

JPS

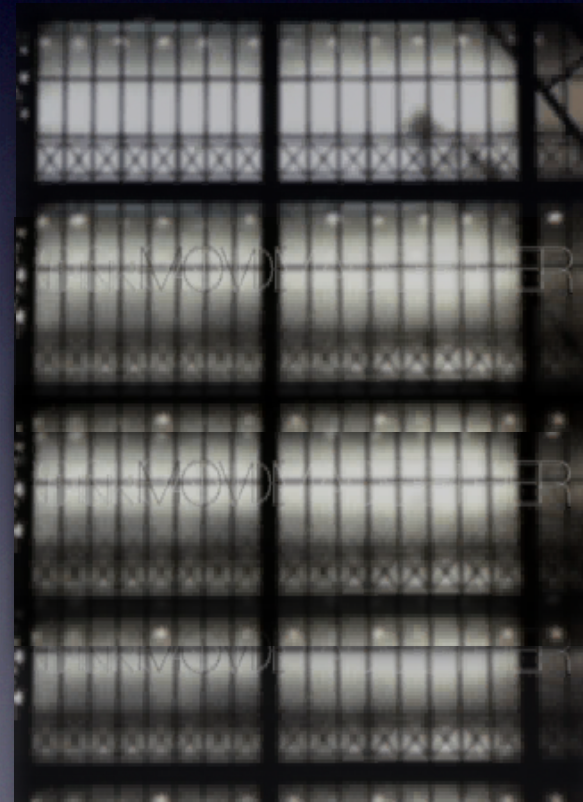
ćwiczenia 2.

Skład danych

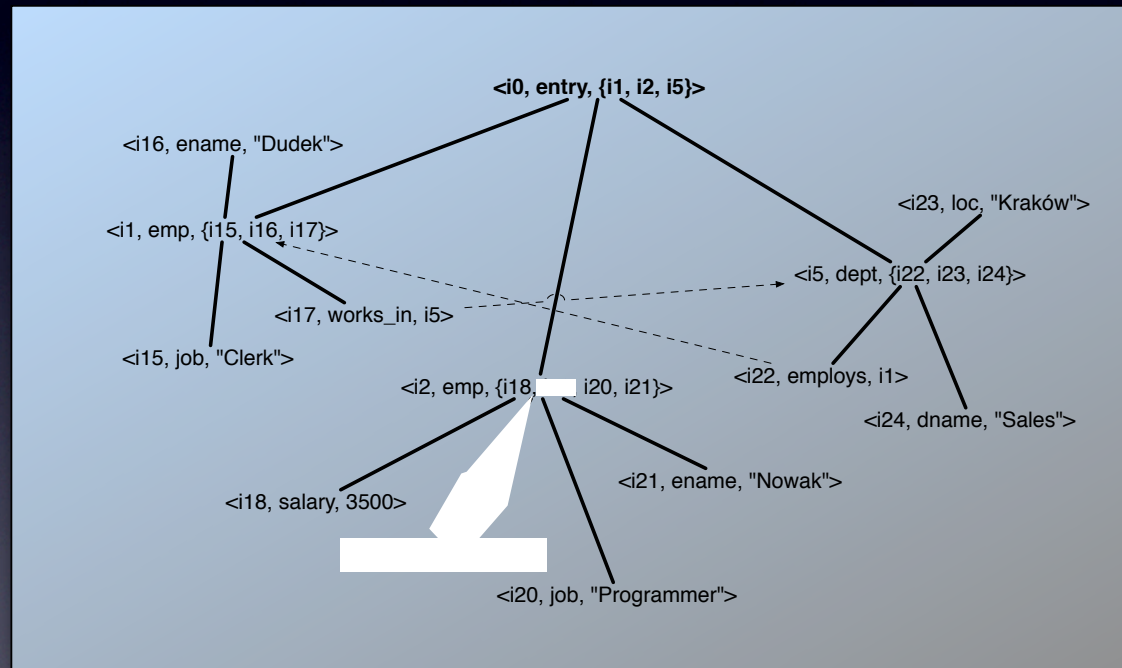


Obiekty

- Proste
<OID, nazwa, wartość>
np. <i0, imie, "Jan">, <i1, pensja, 3354>
- Złożone
<OID, nazwa, { OID1, OID2, OID3, ... }>
np. <i2, pracownik, { i0, i1 }>
- Referencyjne
<OID, nazwa, OID1>
np. <i3, pracuje_w, i4>



Baza danych



CRUD składu

- **Create**
Tworzenie obiektów w składzie.
Alokacja miejsca, aktualizacja wartości obiektu nadrzędnego.
Create(rodzaj_obiektu, nazwa, wartość, OID_nadrzędnego)
- **Retrieve**
Zwracanie wartości obiektów.
Retrieve(OID) = wartość
- **Update**
Zmiana wartości istniejących obiektów.
Update(OID, wartość)
- **Delete**
Usuwanie obiektów ze składu.
Aktualizacja obiektu nadrzędnego, usunięcie podobiektów, usunięcie referencji wskazujących na dany obiekt, zwolnienie miejsca.
Delete(OID)

Create (1)

1. Pusta baza danych
Create(COMPLEX, "entry", { }, -)
<i0, "entry", {}>
2. Tworzymy pracownika
Create(COMPLEX, "emp", { }, i0)
<i0, "entry", {i1}>
<i1, "emp", {}>
3. Tworzymy nazwisko
Create(STRING, "ename", "Kowalski", i1)
<i0, "entry", {i1}>
<i1, "emp", {i2}>
<i2, "ename", "Kowalski">
4. Tworzymy pensję
Create(INTEGER, "salary", 1000, i1)
<i0, "entry", {i1}>
<i1, "emp", {i2, i3}>
<i2, "ename", "Kowalski">
<i3, "salary", 1000>

Create (2)

1. Tworzymy drugiego pracownika

```
Create(COMPLEX, "emp", { }, i0)
<i0, "entry", {i1, i4}>
<i1, "emp", {i2, i3}>
<i2, "ename", "Kowalski">
<i3, "salary", 1000>
<i4, "emp", { }>
```

2. Tworzymy nazwisko drugiego pracownika

```
Create(STRING, "ename", "Nowak", i4);
<i0, "entry", {i1, i4}>
<i1, "emp", {i2, i3}>
<i2, "ename", "Kowalski">
<i3, "salary", 1000>
<i4, "emp", {i5}>
<i5, "ename, "Nowak">
```

Retrieve

- Baza:
 - <i0, "entry", {i1, i4}>
 - <i1, "emp", {i2, i3}>
 - <i2, "ename", "Kowalski">
 - <i3, "salary", 1000>
 - <i4, "emp", {i5}>
 - <i5, "ename, "Nowak">
- Operacje:
 - Retrieve(i0) = {i1, i4}
 - Retrieve(i1) = {i2, i3}
 - Retrieve(i2) = "Kowalski"
 - Retrieve(i3) = 1000

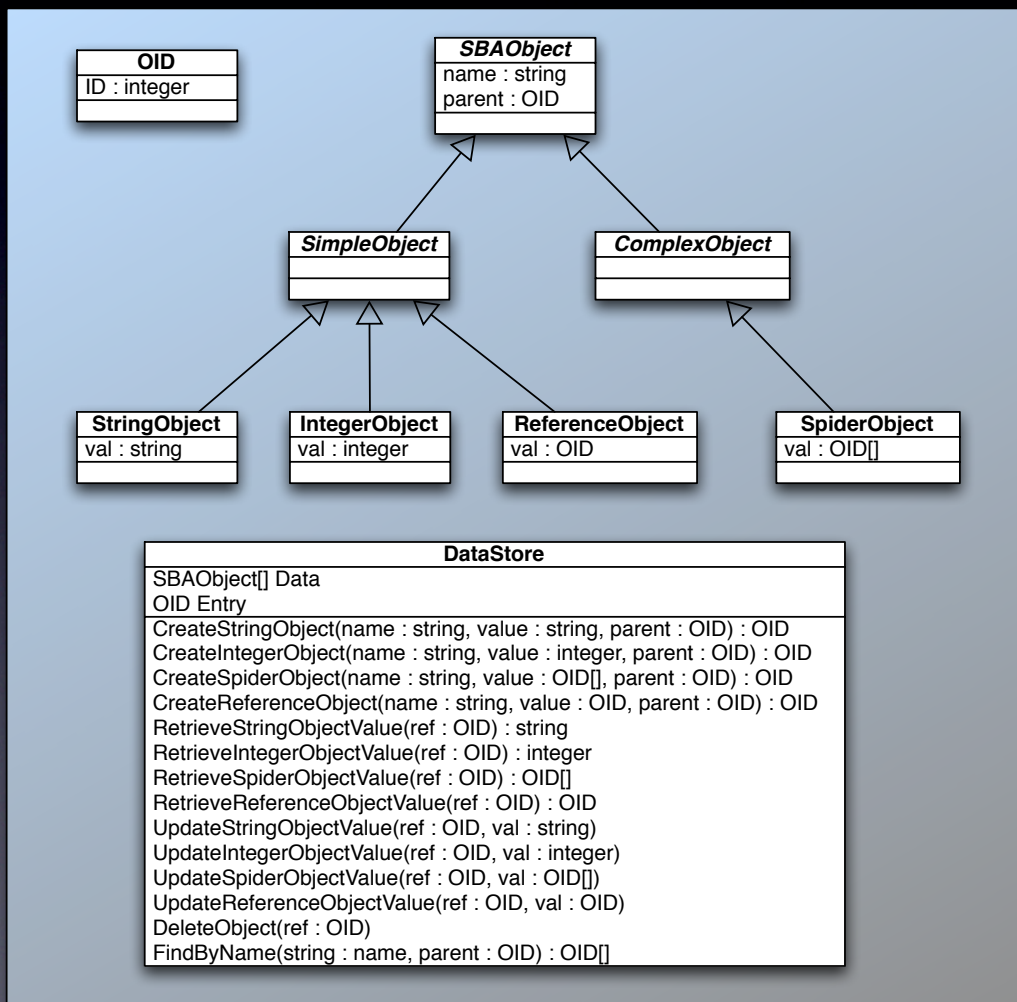
Update

- Baza:
<i0, "entry", {i1, i4}>
<i1, "emp", {i2, i3}>
<i2, "ename", "Kowalski">
<i3, "salary", 1000>
<i4, "emp", {i5}>
<i5, "ename, "Nowak">
- Aktualizacja nazwiska i pensji:
Update(i2, "Walewski")
Update(i3, 2000)
- Baza po modyfikacjach:
<i0, "entry", {i1, i4}>
<i1, "emp", {i2, i3}>
<i2, "ename", "Walewski">
<i3, "salary", 2000>
<i4, "emp", {i5}>
<i5, "ename, "Nowak">

Delete

- Baza:
<i0, "entry", {i1, i4, i7}>
<i1, "emp", {i2, i3, i6}>
<i2, "ename", "Kowalski">
<i3, "salary", 1000>
<i4, "emp", {i5}>
<i5, "ename", "Nowak">
<i6, "works_in", i7>
<i7, "dept", {i8, i9}>
<i8, "dname", "Sales">
<i9, "location", "Warsaw">
- Usunięcie pierwszego pracownika, nazwiska drugiego pracownika i departamentu:
Delete(i5) - usuwa obiekt i5
Delete(i7) - usuwa obiekt i7, wszystkie jego podobiekty i obiekt i6
Delete(i1) - usuwa obiekt i1 i wszystkie jego podobiekty
- Baza
<i0, "entry", {i4}>
<i4, "emp", { }>

Implementacija



Obiekty

```
OID.java
class OID
OID.java:13
1 //
2 // OID.java
3 // Odra
4 //
5 // Created by Michal Lentner on 05-04-17.
6 // Copyright 2005 PJIIT. All rights reserved.
7 //
8
9 package odra.store;
10
11 public class OID {
12     public int ID;
13
14     public OID(int id) {
15         ID = id;
16     }
17
18     public int hashCode() {
19         return ID;
20     }
21
22     public boolean equals(Object obj) {
23         if (obj instanceof OID && ((OID) obj).ID == ID)
24             return true;
25         else
26             return false;
27     }
28 }
29
```

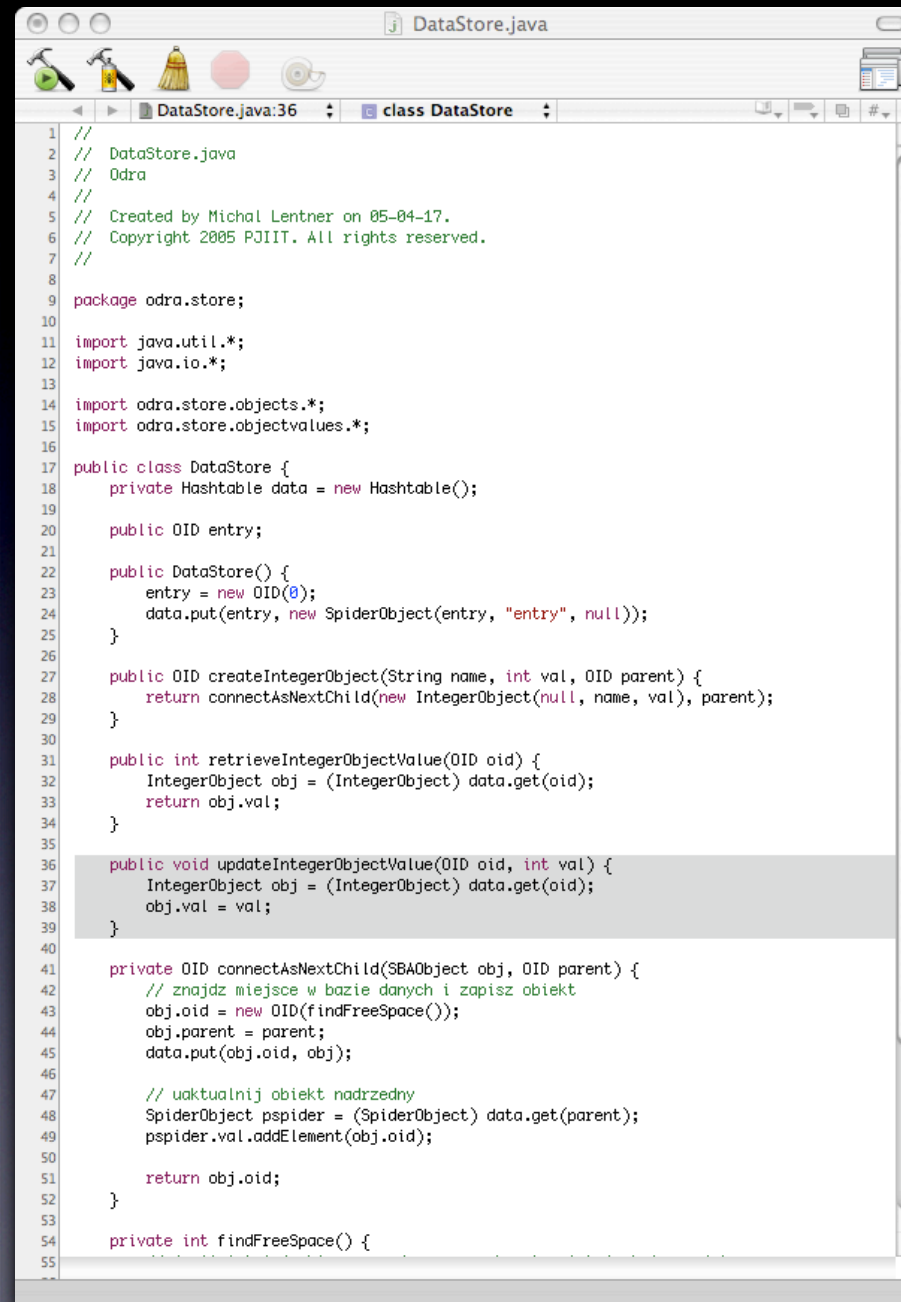
```
SBAObject.java
SBAObject.java:20
1 //
2 // SBAObject.java
3 // Odra
4 //
5 // Created by Michal Lentner on 05-04-17.
6 // Copyright 2005 PJIIT. All rights reserved.
7 //
8
9 package odra.store.objects;
10
11 import java.util.*;
12
13 import odra.store.*;
14
15 public abstract class SBAObject {
16     public String name;
17     public OID oid;
18     public OID parent;
19
20     public SBAObject(OID objid, String objname) {
21         oid = objid;
22         name = objname;
23     }
24 }
25
```

Obiekty

```
IntegerObject.java
1 //
2 // IntegerObject.java
3 // Odra
4 //
5 // Created by Michal Lentner on 05-04-17.
6 // Copyright 2005 PJIIT. All rights reserved.
7 //
8
9 package odra.store.objects;
10
11 import odra.store.OID;
12
13 public class IntegerObject extends SimpleObject {
14     public int val;
15
16     public IntegerObject(OID id, String name, int v) {
17         super(id, name);
18         val = v;
19     }
20 }
21
```

```
SpiderObject.java
1 //
2 // SpiderObject.java
3 // Odra
4 //
5 // Created by Michal Lentner on 05-04-17.
6 // Copyright 2005 PJIIT. All rights reserved.
7 //
8
9 package odra.store.objects;
10
11 import java.util.*;
12
13 import odra.store.OID;
14
15 public class SpiderObject extends ComplexObject {
16     public Vector val = new Vector();
17
18     public SpiderObject(OID id, String name, OID[] v) {
19         super(id, name);
20
21         addElements(v);
22     }
23
24     public void addElements(OID[] v) {
25         for (int i = 0; i < (v == null ? 0 : v.length); i++)
26             val.addElement(v[i]);
27     }
28
29     public OID[] valToArray() {
30         return (OID[]) val.toArray(new OID[val.size()]);
31     }
32 }
33
```


Przykład
implementacji:
tworzenie,
odczytywanie
i aktualizowanie
obiektów



```
1 //
2 // DataStore.java
3 // Odra
4 //
5 // Created by Michal Lentner on 05-04-17.
6 // Copyright 2005 PJIIT. All rights reserved.
7 //
8
9 package odra.store;
10
11 import java.util.*;
12 import java.io.*;
13
14 import odra.store.objects.*;
15 import odra.store.objectvalues.*;
16
17 public class DataStore {
18     private Hashtable data = new Hashtable();
19
20     public OID entry;
21
22     public DataStore() {
23         entry = new OID(0);
24         data.put(entry, new SpiderObject(entry, "entry", null));
25     }
26
27     public OID createIntegerObject(String name, int val, OID parent) {
28         return connectAsNextChild(new IntegerObject(null, name, val), parent);
29     }
30
31     public int retrieveIntegerObjectValue(OID oid) {
32         IntegerObject obj = (IntegerObject) data.get(oid);
33         return obj.val;
34     }
35
36     public void updateIntegerObjectValue(OID oid, int val) {
37         IntegerObject obj = (IntegerObject) data.get(oid);
38         obj.val = val;
39     }
40
41     private OID connectAsNextChild(SBAObject obj, OID parent) {
42         // znajdz miejsce w bazie danych i zapisz obiekt
43         obj.oid = new OID(findFreeSpace());
44         obj.parent = parent;
45         data.put(obj.oid, obj);
46
47         // uaktualnij obiekt nadrzedny
48         SpiderObject pspider = (SpiderObject) data.get(parent);
49         pspider.val.addElement(obj.oid);
50
51         return obj.oid;
52     }
53
54     private int findFreeSpace() {
55
```

XML

- Interpreter powinien mieć moduł ładujący dokumenty XML do składu
- Wykorzystać DOM, SAX, lub coś jeszcze innego



Ćwiczenia