11. Шаблон Пълномощно (Proxy)

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ

ДОЦ. Д-Р ЕМИЛ ДОЙЧЕВ

Общи сведения

- ✓ **Вид:** структурен за обекти
- ✓ **Цел:** Предоставя запазено място или заместител на друг обект, за да контролира достъпа до него.
- ✓ **Известен и като:** Заместник, сурогат (Surrogate)

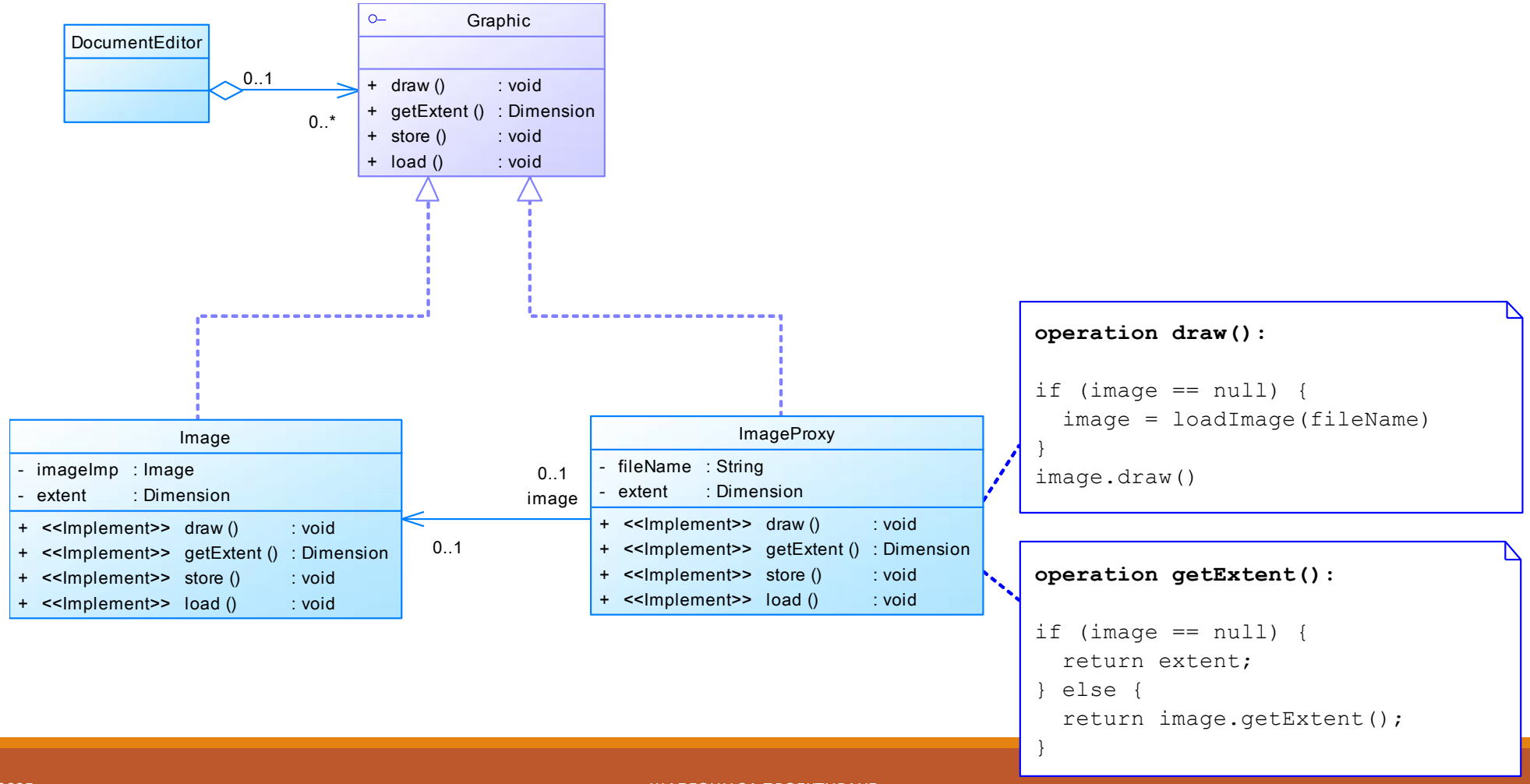
Мотивация

- ✓ Пълномощник е:
  - ✓ лице, упълномощено да извършва определени действия от името на друго лице;
  - ✓ агент или заместник;
- ✓ В определени ситуации клиента не може или не иска да работи директно с даден обект, но въпреки това трябва да взаимодейства с него.
- ✓ В тези случаи обектът „пълномощник“ може да служи за посредник между клиента и целевия обект, като:
  - ✓ пълномощникът има същия интерфейс като целевия обект;
  - ✓ пълномощникът съдържа референция към целевия обект и може да препраща заявките към него;
  - ✓ в резултат пълномощникът е оторизиран да взаимодейства с целевия обект от името на клиента.

Мотивация

- ✓ Някои графични обекти, от рода на големите растерни изображения, може да са тежки за създаване.
- ✓ Решението е да се използва друг обект, пълномощно (proxy), който да служи за заместител на истинското изображение. Пълномощното играе ролята на изображението и се грижи за инстанцирането му, когато е необходимо.
- ✓ Пълномощното на изображението създава истинското изображение само когато редакторът на документи поиска то да се изобрази.

Мотивация


DocumentEditor
0..1
0..*
Graphic
+ draw () : void
+ getExtent () : Dimension
+ store () : void
+ load () : void
Image
- imageImp : Image
- extent : Dimension
+ <<Implement>> draw () : void
+ <<Implement>> getExtent () : Dimension
+ <<Implement>> store () : void
+ <<Implement>> load () : void
0..1
image
0..1
ImageProxy
- fileName : String
- extent : Dimension
+ <<Implement>> draw () : void
+ <<Implement>> getExtent () : Dimension
+ <<Implement>> store () : void
+ <<Implement>> load () : void
operation draw():
if (image == null) {
image = loadImage(fileName)
}
image.draw()
operation getExtent():
if (image == null) {
return extent;
} else {
return image.getExtent();
}

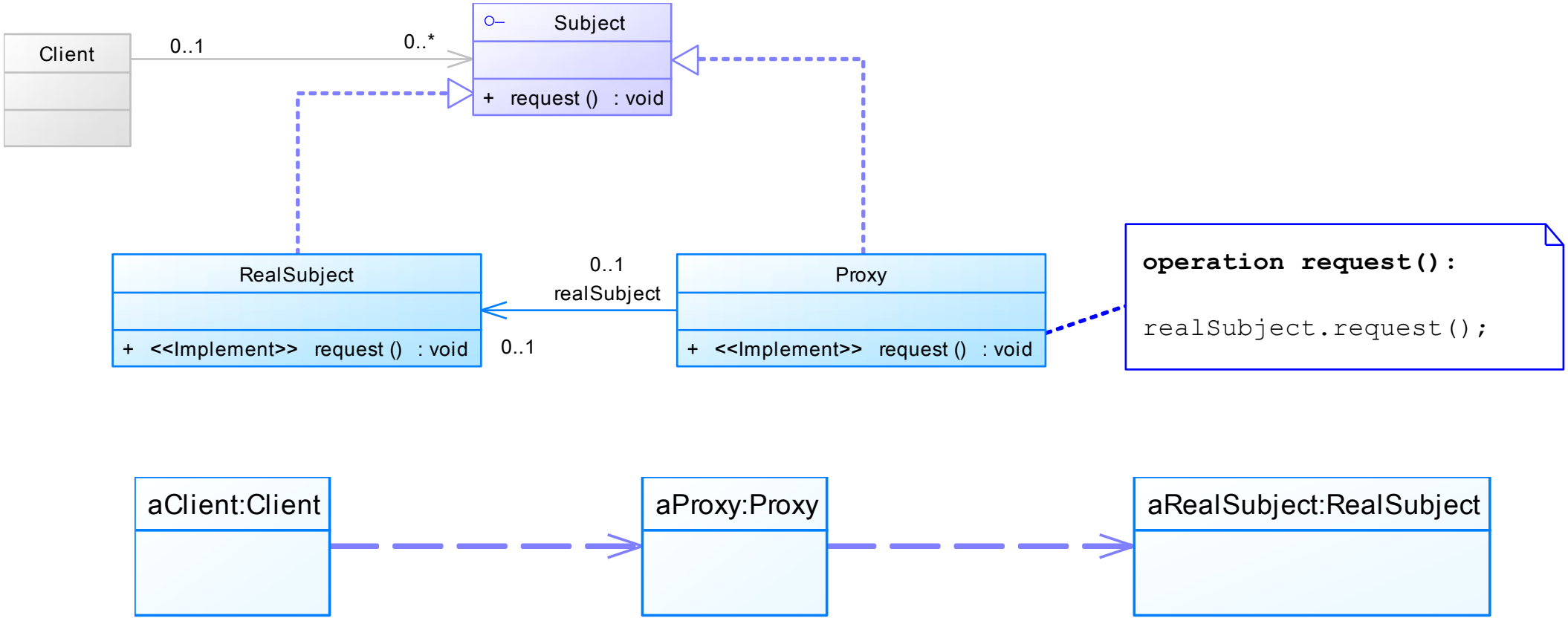
Приложимост

- ✓ **Приложимост:** Шаблонът Пълномощно се използва в случаите когато обикновен указател към някой обект не е достатъчен и е необходимо изграждане на по-гъвкава или по-сложна връзка към него.
- ✓ Обичайните ситуации, в които е приложим шаблонът Пълномощно, могат да се класифицират по следния начин:
  - ✓ **Отдалечено пълномощно (remote proxy)** – предоставя локален представител на обект от друго адресно пространство.
  - ✓ **Виртуално пълномощно (virtual proxy)** – позволява създаването на „тежки“ обекти (които изискват много ресурси) при поискване (on demand). Реалните обекти не се създават докато не са наистина необходими.
  - ✓ **Копиране при запис (copy on write proxy)** – отлага копирането (cloning) на голям и сложен обект докато не се наложи модификация на едно от копията.
  - ✓ **Защитно пълномощно (protection proxy)** – предоставя различни нива на достъп (до целевия обект) на различните клиенти – т.е. осъществява специфичен контрол на достъпа.

Приложимост (продължение)

- ✓**Кеш пълномощно (cache proxy)** – предоставя място за временно съхранение на резултатите от „тежка“ целева операция така, че да могат да бъдат използвани от множество клиенти.
- ✓**Firewall proxy** – защитава целевия обект от „лоши“ клиенти (или обратното).
- ✓ **Смарт референция (Smart reference)** – заместител на прост указател, който извършва допълнителни действия при осъществяване на достъп до обекта. Например:
  - ✓ поддържане на брояч на референциите към истинския обект, така че той да може да бъде освободен автоматично когато брояча достигне 0;
  - ✓ зареждане на статичен обект в паметта при първото му използване;
  - ✓ проверка преди осъществяването на достъп дали обекта не е заключен за да се гарантира, че никой друг обект не може да го промени.

Структура


Client
0..1
0..*
Subject
+ request () : void
RealSubject
+ <<Implement>> request () : void
0..1
realSubject
0..1
Proxy
+ <<Implement>> request () : void
operation request():
realSubject.request();
aClient:Client
aProxy:Proxy
aRealSubject:RealSubject

Участници

- ✓ **Proxy** (ImageProxy)
  - поддържа референция, даваща възможност на пълномощното да достига до реалния обект;
  - предоставя интерфейс, идентичен на този на Subject, така че пълномощното да може да бъде заменено от реалния обект;
  - контролира достъпа до реалния обект и може да бъде отговорен за създаването и изтриването му.
- ✓ **Subject** (Graphic) – дефинира общия интерфейс за RealSubject и Proxy, така че Proxy да може да се използва навсякъде, където се очаква RealSubject.
- ✓ **RealSubject** (Image) – дефинира реалния обект, представляван от пълномощното.

Взаимодействия

- ✓ Когато е необходимо, Proxy препредава заявки към RealSubject в зависимост от вида на пълномощното.

Следствия

- ✓ При отдалечено пълномощно може да се скрие факта, че даден обект се намира в друго адресно пространство.
- ✓ Виртуалното пълномощно може да извърши оптимизации – напр. създаване на обект при поискване.
- ✓ Защитните пълномощни и смарт референциите дават възможност за извършване на допълнителни системни задачи при осъществяване на достъп до обектите.
- ✓ При „копиране при запис“ тежката операция по клониране на голям и сложен обект може да бъде отложена или да не се изпълни никога ако копието на обекта никога не се променя.

Пример: Copy-On-Write Proxy

- ✓ **Сценарий**: имаме голяма колекция от обекти, напр. хеш-таблица, която трябва да може да бъде достъпвана от множество клиенти едновременно. Един от клиентите иска да изпълнява последователно извличане на данните като не позволява на другите да променят съдържанието на хеш-таблицата през това време.

Пример: Copy-On-Write Proxy

- ✓ **Решение 1:** Да се използва възможността за заключване на достъпа до обекта докато продължават операциите по извличане на данните. След това достъпа се отключва отново.

```
public void doFetches(Hashtable ht) {
  synchronized(ht) {
    // Do fetches using ht reference.
  }
}
```

- ✓ **Проблем**: Този метод може да заключи достъпа до обекта за прекалено дълго време като не позволява на други нишки достъп до него.

Пример: Copy-On-Write Proxy

- ✓ **Решение 2:** Клиента прави копие (клонинг) на колекцията преди да извърши операциите по извличане на данните. Това предполага, че колекцията може да бъде клонирана и че предоставя операция за дълбоко копиране на цялото ѝ съдържание.
- ✓ Напр. java.util.Hashtable предоставя clone метод, който копира самата хеш таблица, но не ѝ ключовете и обектите-стойности в нея.

```
public void doFetches(Hashtable ht) {
  Hashtable newht = (Hashtable) ht.clone();
  // Do fetches using newht reference.
}
```

- ✓ Заключването на достъпа до колекцията се извършва само докато се копира самата колекция. След това операциите по извличане на данните се извършват върху копието без да се блокира достъпа до оригиналната колекция.
- ✓ **Проблем**: ако никоя друга нишка не извършва промени по колекцията се губи смисъла от скъпата операция по копирането.

Пример: Copy-On-Write Proxy

- ✓ **Решение 3:** Би било добре ако можем да копираме колекцията само ако се наложи промяна в нея. Например клиента, който ще извършва операции по извличане на данните, извиква clone() метода, но реално копиране не се извършва докато някой друг не се опита да модифицира колекцията. Това е *copy-on-write* операцията.
- ✓ Това решение може да се реализира, чрез използване на пълномощник.
- ✓ Примера е написан от Mark Grand в книгата *Patterns in Java, Volume 1*

Пример: Copy-On-Write Proxy

- ✓ Класът пълномощник е LargeHashtable. Когато се извика clone() метода на пълномощника се връща копие на пълномощника, като двата пълномощника референцират една и съща хеш таблица. Когато някой от пълномощниците модифицира колекцията, тя се копира.
- ✓ Класът ReferenceCountedHashTable се използва за да са наясно пълномощниците, че работят със споделена хеш таблица. Този клас пази информация за броя на пълномощниците, които използват споделената колекция.

Пример: Copy-On-Write Proxy

```
// The proxy.
public class LargeHashtable extends Hashtable {

  // The ReferenceCountedHashTable that this is a proxy for.
  private ReferenceCountedHashTable theHashTable;

  // Constructor
  public LargeHashtable() {
    theHashTable = new ReferenceCountedHashTable();
  }

  // Return the number of key-value pairs in this hashtable.
  public int size() {
    return theHashTable.size();
  }
```

Пример: Copy-On-Write Proxy

```
// Return the value associated with the specified key.
public synchronized Object get(Object key) {
  return theHashTable.get(key);
}

// Add the given key-value pair to this Hashtable.
public synchronized Object put(Object key, Object value) {
  copyOnWrite();
  return theHashTable.put(key, value);
}

// Return a copy of this proxy that accesses the same Hashtable.
public synchronized Object clone() {
  Object copy = super.clone();
  theHashTable.addProxy();
  return copy;
}
```

Пример: Copy-On-Write Proxy

```
// This method is called before modifying the underlying
// Hashtable.  If it is being shared then this method clones it.
private void copyOnWrite() {
  if (theHashTable.getProxyCount() > 1) {
    synchronized (theHashTable) {
      theHashTable.removeProxy();
      try {
        theHashTable =  (ReferenceCountedHashTable)
                         theHashTable.clone();
      } catch (Throwable e) {
         theHashTable.addProxy();
      }
    }
  }
}
...
```

Пример: Copy-On-Write Proxy

```
// Private class to keep track of proxies sharing the hash table.
private class ReferenceCountedHashTable extends Hashtable {

  private int proxyCount = 1;

  // Constructor
  public ReferenceCountedHashTable() {
    super();
  }

  // Return a copy of this object with proxyCount set back to 1.
  public synchronized Object clone() {
    ReferenceCountedHashTable copy;
    copy = (ReferenceCountedHashTable)super.clone();
    copy.proxyCount = 1;
    return copy;
  }
```

Пример: Copy-On-Write Proxy

```
    // Return the number of proxies using this object.
    synchronized int getProxyCount() {
      return proxyCount;
    }

    // Increment the number of proxies using this object by one.
    synchronized void addProxy() {
      proxyCount++;
    }

    // Decrement the number of proxies using this object by one.
    synchronized void removeProxy() {
      proxyCount--;
    }
  }

}
```

Край: Шаблон Пълномощно

ЛЕКЦИОНЕН КУРС: ШАБЛОНИ ЗА ПРОЕКТИРАНЕ