Algorytmy i struktury danych

Przemysław Stokłosa

13 marca 2020

Plan wykładu

1. Plan wykładu
2. Typy generyczne
3. Zalety stosowania typów generycznych
4. Ćwiczenia 1
5. Ograniczenie parametrów typów
6. Ćwiczenia 2

Typy generyczne

Typy generyczne umożliwiają tworzenie algorytmów bez konieczności używania zdefiniowanych typów danych (np. String, Integer, Double, Boolean). Nowe klasy itp. mogą być definiowane przy pomocy parametrów przybierających dowolne wartości a nie ściśle określone typy danych.

```
class AnimatedFigure{
    private Object figure;
    public void set(Object object){
        this.figure = object;
    }

    public void setupAnimation(){
        ...
    }
}
```

```
class AnimatedFigure<T>{
    private T figure;
    public void set(T object){
        this.figure = object;
    }

    public void setupAnimation(){
        ...
    }
}
```

Zalety stosowania typów generycznych

- Brak konieczności stosowania castingów
- Mocniejsza kontrola typów na poziomie kompilacji
- Możliwość projektowania algorytmów niezależnych od typów

Zalety stosowania typów generycznych

Przykład błędnego castingu oraz kodu zawierającego typ generyczny.

Poniżej przedstawiono przykładowy fragment kodu, który nie używa typów generycznych oraz kodu, który go używa.

```
ArrayList arrayList =
    new ArrayList();

arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add("text");

String value = (String)arrayList.get(0);
```

```
ArrayList<Double> arrayList =
    new ArrayList<>();

arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add(10.0);
arrayList.add("text");

String value = arrayList.get(0);
```

Kod po lewej stronie kompiluje się, jednak po uruchomieniu wyrzuca wyjątek. Kod po prawej stronie nie kompiluje się.

Ćwiczenie 1

Proszę poprawić kod po prawej stronie.

Ćwiczenia

Proszę wykonać ćwiczenia zawarte w repozytorium Zajecia28:
Link do repozytorium

Ograniczenie parametrów typów

Jeżeli projektujemy algortym działający na pewnej grupie funkcji możemy zastosować ograniczenie parametrów typu. Przykładowo chcielibyśmy, żeby figury z poprzedniego przykładu były animowane:

```
class AnimatedFigure<T extends Figure>{...}
```

Wykorzystajmy w tym celu metodę getShape() klasy Figure:

```
abstract public class Figure {
    Group group;

    Figure(Group group){
        this.group = group;
    }

    abstract void initFigure();
    abstract void showFigure();

    abstract Shape getShape();
}
```

```
public class AnimatedFigure<T extends Figure> {
    T figure;
    RotateTransition animation;
    AnimatedFigure(T figure,
        RotateTransition animation){
        this.figure = figure;
        this.animation = animation;

        animation.setNode(this.figure.getShape());
    }
    ...
}
```

Ćwiczenia

Proszę wykonać ćwiczenia zawarte w repozytorium Zajecia29:
Link do repozytorium