Klasy pomocnicze JAVA

Tomasz Kubik

Klasa `Properties`

- przechowuje pary łańcuchów: klucz – wartość
- dostarcza metod do zapisu/odczytu tych par (poprzez strumienie)
- pozwala na definiowanie wartości domyślnych (indywidualnie lub poprzez wewnętrzny słownik wartości domyślnych)

Klasa `Properties`

- Przykład deklaracji

```
Properties settings = new Properties();
```

- Przykład ustawienia klucza i wartości

```
settings.put("color", "white");
settings.put("text", "Witajcie");
```

- Zapis do pliku `my.properties`

```
FileOutputStream out = new FileOutputStream("my.properties");
settings.store(out, "Moje ustawienia");
```

- Zawartość pliku wynikowego:

```
#Moje ustawienia
#Mon Jun 19 10:00:00 2019
color=white
text=Hello
```

- Odczyt z pliku

```
FileInputStream in = new FileInputStream("my.properties");
settings.load(in);
```

Klasa `Properties`

- Ustawienie wartości domyślnych
  - w metodzie `getProperty` (jeśli klucz nie zostanie znaleziony, zwrócona zostanie wartość domyślna)

```
String title = settings.getProperty("text",
"Default text");
```

  - w wewnętrznym słowniku wartości domyślnych

```
Properties defaultSettings = new
Properties();
defaultSettings.put("color", "white");
defaultSettings.put("text", "Default text");
...
Properties settings = new
Properties(defaultSettings);
```

Klasa `Properties`

- Wykorzystanie właściwości systemowych

```
String userDir =
System.getProperty("user.home");
Properties systemSettings =
System.getProperties();
```

- Do pobrania są następujące właściwości:

```
java.version
java.vendor
java.vendor.url
java.class.version
os.name
os.version
os.archfile.separator
path.separator
line.separator
java.specification.version
```

```
java.vm.specification.version
java.vm.specification.vendor
java.vm.specification.name
java.vm.version
java.vm.vendor
java.vm.name
```

Klasa `Preferences`

- przechowuje wartości w strukturze drzewiastej
- struktura ta może mieć wiele węzłów
- węzły mają nazwy podobne do unixowych ścieżek,
  - odzwierciedlają one położenie węzłów w drzewie
  - zalecane jest: stosowanie konwencji odwróconej nazwy domenowej oraz stosowanie nazw odpowiadających nazwom pakietów
- każdy węzeł może mieć podwęzeł lub mapę klucz-wartość jako potomka
- każdy węzeł zawiera oddzielną mapę klucz–wartość, w której można przechowywać wartości typów: `String`, `int`, `long`, `float`, `double`, `boolean`, jak również tablice bajtów `byte[]` (zapisywanie tam obiektów serializowanych jest niezalecane).

Klasa `Preferences`

- Dostęp do korzenia drzewa użytkownika:

```
Preferences root = Preferences.userRoot();
```

- Dostęp do korzenia drzewa systemowego

```
Preferences root = Preferences.systemRoot();
```

- Dostęp do węzła drzewa

```
Preferences node = root.node("/com/example/myapp");
```

- Dostęp do węzła drzewa o nazwie jak pakiet wybranej klasy

```
Preferences node =
Preferences.userNodeForPackage(obj.getClass());

Preferences node =
Preferences.systemNodeForPackage(obj.getClass());
```

Klasa `Preferences`

- Przy pobieraniu wartości koniecznie należy podawać wartość domyślną na wypadek niewystąpienia klucza

```
String get(String key, String defval)
int getInt(String key, int defval
long getLong(String key, long defval)
float getFloat(String key, float defval)
double getDouble(String key, double defval)
boolean getBoolean(String key, boolean defval)
byte[] getByteArray(String key, byte[] defval)
```

- Zapis danych w węźle

```
put(String key, String value)
putInt(String key, int value)
```

- Dostęp do wszystkich kluczy zapisanych w węźle

```
String[] keys
```

  – (nie można sprawdzić, jakiego typu jest wartość dla określonego klucza)

Klasa `Preferences`

- Eksportowanie poddrzewa lub węzła w formacie XML

```
void exportSubtree(OutputStream out)
void exportNode(OutputStream out)
```

- Importowanie wyeksportowanych wcześniej danych

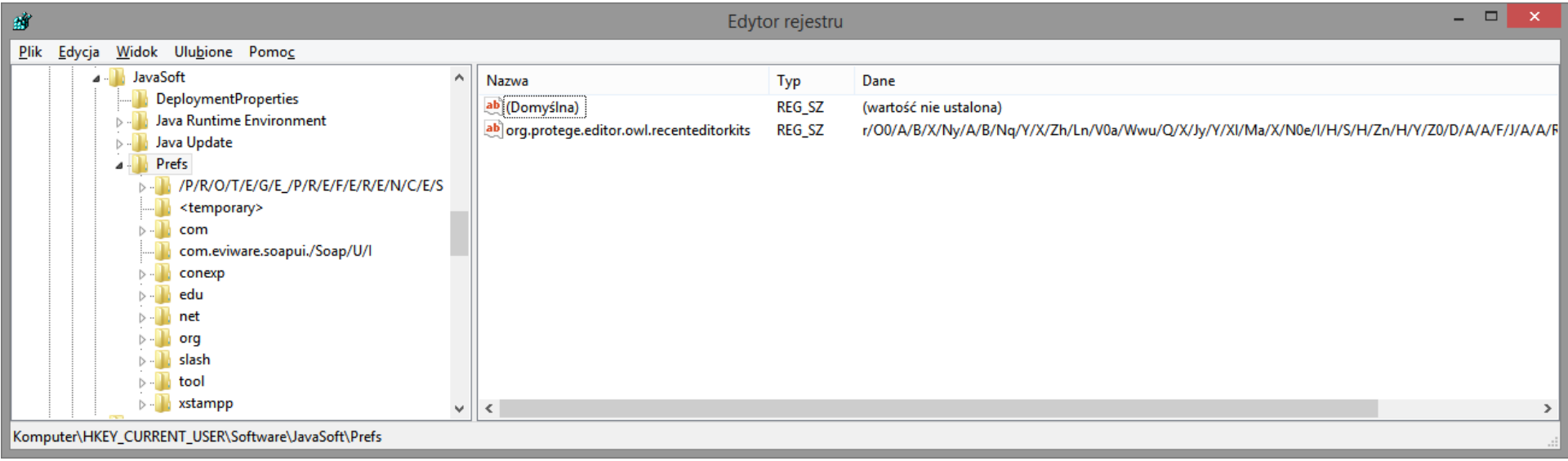
```
void importPreferences(InputStream in)
```

- Postać pliku ustawień

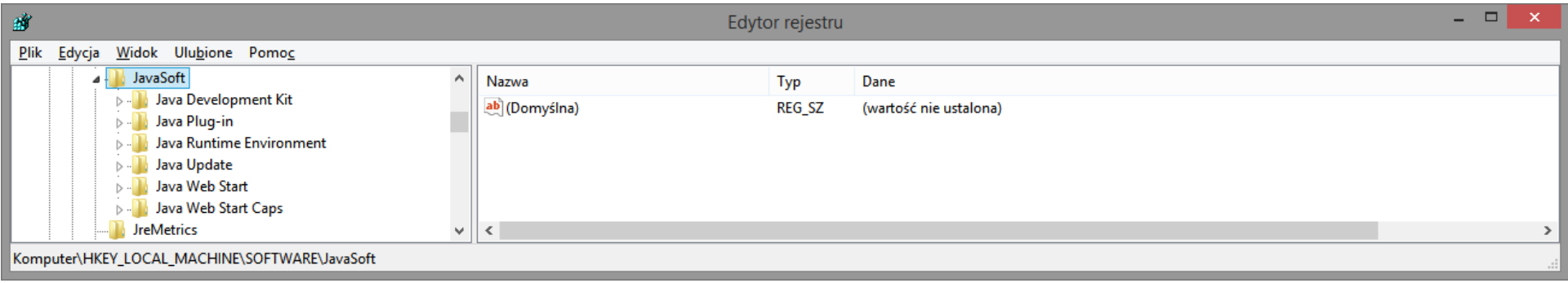
```
<?xml version="1.0" encoding="UTF-8"?>
<preferences EXTERNAL_XML_VERSION="1.0">
    <root type="user">
        <map />
        <node name="com">
            <map />
            <node name="example">
                <map>
                    <entry key="color" value="white" />
                    <entry key="text" value="Hello!" />
                </map>
            </node>
        </node>
    </root>
</preferences>
```

Klasa `Preferences`

- Windows (wpisy w rejestrze)
  - userRoot:
    HKEY_CURRENT_USER\Software\JavaSoft\Prefs



  - systemRoot:
    HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Prefs



https://stackoverflow.com/questions/4947555/where-are-the-java-preferences-stored-in-windows-7

Klasa `Preferences`

- Unix (wpisy w plikach)
  - userRoot

    `"${user.home}/.java/.userPrefs"`
  - systemRoot

    `"/etc/.java"`
- Położenie preferencji można zmienić przez wyspecyfikowanie właściwości:
  - `"java.util.prefs.userRoot"`
    `"java.util.prefs.systemRoot"`

https://docs.oracle.com/javase/8/docs/technotes/guides/preferences/overview.html

JNDI

- http://edu.pjwstk.edu.pl/wyklady/zap/scb/W11/W11.htm

- https://docs.oracle.com/javase/jndi/tutorial/trailmap.html
- https://docs.oracle.com/javase/tutorial/jndi/overview/index.html

- https://www.javaworld.com/article/2076888/jndi-overview--part-1--an-introduction-to-naming-services.html
- https://www.javaworld.com/article/2076901/jndi-overview--part-2--an-introduction-to-directory-services.html
- https://self-learning-java-tutorial.blogspot.com/2016/05/jndi-tutorial.html
- https://docs.oracle.com/javase/8/docs/index.html

Tools and utilities

- https://docs.oracle.com/javase/8/docs/technotes/tools/index.html#monitor