
# Podstawy programowanie w javie cz. 2

Trener: Mariusz Pawłowski

Gdańsk, 13 stycznia 2018 roku

# Kolekcje

- Kolekcja to zbiór obiektów
  - duża wydajność, dzięki odpowiednim strukturom danych
- Podstawowe typy kolekcji
  - zbiory (Set) - unikatowość elementów
  - listy  (List) - uporządkowanie elementów
  - kolejki (Queue) – kolejkowanie elementów
  - mapy (Map) – tablica klucz-wartość

# ArrayList vs LinkedList

- ArrayList
  - Uniwersalne zastosowanie
  - Inne mają zastosowanie w specyficznych przypadkach
  - Szybsze pobieranie elementów niż w LinkedList
- LinkedList
  - Przechowuje elementy jako listę powiązanych ze sobą obiektów
  - Ma przewagę w przypadku dodawania pojedynczych elementów

# Typy generyczne

- rozszerzenie języka Java pozwalające na parametryzowanie typów
- tworzenie szablonów kodu, zawierających typy parametryzowane-
- typy generyczne sprawdzane są podczas kompilacji,
- informacja o typach parametryzowanych jest usuwana z definicji klasy w runtime, typy parametryzowane zamieniane są na typy ogólne (type erasure)

# Generics

- `Set mySet = new HashSet();`
- `Set<Integer> mySet = new HashSet<Integer>();`
- `Set<Integer> mySet = new HashSet<>();`
- od javy 1.5

# Interfejs

*nazwa interfejsu*

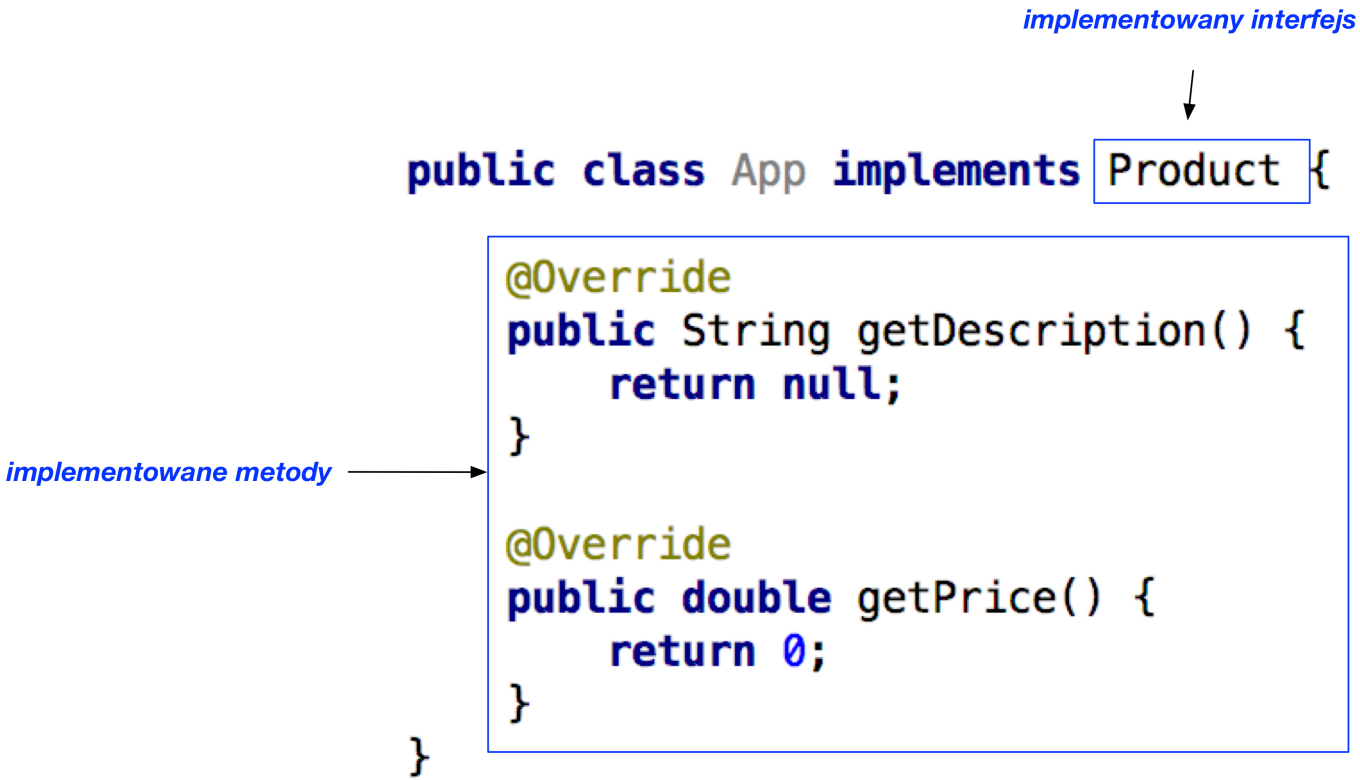
```
public interface Product {

    String getDescription();

    double getPrice();
}
```

*metody w interfejsie*

# Interfejs


implementowany interfejs
public class App implements Product {
@Override
public String getDescription() {
return null;
}
@Override
public double getPrice() {
return 0;
}
}
implementowane metody

# Zadanie 15

- Zaimplementuj generyczną metodę statyczną, która kopiuje elementy z tablicy do listy

# Zadanie 16

- Stworzyć metodę: List<String> distinct(List<String> list) { }, która usunie duplikaty z Listy.
  - Należy użyć Set – obie kolekcje implementują Collection, czyli posiadają konstruktory kopiujące Collection(Collection).

# Zadanie 16a (domowe)

- Stworzyć metodę która zliczy wystapienia elementów w liście.

# Zadanie 17

- Stworzyć klasę Person (String name, String surname, String pesel).
- Stworzyć kolekcję Map<String, Person> persons = new HashMap();
- Dodać 4 przykładowe osoby do której kluczem jest pesel.
- Wydrukować wszystkie pesele z kolekcji
- Stwórz listę osób i posortuj ją alfabetycznie

# Zadanie 17a (domowe)

- Stworzyc klasę Book (tytuł, imię i nazwisko autora, rok wydania, wydawnictwo)
- Stworzyc klase PozycjaNaPolce
- Stworzyć mapę <PozycjaNaPolce, Ksiazka>
- Wypisać wszystkie zajętę półki

# Sortowanie - TreeSet

- TreeSet to struktura, która automatycznie przechowuje elementy posortowane (sortowanie naturalne lub poprzez komparator)
- Element, który wstawiamy do TreeSet musi implementować interfejs Comparable<TypElementu>

# Zadanie 18

- Stwórz TreeSet posortowany rosnąco, wg nazwiska, korzystając z kodu z poprzedniego zadania

# Zadanie 18a

- Stwórz TreeSet z posortowny wg imion i nazwisk

# == vs equals

- == referencje
- equals – sprawdza czy obiekty są równe

# Zadanie 19

- Stwórz dwie zmienne String a1 = „A1” oraz String a2 = „A1”
- Sprawdź co zwróci porównanie equals, a co zwróci ==

# Niemutowalność

- Klasy, których instancje nie moga zmienić swojego stanu
- String, Integer, BigDecimal, …
- JVM tworzy pule obiektów w celach optymalizacyjnych

# Hash code i equals

- hash code - liczba całkowita, reprezentująca instancję klasy
- dwa obiekty o tym samym hash code – mogą być równe
- dwa równe obiekty – muszą mieć ten sam hash code
- wykorzystywane np. jako klucz w mapie
- jeśli implementujemy je samodzielnie – zawsze obydwie jednocześnie

# Obsługa wyjątków

```
try {
    // wykonywany kod, który może powodować wyjątek
} catch (Exception e){
    // zachowanie w przypadku wystąpienia wyjątku
} finally {
    // zachowanie po wykonaniu try lub catch
}
```

# Checked exceptions vs runtime exceptions

- Runtime exceptions
  - Metoda nie musi deklarować, że może rzucić tego typu wyjątek
  - Przykłady: IllegalArgumentException, IllegalStateException, NullPointerException, ...
- Exception – tzw. Checked exception
  - Metoda musi deklarować możliwość rzucenia wyjąteku
  - Obowiązkowa obsługa wyjątku w `try {} catch (Exception e)`
  - Przykłady: IOException, ...

# Zadanie 20

- Napisać metodę void foo(int a), która w przypadku kiedy a == 0 rzuci wyjątek IllegalArgumentException
- Wywołaj metodę z błędnym parametrem
- Zamień wyjątek IllegalArgumentException, na Exception i obsłuż go w bloku try {} catch() {}.

# Zadanie 20a

- Zdefiniuj własny wyjątek (Dziedziczący po RuntimeException)
- Sprawdz parametry przekazywane do aplikacji, w przypadku gdy nie ma zadnych parametrów to rzuc ten wyjątek

# Wyjątki – dobre praktyki

- obsługa wyjątków
  - blok try catch
  - przekazanie dalej jako RuntimeException lub zalogowanie

# Strumienie

- I/O Stream = dowolne wejście lub wyjście
- Plik dyskowy, Inny program, ...
- Wszystkie strumienie dziedziczą z klas InputStream and OutputStream
- Wszystkie rodzaje strumieni posiadają takie samo API i ich idea jest podobna

# Strumienie znakowe

- Wszystkie znakowe strumienie dziedziczą z klas Reader i Writer

# Zadanie 21

- Stwórz CopyCharacters przepisując CopyFile tak, aby korzystało ze znakowych strumieni.

# Buforowane strumienie wejścia/wyjścia

- Buferowane strumienie „owijają” strumienie np.: znakowy, w celach optymalizacyjnych.
  - inputStream = new BufferedReader(new FileReader("xanadu.txt"));

# Zadanie 22

- Stwórz CopyLines przepisując CopyCharacters tak, aby używała buforowanych strumieni.

# Zadanie 23

- Napisz metodę diff(String fileName1, String fileName2), która wypisze na ekranie różnicę pomiędzy dwoma plikami
- Linie dodane – występuje w fileName2, a nie ma jej w fileName1 powinna być poprzedzona „+”
- Linie usunięte – występuje w fileName1, a nie ma jej w fileName2, powinna być poprzedzona „-”

# Optional<T> vs null

- Optional jest to kontener na obiekty, które mogą być null'em.
- Optional nie chroni nas przed NullPointerException, ale pozwala jasno w API oznaczyć, że jakaś wartość może być null'em.

# Optional<T> - podstawowe API

- Tworzenie obiektu Optional
  - Optional.of(T value) – jeżeli value==null – wyjątek NullPointerException
  - Optional.ofNullable(T value)
  - Optional.empty – zwraca pusty obiekt Optional

- Pobieranie wartości z Optional
  - isPresent – true jeżeli Optional zawiera jakąkolwiek wartość
  - get() – pobiera wartość lub NoSuchElementException w przypadku null
  - orElse(T other) – zwraca wartość lub other
  - orElseThrow(Exception) zwraca wartość lub rzuca wyjątek

# Zadanie 24

- Po ściągnięciu zadania z git huba zastąp wartości opcjonalne “opakowaniem” Optional

# Wątki

- 3 sposoby na uruchamianie wątków
  - Dziedziczenie po klasie Thread
  - Implementacja interfejsu Runnable i użycie klasy Thread do jego uruchomienia
  - Implementacja interfejsu Runnable i użycie implementacji ExecutorService

# Synchronizacja

- Przy odczycie / zapisie tych samych obiektów przez więcej niż jeden wątek
- Element *{obiekt synchronizujący}* to referencja do obiektu, na którym synchronizowany będzie kod *{kod synchronizowany}*.

# Zadanie 25

- Napisz aplikację, która uruchamia wątki w sposoby opisane na wcześniejszym slajdzie

# Dziękuję za uwagę