# JAVA Standard Edition

Mikołaj Jaskulski

# Programowanie obiektowe

1. **Klasy – program też jest klasą**
2. Obiekty – instancje klas
3. Korzystanie z obiektów – String
4. Dziedziczenie
5. Wyjątki
6. Interfejsy
8. Przykład
7. Cykl życia obiektu

1. **Klasa - program też jest klasą**
2. **Zmienne → pola**
3. **Funkcje → metody**
4. **Konstruktory**
5. **Przeciążanie metod (i konstruktorów)**
6. **Przysłanianie pól**
7. **final:**
   **klasa – nie można dziedziczyć**
   **metoda – nie można nadpisać**
   **zmienna – nie można zmienić wartości**
8. **static - Math, String**

```
public class Square implements Figure {

    protected double sideA;

    public Square(double sideA) {
        this.sideA = sideA;
    }

    public double countField() {
        return this.sideA * this.sideA;
    }

    public double countPerimeter() {
        return this.sideA * 4.0;
    }
}
```

# Programowanie obiektowe

1. Klasy – program też jest klasą
2. **Obiekty – instancje klas**
3. Korzystanie z obiektów – String
4. Dziedziczenie
5. Wyjątki
6. Interfejsy
8. Przykład
7. Cykl życia obiektu

1. **Tworzenie instancji → new**
2. **Obiekt posiada stan oraz zachowanie**
3. **Porównywanie → == vs equals()**
4. **Krótko o hashCode()**

# Programowanie obiektowe

1. Klasy – program też jest klasą
2. Obiekty – instancje klas
3. **Korzystanie z obiektów – String**
4. Dziedziczenie
5. Wyjątki
6. Interfejsy
8. Przykład
7. Cykl życia obiektu



1. **Łączenie stringów**
2. **Wyszukiwanie**
3. **Rozdzielanie**
4. **Prównywanie → equals vs ==**
5. **Regex**

# Programowanie obiektowe

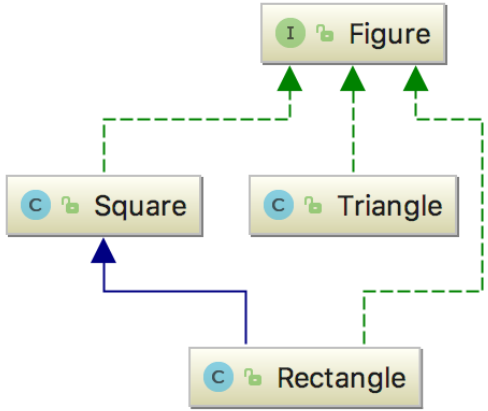
1. Klasy – program też jest klasą
2. Obiekty – instancje klas
3. Korzystanie z obiektów – String
4. **Dziedziczenie**
5. Wyjątki
6. Interfejsy
8. Przykład
7. Cykl życia obiektu

1. **extends**
2. **Można dziedziczyć po jednej klasie**
3. **Modyfikatory dostępu: public, default, protected, private**
4. **Rzutowanie, polimorfizm**
5. **instanceof**
6. **Dobre praktyki**

# Programowanie obiektowe

1. Klasy – program też jest klasą
2. Obiekty – instancje klas
3. Korzystanie z obiektów – String
4. **Dziedziczenie**
5. Wyjątki
6. Interfejsy
8. Przykład
7. Cykl życia obiektu

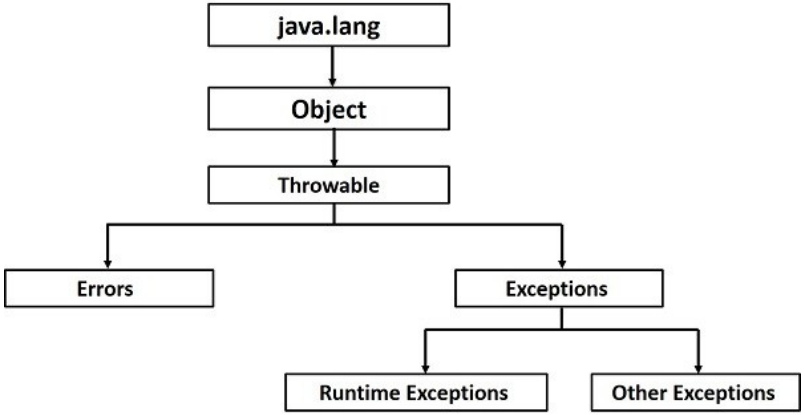
## Polimorfizm




```
public double countSumOfFields(Figure[] figures) {
    double sum = 0.0;
    for (Figure figure : figures) {
        sum += figure.countField();
    }
    return sum;
}
```

# Programowanie obiektowe

1. Klasy – program też jest klasą
2. Obiekty – instancje klas
3. Korzystanie z obiektów – String
4. Dziedziczenie
5. **Wyjątki**
6. Interfejsy
8. Przykład
7. Cykl życia obiektu




1. **Wyjątek jest klasą**
2. **throw, catch, finally**
3. **catch lub deklaracja – wyjątek trzeba obsłużyć**
4. **Dobre praktyki**

# Programowanie obiektowe

1. Klasy – program też jest klasą
2. Obiekty – instancje klas
3. Korzystanie z obiektów – String
4. Dziedziczenie
5. Wyjątki
6. **Interfejsy**
8. Przykład
7. Cykl życia obiektu



1. **Interfejs jest wymuszeniem na klasie określonego zachowania (wymuszamy implementację metod)**

2. **implements**

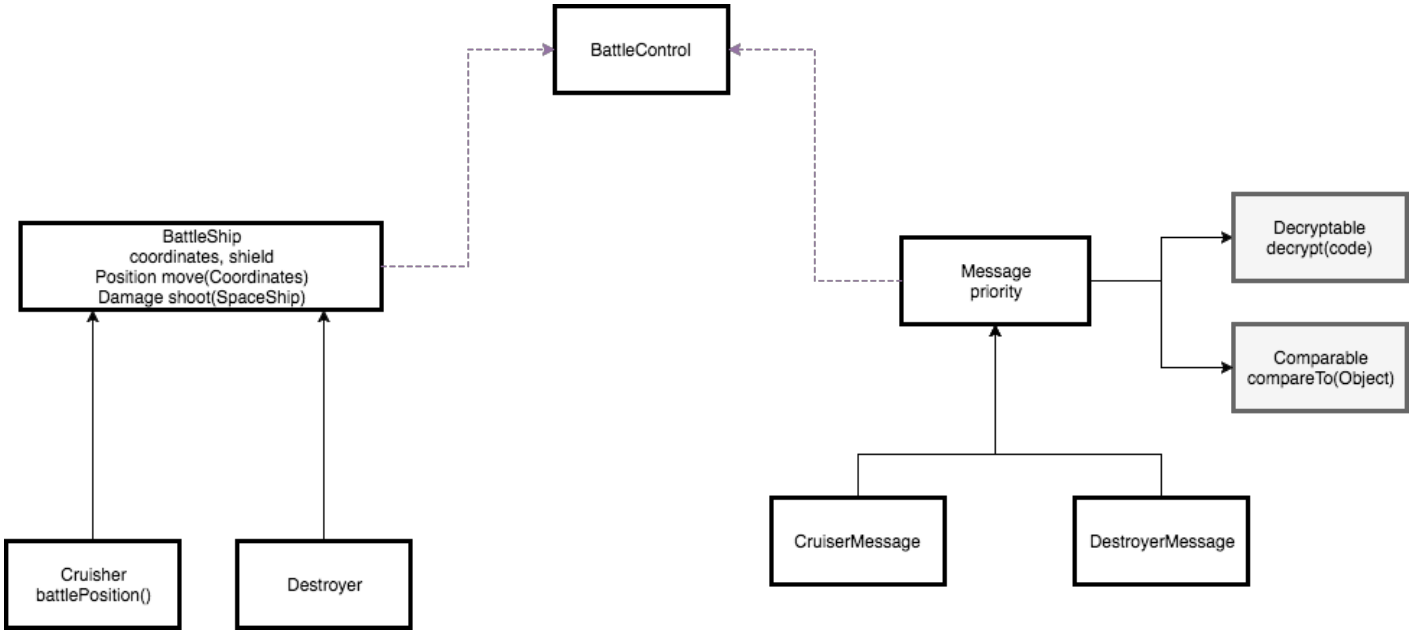
3. **Można implementować wiele interfejsów**

4. **Rzutowanie, polimorfizm, instanceof również tu działają**

5. **Dobre praktyki**
   - **enkapsulacja / hermetyzacja**
   - **klasa oraz metoda powinny robić jedną rzecz**
   - **złożone warunki ukrywamy w metodach**
   - **enum lub stałe zamiast „magic numbers”**

# Programowanie obiektowe

1. Klasy – program też jest klasą
2. Obiekty – instancje klas
3. Korzystanie z obiektów – String
4. Dziedziczenie
5. Wyjątki
6. Interfejsy
8. **Przykład**
7. Cykl życia obiektu



# Programowanie obiektowe



1. **new – stworzenie obiektu**

2. **Obiekt żyje do póki istnieje do niego referencja**

3. **Garbage Collector usuwa z pamięci obiekty**

# Dziękuję!