Proxy Pattern

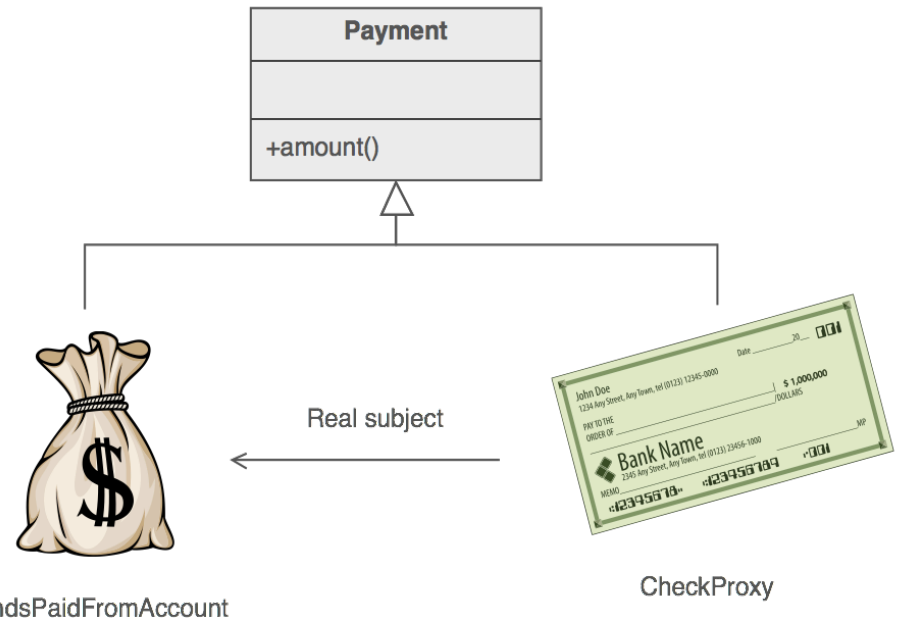
CS356 Object-Oriented Design and Programming
http://cs356.yusun.io
November 12, 2014

Yu Sun, Ph.D.
http://yusun.io
yusun@csupomona.edu

CAL POLY POMONA

Proxy

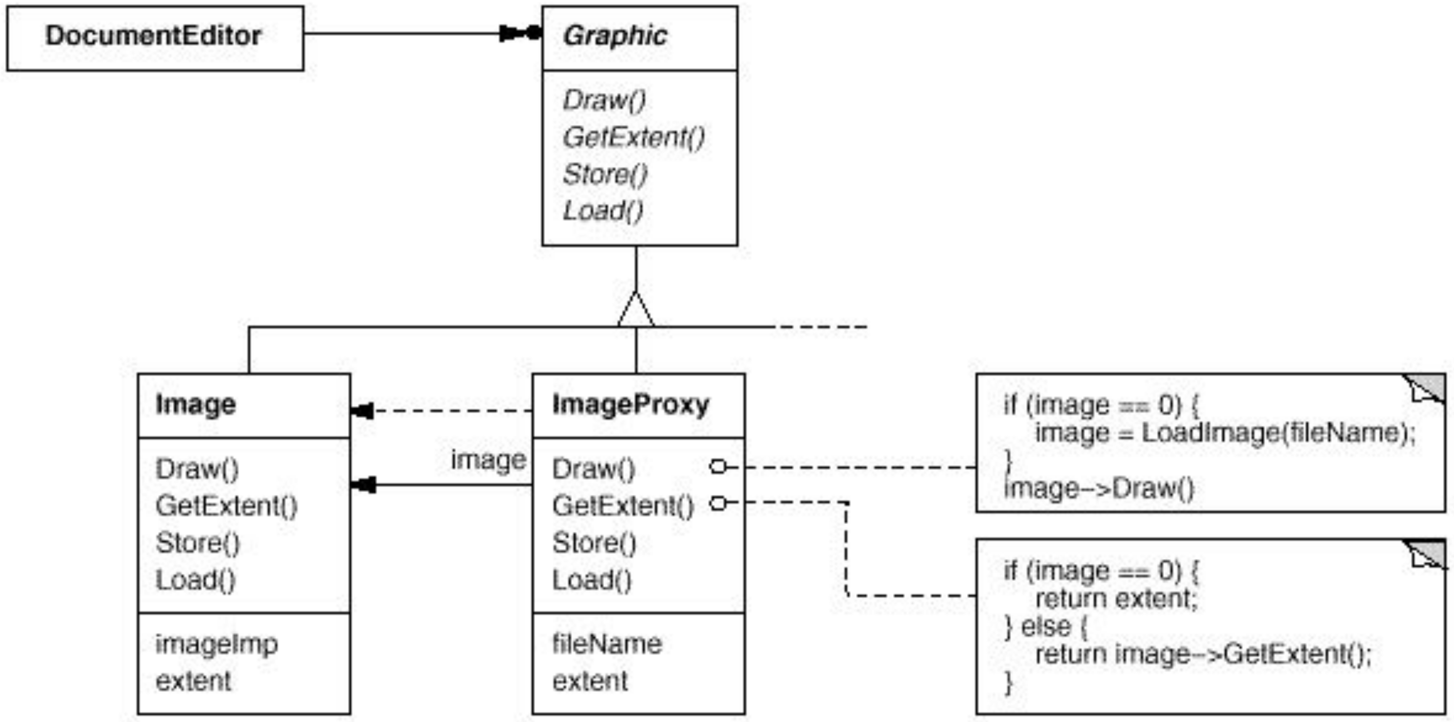
- *Intent*
  - Provide a surrogate or placeholder for another object to control access to it
- *Also Known As* – Surrogate



Loading "Heavy" Objects

- Document Editor that can embed multimedia objects
  - MMobjects are expensive to create → opening of document slow
  - Avoid creating expensive objects
    - They are not all necessary as they are not all visible at the same time
- Creating each expensive object on demand!
  - i.e., when image has to be displayed
- What should we refer to instead of actual object?
  - Hide the fact that we are "lazy"!
  - Don't complicate the document editor!

Idea: Use a Placeholder!



```
if (image == 0) {
    image = LoadImage(fileName);
}
image->Draw()
```

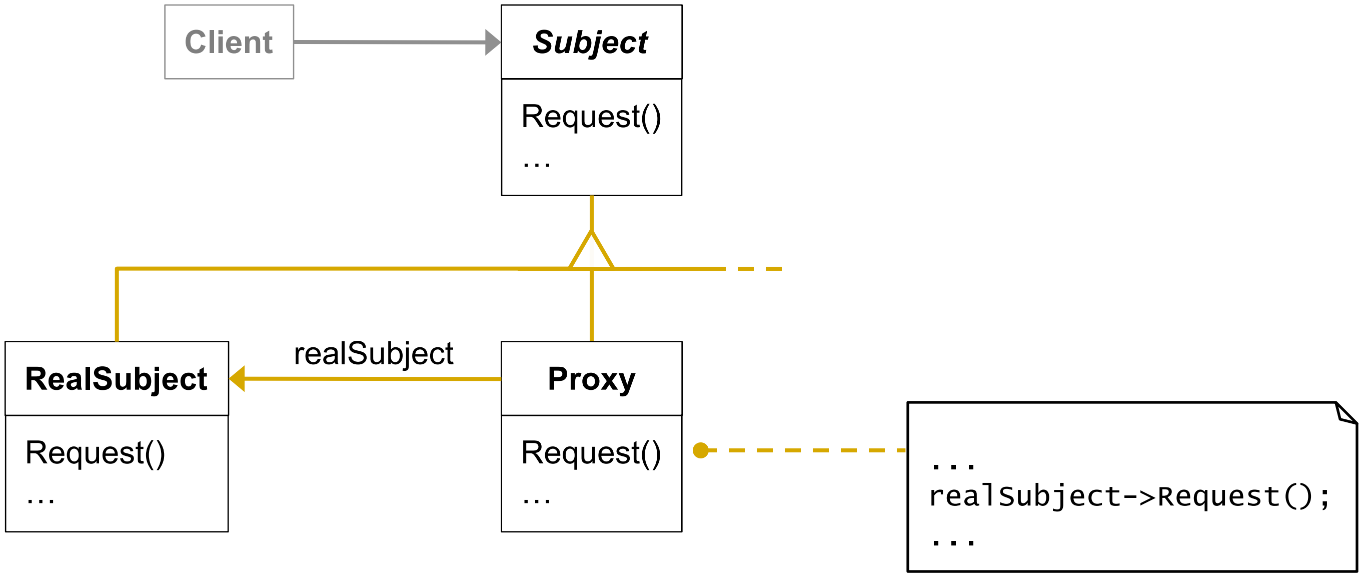
```
if (image == 0) {
    return extent;
} else {
    return image->GetExtent();
}
```

- Create only when needed for drawing
- Keeps information about the dimensions (extent)

Proxy

- *Applicability*
  - Whenever there is a need for a more sophisticated reference to an object than a simple pointer or simple reference
    - *Remote proxy* – reference an object in a different address space on the same or different machine
    - *Virtual proxy* – creation of a memory intensive object on demand (only until it is really needed)
    - *Protection proxy* – provides different clients with different levels of access to a target object
    - *Smart Reference Proxy* – additional actions whenever a target object is referenced such as counting the number of references to the object

Structure

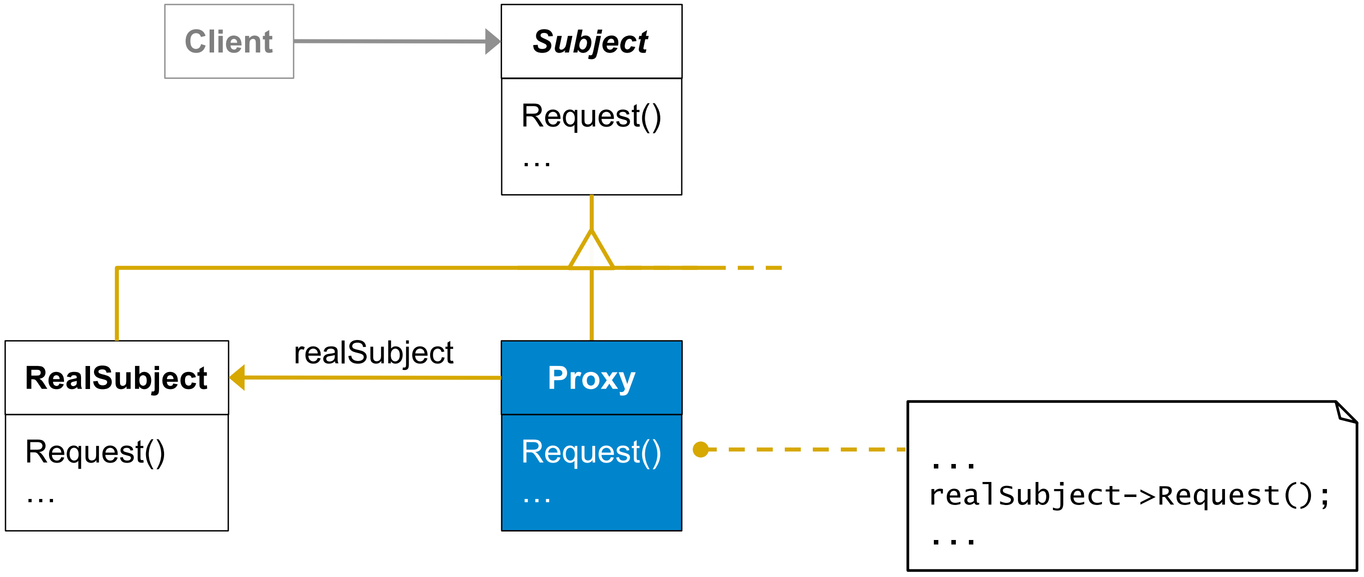


```
...
realSubject->Request();
...
```

- Proxy “stands in” for target object
- Proxy exhibits same interface as target object
  - Forwards method invocations it receives to the target object

Proxy

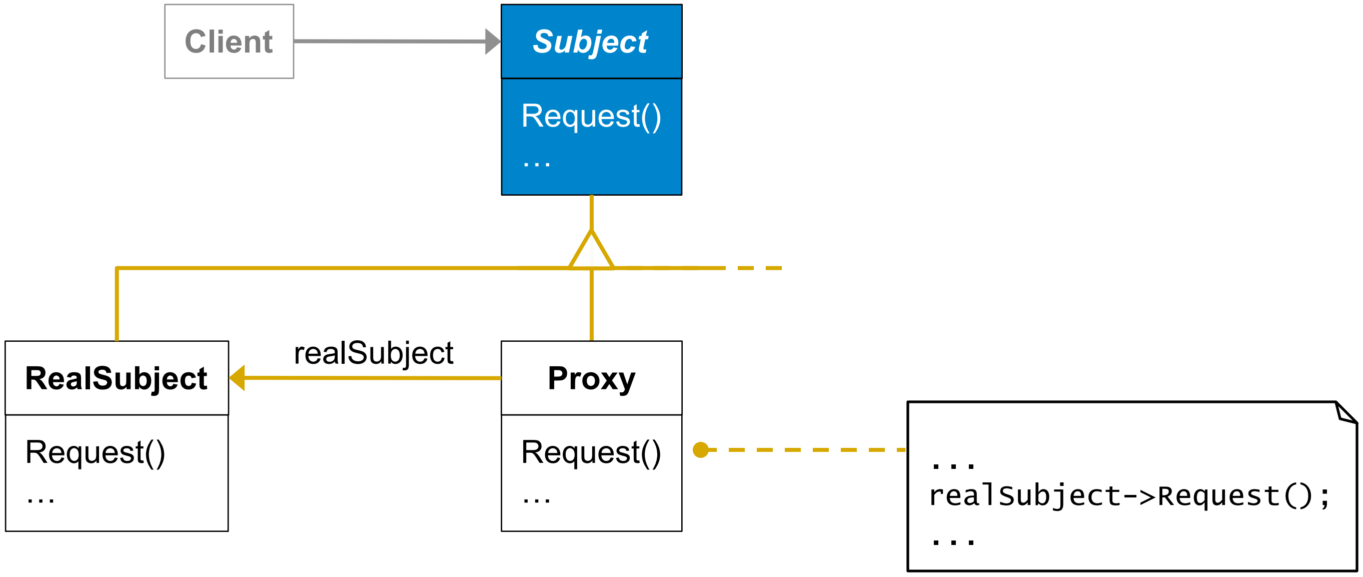
- Maintains a reference that lets the proxy access the real subject
- Provides an interface identical to Subject's
  - So that proxy can be substituted for the real subject
- Controls access to the real subject
  - May be responsible for creating or deleting it



```
...
realSubject->Request();
...
```

Subject

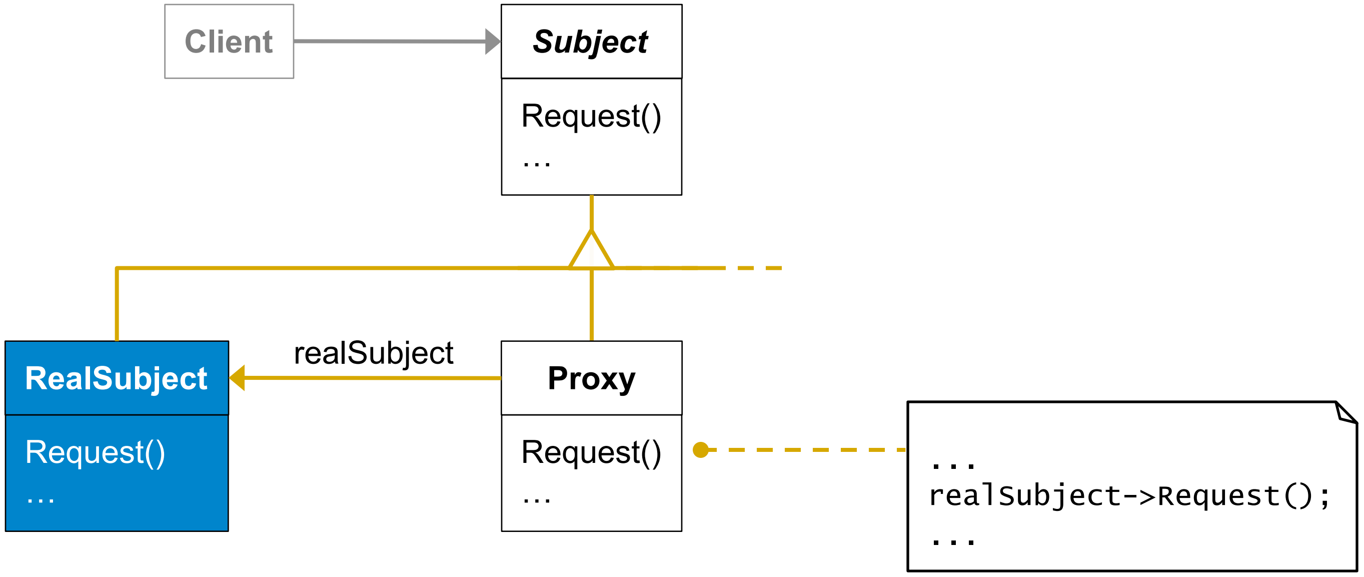
- Defines the common interface for RealSubject and Proxy



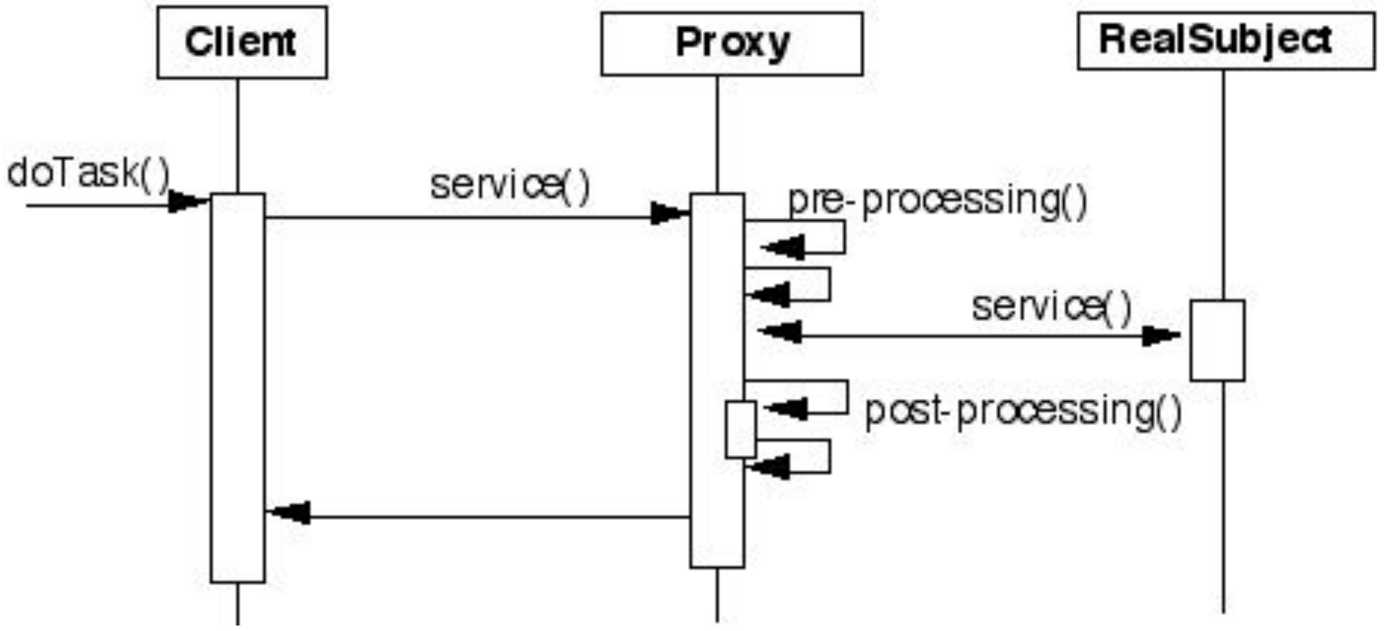
```
...
realSubject->Request();
...
```

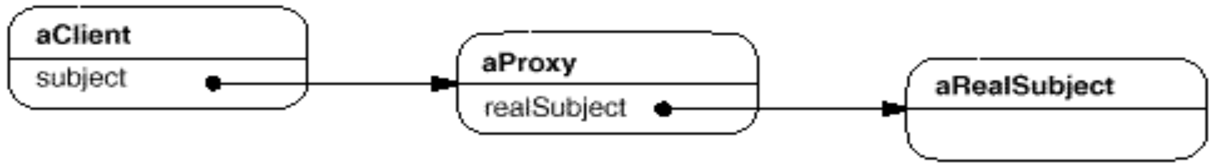
Structure

- Defines the real object that the proxy holds place for

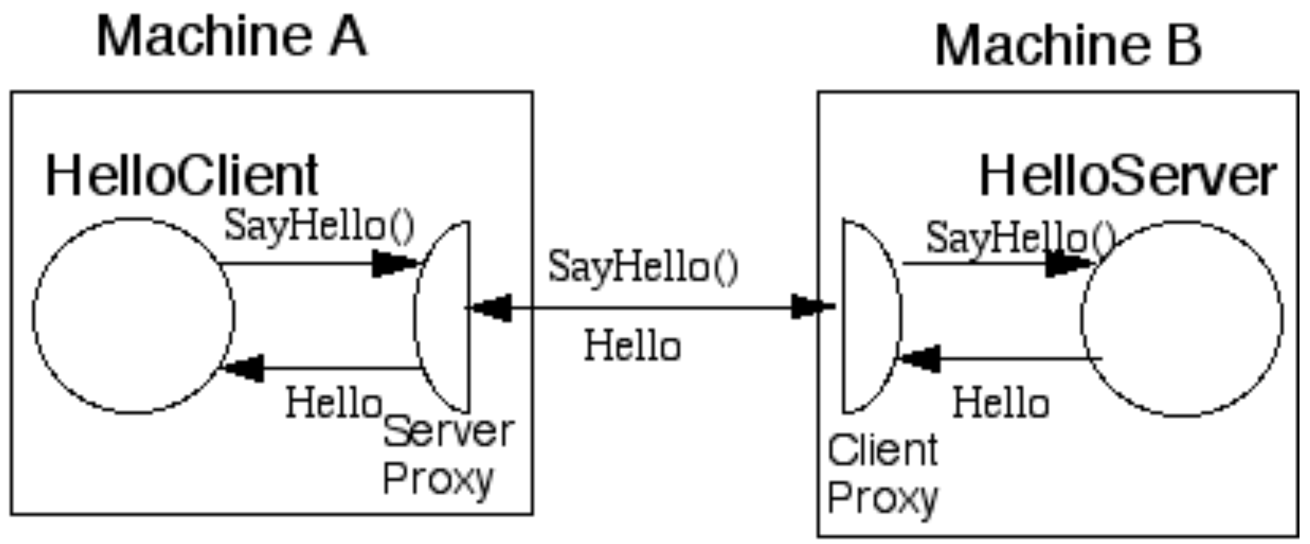


Collaborations





Remote Proxy



- Hide real details of accessing an object
  - Actual object is on a remote machine (remote address space)
  - Used in RMI and CORBA

Synchronization/Protection Proxy

- Synchronize multiple accesses to real subject

```
public class Table {
  public Object elementAt(int row, int column){ blah }
  public void setElementAt(Object element,int row,int col)
  { blah}
}

public class RowLockTable {
   Table realTable;
   Integer[] locks;

   public RowLockTable( Table toLock) {
      realTable = toLock;
      locks = new String[ toLock.numberOfRows() ];
      for (int row = 0; row< toLock.numberOfRows(); row++ )
         locks[row] = new Integer(row);
   }
   public Object elementAt( int row, int column ) {
      synchronized (locks[row]) {
         return realTable.elementAt( row, column);
      }
   }
   public void setElementAt(Object element,int row,int col)
   {
      synchronized ( locks[row] )
        return realTable.setElementAt(element, row, col);
   }
}
```

Consequences

- Proxies introduce a level of indirection
  - Used differently depending on the kind of proxy
    - Remote proxy – hide different address space
    - Virtual proxy – creation on demand
    - Protection, smart pointers – allow additional housekeeping activities