Programowanie komputerów I

Archiwa Javy (JAR)

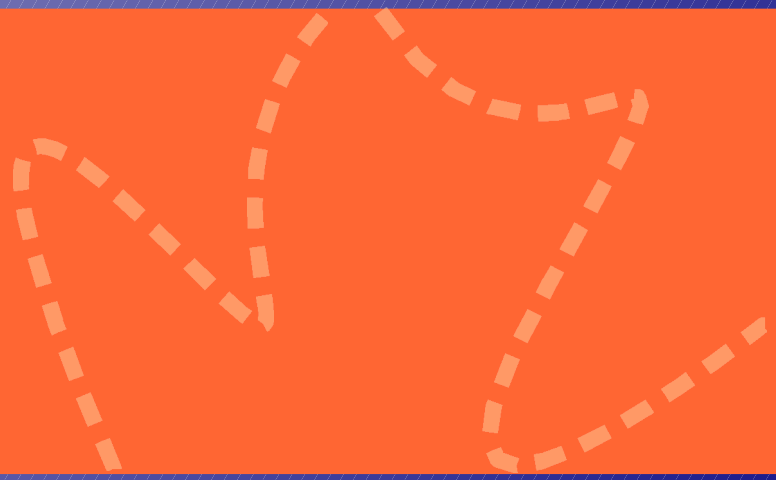
Refleksja

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

Agenda

- Cel(e) wykładu
- CLASSPATH
- Archiwa Javy (JAR)
- Refleksja
- Przykłady
- Podsumowanie

Cel(e) 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

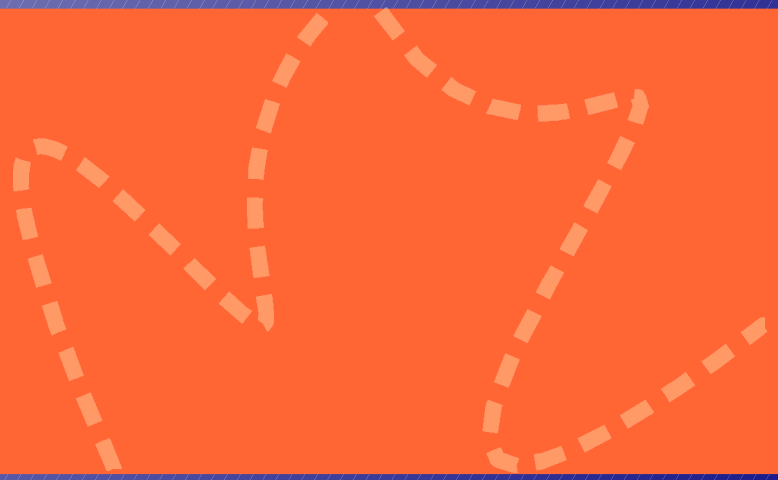
CELE WYKŁADU

Cel na dziś

Wprowadzić archiwa (JAR) i refleksję w Javie

CELE WYKŁADU

CLASSPATH



Uruchomienie programów w Javie

- W Eclipse
- Za pomocą JDK
  - `java <pełnaNazwaKlasy> <argumenty>`
- Przykład
  - `java pl.kti.CompProg.BasketDemo`
  - `java pl.kti.CompProg.Calc 12 64`
- Uwaga!
  - Kompilacja `klasa.java → klasa.class`
  - Interpretacja `java klasa`

CLASSPATH

CLASSPATH

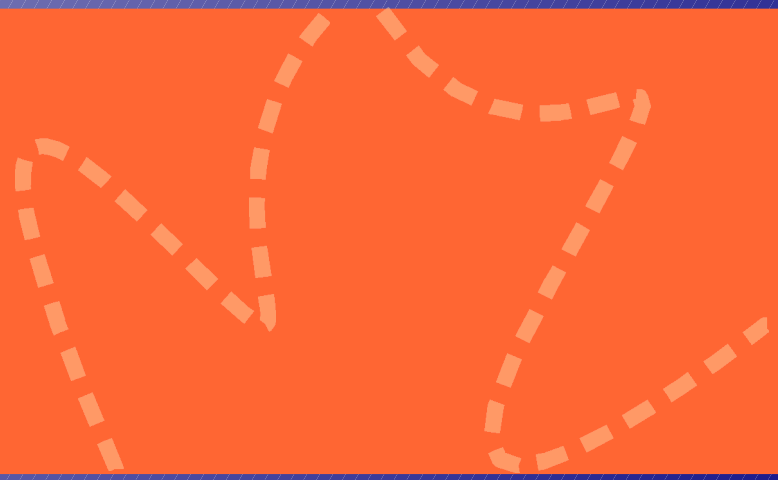
- Potrzebne dla `java` by znaleźć klasy
- Lista katalogów i archiwów klas
- Archiwa
  - Pliki Zip i Jar
- Separatory
  - ; (Windows) lub : (Unices)
- Przykład
  - `set CLASSPATH=.;c:\Fridge;c:\fridge.jar`
  - `export CLASSPATH=.:/fridge:/fridge.jar`

Pakiety i katalogi

- `set CLASSPATH=.;c:\Fridge`
- Odwzorowywanie
  - Pakiet `pl.kti.CompProg`
  - Katalog `pl\kti\CompProg`

  - Klasa `pl.kti.CompProg.Fridge`
  - Plik `pl\kti\CompProg\Fridge.class`

Archiwa Javy (JAR)



Pliki Zip i Jar

- Zip
  - Phil Katz (PKWARE)
  - Rozszerzenie `.zip`
  - Skompresowany zbiór plików i katalogów
- Jar
  - Sun Microsystems
  - Rozszerzenie `extension.jar`
  - Pliki Zip + opcjonalny katalog META-INF

Jar Manifest

- Plik MANIFEST.MF
- W katalogu META-INF
- Struktura manifestu

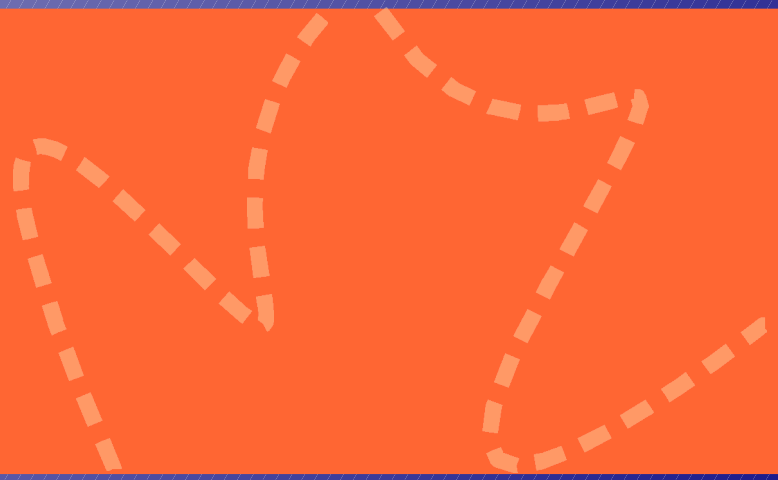
```
Manifest-Version: 1.0
Created-By: 1.2 (Sun Microsystems Inc.)

Main-Class: pl.kti.CompProg.Fridge
```

Narzędzie `Jar`

- Tworzenie pliku .jar
  - `jar cf myFile.jar Cat.class animals/`
- Tworzenie pliku .jar z manifestem
  - `jar cmf myManifestFile myFile.jar *.class`
- Rozpakowanie pliku .jar
  - `jar xf myFile.jar`
- Uaktualnienie pliku .jar
  - `jar uf myFile.jar newVersion.class`

Refleksja



Przegląd refleksji

- Manipulowanie
  - Interfejsów
  - Klas
  - Konstruktorów
  - Metod
  - Atrybutów
- The Reflection API
  - Zbiór klas reprezentujących powyższe pojęcia

Klasa `java.lang.Class`

- Otrzymanie obiektu `java.lang.Class` od obiektu
  - `IZwierzę zwierzę = new Kot();`
  - `Class k = zwierzę.getClass();`
- Uzyskanie informacji za pomocą `java.lang.Class`
  - `Class nadklasa = k.getSuperclass();`
  - `String nazwaKlasy = k.getName();`
  - `boolean jestInterfejsem = k.isInterface();`
- Tworzenie instancji `java.lang.Class`
  - `Class k = new Class("pl.Kot");`

Tworzenie obiektów

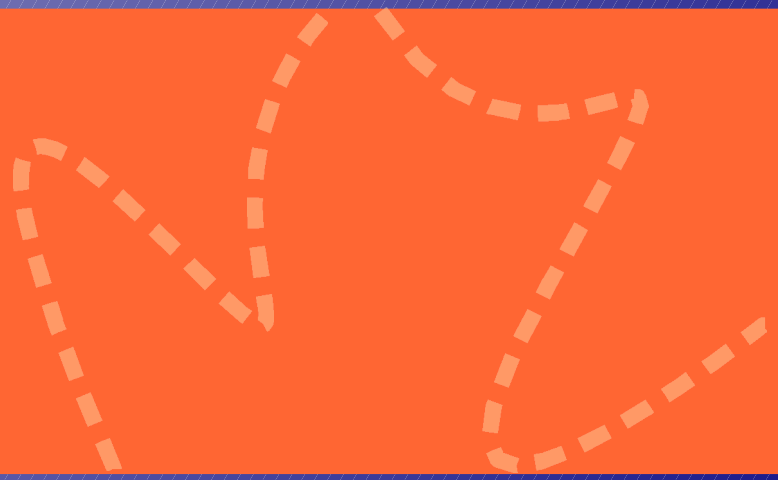
- Pusty konstruktor
  - `Object o = k.newInstance();`
- Uzyskanie informacji nt. konstruktorów
  - `Constructor[] konstry = k.getConstructors();`
  - `Class[] typParam = new Class[]{String.class};`
  - `Constructor konstr= k.getConstructor(typParam);`
- Tworzenie obiektu
  - `Object[] param = new Object[]{“Felix”};`
  - `Object kotFelix = konstr.newInstance(param);`

Wywołanie metod

- Uzyskanie informacji nt. metod
  - `Method[] metody = k.getMethods();`
  - `Class[] typParam = new Class[]{Integer.class};`
  - `Method metoda= c.getMethod("jedz", typParam);`
- Wywołanie metody
  - `Object[] param = new Object[]{new Integer(200)};`
  - `Object mójWynik = metoda.invoke(kotFelix, param);`

REFLEKSJA

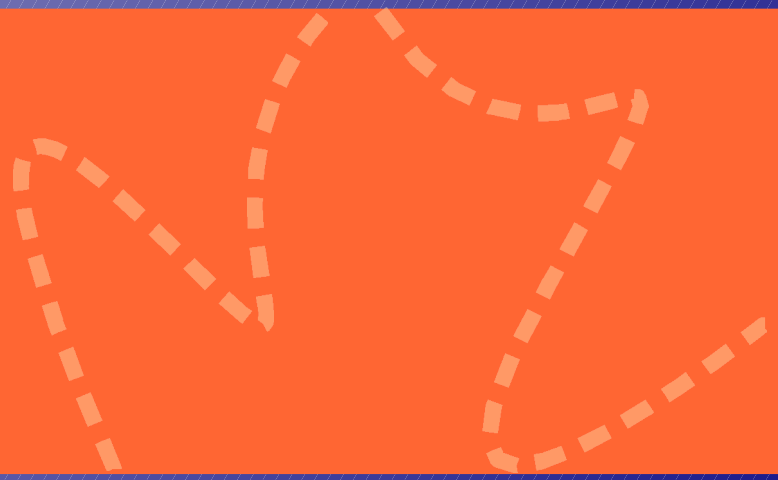
Przykłady



Przykład refleksji

Dynamiczne zwierzęta

Podsumowanie



Podsumowanie

- CLASSPATH
  - Częste źródło problemu
- Archiwa Javy
  - Kliknij, aby uruchomić! :-)
- Refleksja
  - Zaawansowana technologia
  - Dynamiczne programy

Do zobaczenia za tydzień