> doubleList;

    // Konstruktor
    public MyClass(ArrayList<Double> doubleList) {
        this.doubleList = new ArrayList<>(doubleList);
    }

    // Getter
    public ArrayList<Double> getDoubleList() {
        return new ArrayList<>(doubleList);
    }

    // Setter
    public void setDoubleList(ArrayList<Double> doubleList) {
        this.doubleList = new ArrayList<>(doubleList);
    }
}
```

W powyższym kodzie zmieniliśmy typ pola z ArrayList na ArrayList. Następnie zmieniliśmy typy argumentów konstruktora, gettera i settera na ArrayList. Reszta kodu pozostaje taka sama, ponieważ sposób przekazywania argumentów i tworzenia kopii listy nie zależy od typu elementów w liście (w tym kontekście).

Aby dodać metodę, która pozwala dodawać pojedyncze elementy do pól w klasie, możemy dodać metodę `addElement`, która przyjmuje pojedynczy element jako argument. Oto jak można to zrobić dla klas z ArrayList oraz ArrayList:

Dla klasy z ArrayList:

```
import java.util.ArrayList;

class MyClass {
    private ArrayList<String> stringList;

    // Konstruktor
    public MyClass(ArrayList<String> stringList) {
        this.stringList = new ArrayList<>(stringList);
    }

    // Getter
```

```
    public ArrayList<String> getStringList() {
        return new ArrayList<>(stringList);
    }

    // Setter
    public void setStringList(ArrayList<String> stringList) {
        this.stringList = new ArrayList<>(stringList);
    }

    // Metoda dodająca pojedynczy element
    public void addElement(String element) {
        stringList.add(element);
    }
}
```

Dla klasy z ArrayList:

```
import java.util.ArrayList;

class MyClass {
    private ArrayList<Double> doubleList;

    // Konstruktor
    public MyClass(ArrayList<Double> doubleList) {
        this.doubleList = new ArrayList<>(doubleList);
    }

    // Getter
    public ArrayList<Double> getDoubleList() {
        return new ArrayList<>(doubleList);
    }

    // Setter
    public void setDoubleList(ArrayList<Double> doubleList) {
        this.doubleList = new ArrayList<>(doubleList);
    }

    // Metoda dodająca pojedynczy element
    public void addElement(Double element) {
        doubleList.add(element);
    }
}
```

Jeśli polem klasy jest tablica stringów zamiast ArrayList, będziesz musiał zmienić typ pola, argumentu konstruktora oraz gettera i settera. Ponadto, ze względu na stałą długość tablic, będziesz musiał przekazać długość tablicy jako argument konstruktora i zainicjalizować nową tablicę o tej długości.

```
class MyClass {
    private String[] stringArray;

    // Konstruktor
    public MyClass(int arrayLength) {
        this.stringArray = new String[arrayLength];
    }

    // Getter
    public String[] getStringArray() {
        String[] copy = new String[stringArray.length];
        System.arraycopy(stringArray, 0, copy, 0, stringArray.length);
        return copy;
    }

    // Setter
    public void setStringArray(String[] stringArray) {
        this.stringArray = new String[stringArray.length];
        System.arraycopy(stringArray, 0, this.stringArray, 0, stringArray.length);
    }

    // Metoda dodająca pojedynczy element
    public void addElement(int index, String element) {
        if (index >= 0 && index < stringArray.length) {
            stringArray[index] = element;
        } else {
            throw new IndexOutOfBoundsException("Index " + index + " is out of bounds.");
        }
    }
}
```

Inna wersja:

```
class MyClass {
    private String[] stringArray;

    // Konstruktor
```

```
    public MyClass(String[] stringArray) {
        this.stringArray = new String[stringArray.length];
        System.arraycopy(stringArray, 0, this.stringArray, 0, stringArray.length);
    }

    // Getter
    public String[] getStringArray() {
        String[] copy = new String[stringArray.length];
        System.arraycopy(stringArray, 0, copy, 0, stringArray.length);
        return copy;
    }

    // Setter
    public void setStringArray(String[] stringArray) {
        this.stringArray = new String[stringArray.length];
        System.arraycopy(stringArray, 0, this.stringArray, 0, stringArray.length);
    }

    // Metoda dodająca pojedynczy element na koniec tablicy
    public void addElement(String element) {
        String[] newArray = new String[stringArray.length + 1];
        System.arraycopy(stringArray, 0, newArray, 0, stringArray.length);
        newArray[stringArray.length] = element;
        stringArray = newArray;
    }
}
```

Jeśli polem klasy będzie tablica intów, musisz zmienić typ pola, argumentu konstruktora oraz gettera i settera. Ponadto, metoda `addElement` będzie przyjmować argument typu `int` zamiast `String`.

```
class MyClass {
    private int[] intArray;

    // Konstruktor
    public MyClass(int[] intArray) {
        this.intArray = new int[intArray.length];
        System.arraycopy(intArray, 0, this.intArray, 0, intArray.length);
    }

    // Getter
```

```
    public int[] getIntArray() {
        int[] copy = new int[intArray.length];
        System.arraycopy(intArray, 0, copy, 0, intArray.length);
        return copy;
    }

    // Setter
    public void setIntArray(int[] intArray) {
        this.intArray = new int[intArray.length];
        System.arraycopy(intArray, 0, this.intArray, 0, intArray.length);
    }

    // Metoda dodająca pojedynczy element na koniec tablicy
    public void addElement(int element) {
        int[] newArray = new int[intArray.length + 1];
        System.arraycopy(intArray, 0, newArray, 0, intArray.length);
        newArray[intArray.length] = element;
        intArray = newArray;
    }
}
```