Programowanie komputerów I

Biblioteka standardowa

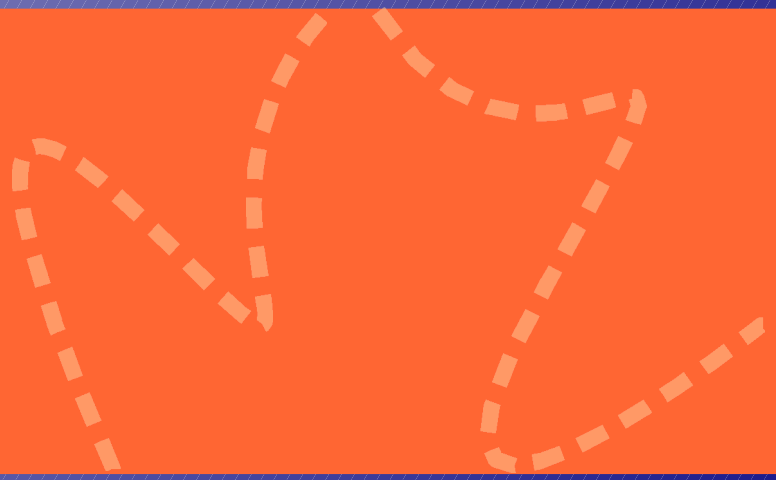
Pakiet `java.lang`

Willy Picard
Katedra Technologii Informacyjnych
Akademia Ekonomiczna w Poznaniu
<*picard@kti.ae.poznan.pl*>

Agenda

- Cel(e) wykładu
- Odświeżenie i przekąski
- Przegląd pakietu `java.lang`
- Klasa `Object`
- Klasy `String` i `StringBuffer`
- Klasy `System` i `Runtime`
- Podsumowanie

Cel(e) wykładu



Przegląd wykładu

Podstawowe pojęcia

- 1: Wprowadzenie
- 2: Podstawowe struktury danych & instrukcje
- 3: Programowanie obiektowe I
- 4: Programowanie obiektowe II
- 5: Programowanie obiektowe III
- 6: Zaawansowane struktury danych
- 7: Wątki & Wyjątki

CELE WYKŁADU

Przegląd wykładu

Java

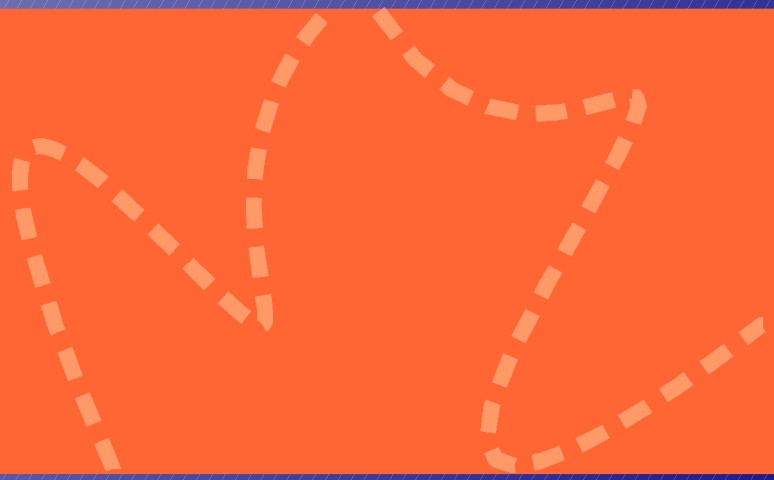
- 8: Przykład podsumowujący
- 9: Pakiet standardowy
- 10: Interfejsy graficzne – AWT
- 11: Interfejsy graficzne – Swing
- 12: Programowanie We/Wy
- 13: Programowanie sieciowe
- 14: JAR & Refleksja
- 15: Podsumowanie

CELE WYKŁADU

Cel na dziś

Wprowadzić
standardową
bibliotekę
w pakiecie
`java.lang`

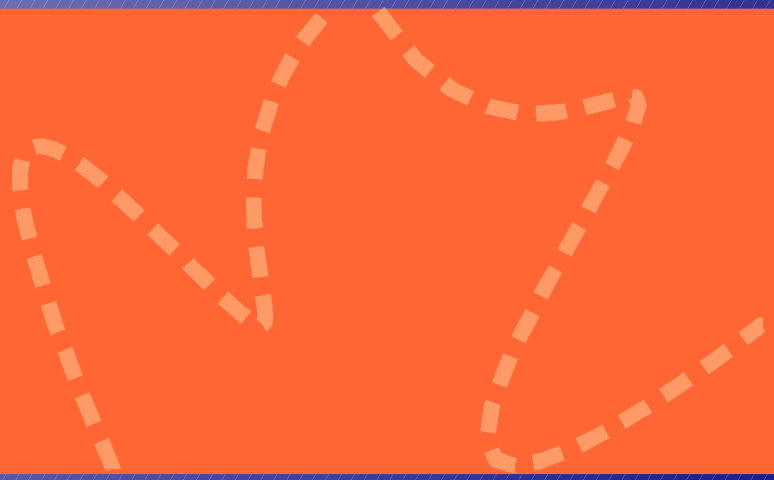
Odświeżenie i przekąski



The API Specification

- Dokumentacja
  - Pakietów
  - Interfejsów
  - Klas
  - Dziedziczenia
  - Atrybutów
  - Metod

Przegląd pakietu
`java.lang`

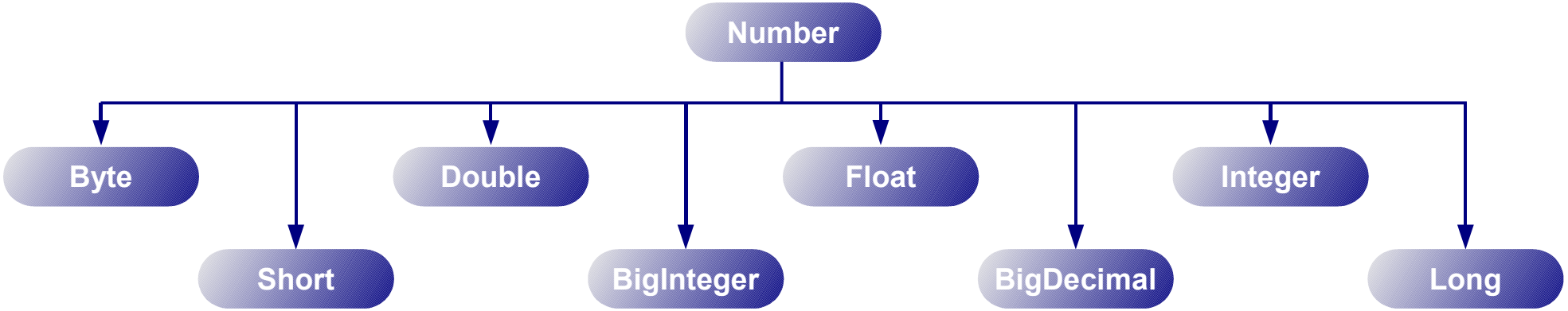


W API: `java.lang`

Zawiera klasy które są podstawowe dla języka programowania Java

Obiektowe otoczki typów podstawowych

- Number
  - Byte
  - Double
  - Float
  - Integer
  - Long
  - Short
  - BigDecimal
  - BigInteger
- Boolean
- Character
- Void



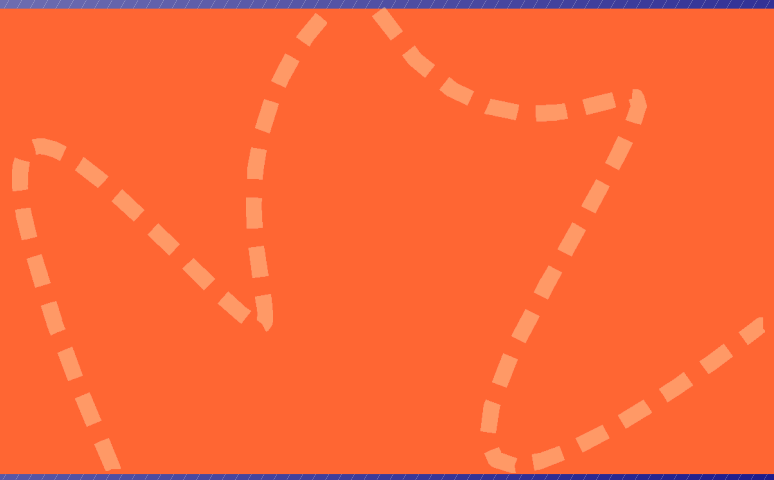
Dlaczego obiektowe otoczki?

- Błędny przykład (do J2SE 1.6)
  - List mojeLiczby = new ArrayList();
  - int a = 1;
  - ~~mojeLiczby.add(a);~~
- Poprawny przykład
  - List mojeLiczy = new ArrayList();
  - Integer a = new Integer(1);
  - mojeLiczby.add(a);

Inne klasy

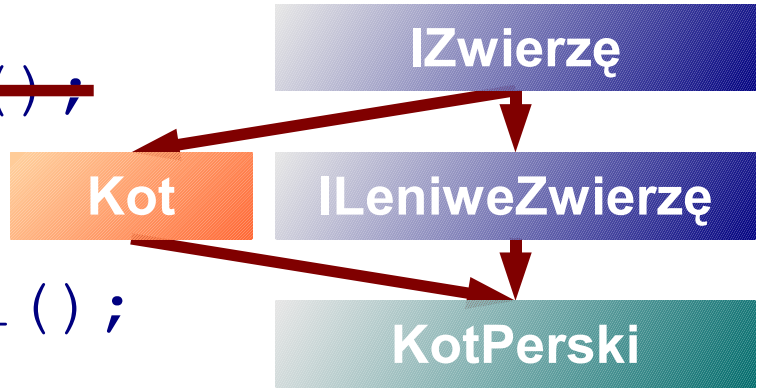
- Klasa `Object`
- Wsparcie dla łańcuchów znaków
  - Klasa `String`
  - Klasa `StringBuffer`
- Interacja z JVM
  - Klasa `System`
  - Klasa `Runtime`

Klasa `Object`



Jeden obiekt, wiele postaci w Javie

- ~~IZwierzę kot = new IZwierzę();~~
- IZwierzę kot = new Kot();
- IZwierzę kot = new KotPerski();
- ~~IZwierzę kot = new ILeniweZwierzę();~~
- Kot kot = new Kot();
- Kot kot = new KotPerski();
- ILeniweZwierzę kot = new KotPerski();
- KotPerski kot = new KotPerski();



Klasa `Object`

- `Object kot = new Kot();`
- Każda klasa rozszerza klasę `Object`
- Klasa `Object`
  - Zbiór metod
  - Pusty konstruktor

Metody klasy `Object`

- Metoda `toString()`
  - `System.out.println(mójObiekt);`
- Metoda `equals()`
  - Porównanie obiektów
- Metoda `hashCode()`
  - Funkcja haszująca (funkcja skrótu)
  - Musi być zaimplementowana razem z metodą `equals()`
  - Równe obiekty muszą mieć równy skrót

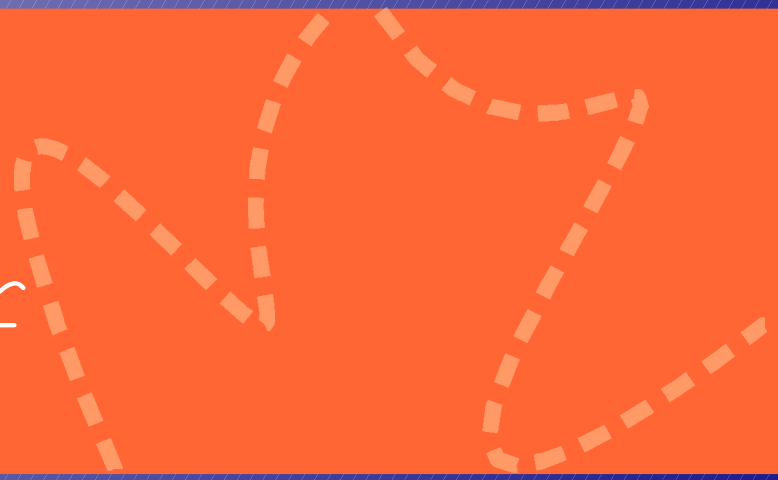
Metoda `hashCode`

- `int wynik = 17;`
- Dla każdego znaczącego atrybutu `f`, oblicz `c`
  - Boolean → `c = ( f ? 0 : 1)`
  - byte, char, short, int → `c = (int) f;`
  - Long → `c = (int) (f ^ (f >>> 32));`
  - Float → `c= Float.floatToIntBits(f)`
  - Double →
```
d = Double.doubleToLongBits(f);
c = (int) (d ^ (d >>>32));
```
  - Object → `c = ((f == null)? 0:f.hashCode());`
  - `wynik = 37*wynik+c;`

Przykład

Pełna wersja Kota

Klasa `String`
Klasa `StringBuffer`



Klasa `String`

- Łańcuchy znaków
- Stałe
- String jako tablica
  - `length()`
  - `charAt()`
- Przykład

```
String name=”Willy Picard”;
int length = name.length(); // 12
char c = name.charAt(0);    // 'W'
```

Klasa `String`

- Podstawowe metody
  - `indexOf()`
  - `substring()`
- Przykład

```
String nazwa = "Willy-Picard";
int indeks = nazwa.substring("Pic");
String początek = nazwa.substring(0,indeks);
   //Willy-
String koniec = nazwa.substring(indeks);
   //Picard
```

Klasa `StringBuffer`

- Zmienne łańcuchy znaków
- Metoda `append()`

```
String spacja = " ";
StringBuffer buff = new StringBuffer();
buff.append("Willy");
buff.append(spacja);
buff.append("Picard");
```

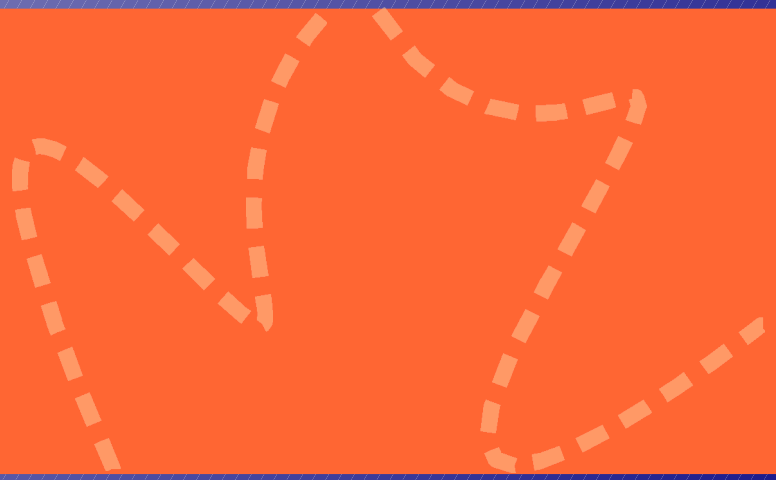
- Metoda `toString()`

```
String nazwa = buff.toString();
```

Przykład

Przykład łańcuchów znaków

Klasa System
Klasa Runtime



Klasa `System`

- Zbiór pożytecznych pól
- Brak konstruktora
- Wyłącznie pola statyczne (ang. *static*)
- Trzy atrybuty
  - in: standardowe wejście
  - out: standardowe wyjście
  - err: standardowe wyjście błędów

Przykład

Przykład standardowych wejścia/wyjść

Właściwości a `System`

- Zbiór właściwości nt. środowiska
- Pary (Klucz, Wartość)
- Informacje nt.
  - JVM
  - Systemu operacyjnego
  - Użytkownika
  - Separatory
    - Linii, Pliku, Ścieżki

Przykład

Przykład właściwości systemowych

Inne metody

- Kopiowanie tablic
- Obecny czas
  - `currentTimeMillis()`
  - Liczba milisekund od 1 stycznia 1970 UTC
- Odśmiecacz (ang. *Garbage Collector*)
  - `gc()`
  - Próba wymuszenia sprzątania pamięci

Przykład

Przykład obecnego czasu

Klasa `Runtime`

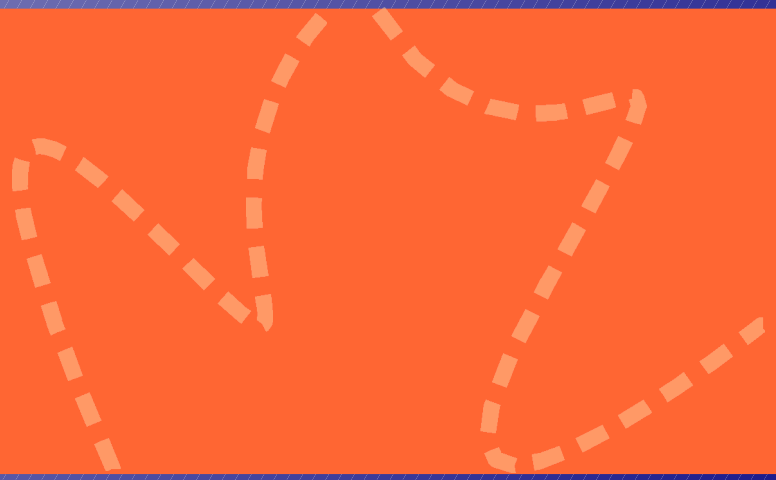
- `Runtime.getRuntime()`
  aby otrzymać instancję
- Informacje nt. pamięci
  - `freeMemory()`
  - `maxMemory()`
  - `totalMemory();`
- Uruchomienie zewnętrznych programów
- UWAGA!
  - **Wieloplatformowość**

Przykład

Przykład `Runtime`

SYSTEM

Podsumowanie



Standardowa biblioteka Javy

- Dokumentacja w API!!!
- Podstawowe klasy

Do zobaczenia za tydzień