

13. Lekcja

Klasy abstrakcyjne, generyczne interfejsy i polimorfizm



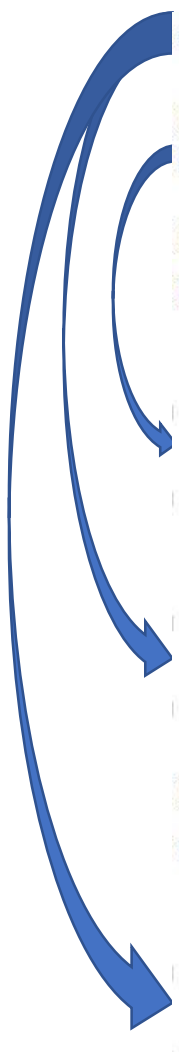
Interfejs

Interfejs w kontekście programowania w języku Java to zestaw metod bez ich implementacji (bez kodu definiującego zachowanie metody). Właściwa implementacja metod danego interfejsu znajduje się w klasie implementującej dany interfejs.

W języku Java do definiowania interfejsów używamy słowa kluczowego „interface”. Interfejsy, podobnie jak klasy, definiujemy w osobnych plikach. Nazwa pliku musi odpowiadać nazwie interfejsu.



Implementacja interfejsu



```
public interface Usable {  
    void use();  
}  
  
public interface Fixable {  
    void fix();  
}  
  
public class Hammer implements Usable, Fixable {  
  
    @Override  
    public void fix() {  
        System.out.println("The hammer is put on handle");  
    }  
  
    @Override  
    public void use() {  
        System.out.println("The nail is hammered");  
    }  
}  
  
public class Screwdriver implements Usable {  
  
    @Override  
    public void use() {  
        System.out.println("The screw is screwed in");  
    }  
}
```

The diagram illustrates the implementation of interfaces. Blue arrows point from the `implements` keyword in the class declarations to the interfaces they implement. Specifically, one arrow points from `Hammer implements Usable, Fixable` to the `Usable` interface, and another points from the same line to the `Fixable` interface. A third arrow points from `Screwdriver implements Usable` to the `Usable` interface.

- Do zaimplementowania interfejsu do klasy służy słówko „implements” po deklaracji klasy.
- Od tej pory klasa musi posiadać wszystkie metody zadeklarowane w interfejsie.
- Istnieje możliwość implementacji więcej niż jednego interfejsu przez klasę.

Klasa abstrakcyjna



To klasa która nie może zostać zainicjalizowana w żaden sposób, innymi słowy nie istnieje żaden obiekt który jest bezpośrednio tej klasy. Może jednak zostać wykorzystana do dziedziczenia.

- mogą zawierać metody abstrakcyjne, czyli takie, które nie posiadają implementacji (ani nawet nawiasów klamrowych)
- może zawierać stałe (zmiennne oznaczone jako public static final)
- mogą zawierać zwykłe metody, które niosą jakąś funkcjonalność, a klasy rozszerzające mogą ją bez problemu dziedziczyć
- klasy rozszerzające klasę abstrakcyjną muszą stworzyć implementację dla metod oznaczonych jako abstrakcyjne w klasie abstrakcyjnej
- metod abstrakcyjnych nie można oznaczać jako statyczne (nie posiadają implementacji)

Dziedziczenie klasy abstrakcyjnej

```
public abstract class Tool {  
  
    protected String brand;  
    protected double weight;  
    protected double prize;  
  
    public abstract double buy(double money);  
  
    public abstract void getName();  
  
}  
  
public class Screwdriver extends Tool implements Usable {  
  
    @Override  
    public void use() { System.out.println("The screw is screwed in"); }  
  
    @Override  
    public double buy(double money) {  
        if(money >= prize){  
            System.out.println("Sold");  
            return money - prize;  
        } else {  
            System.out.println("Not enough money");  
            return money;  
        }  
    }  
  
    @Override  
    public void getName() {  
        System.out.println("Im a Screwdriver");  
    }  
}  
  
public class Hammer extends Tool implements Usable, Fixable {  
  
    @Override  
    public void fix() {  
        System.out.println("The hammer is put on handle");  
    }  
  
    @Override  
    public void use() { System.out.println("The nail is hammered"); }  
  
    @Override  
    public double buy(double money) {  
        System.out.println("The hammer is free");  
        return money;  
    }  
  
    @Override  
    public void getName() {  
        System.out.println("Im a Hammer");  
    }  
}
```

Klasy generyczne



- Klasy generyczne są to klasy o parametryzowanych typach danych. Klasy generyczne posiadają kompletną implementację, jednak nie definiują typów danych wykorzystanych w tej implementacji.
- Klasy generyczne pozwalają na jednoczesne osiągnięcie elastyczności i bezpieczeństwa programów komputerowych.
- Klasa generyczna staje się szablonem do tworzenia innych klas bez zwracania uwagi na typ danych.

Implementacja klasy generycznej

```
public class ToolBox {  
  
    private Tool content;  
  
    public ToolBox(Tool tool) {  
        this.content = tool;  
    }  
  
    public void doAdvertisement() {  
        System.out.println("Super offer. Buy me only for " + content.prize + " zł");  
    }  
}
```

```
public class ToolBox<T extends Tool> {  
  
    private T content;  
  
    public ToolBox(T tool) {  
        this.content = tool;  
    }  
  
    public void doAdvertisement() {  
        System.out.println("Super offer. Buy me only for " + content.prize + " zł");  
    }  
}
```

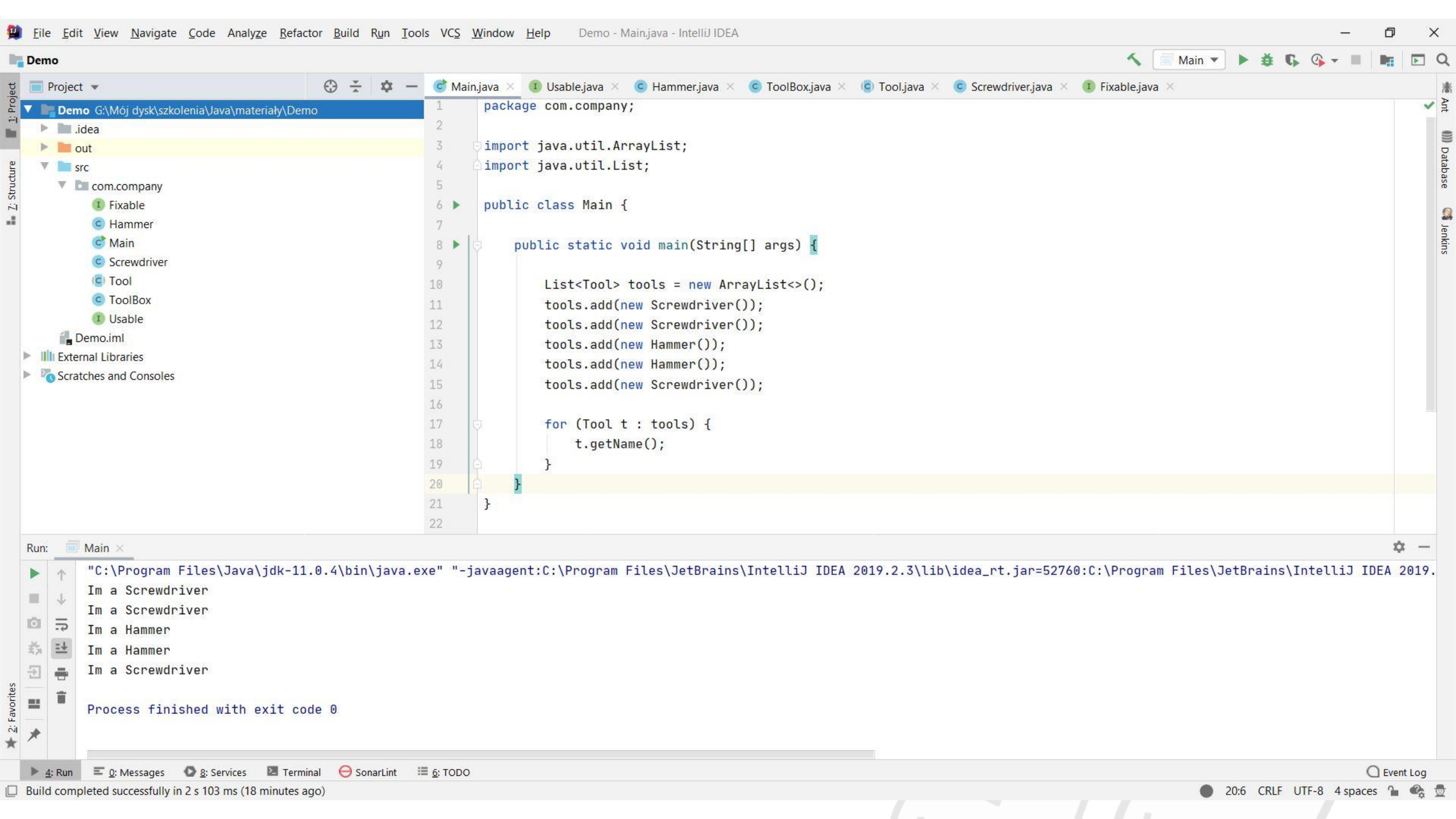
```
ToolBox boxWithTool = new ToolBox(new Hammer());  
boxWithTool.doAdvertisement();
```

```
ToolBox<Hammer> boxWithHammers = new ToolBox<>(new Hammer());  
boxWithHammers.doAdvertisement();  
ToolBox<Screwdriver> boxWithScrewdriver = new ToolBox<>(new Hammer());  
boxWithScrewdriver.doAdvertisement();  
ToolBox boxWithTool = new ToolBox(new Hammer());  
boxWithTool.doAdvertisement();
```

Polimorfizm

Mechanizmy pozwalające programiście używać wartości, zmiennych i podprogramów na kilka różnych sposobów.





Koniec podstaw programowania 😊

