## 代理（Proxy）

### Intent

控制对其它对象的访问。

### Class Diagram

代理有以下四类：

* 远程代理（Remote Proxy）：控制对远程对象（不同地址空间）的访问，它负责将请求及其参数进行编码，并向不同地址空间中的对象发送已经编码的请求。
* 虚拟代理（Virtual Proxy）：根据需要创建开销很大的对象，它可以缓存实体的附加信息，以便延迟对它的访问，例如在网站加载一个很大图片时，不能马上完成，可以用虚拟代理缓存图片的大小信息，然后生成一张临时图片代替原始图片。
* 保护代理（Protection Proxy）：按权限控制对象的访问，它负责检查调用者是否具有实现一个请求所必须的访问权限。
* 智能代理（Smart Reference）：取代了简单的指针，它在访问对象时执行一些附加操作：记录对象的引用次数；当第一次引用一个对象时，将它装入内存；在访问一个实际对象前，检查是否已经锁定了它，以确保其它对象不能改变它。

### Implementation

以下是一个虚拟代理的实现，模拟了图片延迟加载的情况下使用与图片大小相等的临时内容去替换原始图片，直到图片加载完成才将图片显示出来。

public interface Image {
 void showImage();
}

public class HighResolutionImage implements Image {

 private URL imageURL;
 private long startTime;
 private int height;
 private int width;

 public int getHeight() {
 return height;
 }

 public int getWidth() {
 return width;
 }

 public HighResolutionImage(URL imageURL) {
 this.imageURL = imageURL;
 this.startTime = System.currentTimeMillis();
 this.width = 600;
 this.height = 600;
 }

 public boolean isLoad() {
 // 模拟图片加载，延迟 3s 加载完成
 long endTime = System.currentTimeMillis();
 return endTime - startTime > 3000;
 }

 @Override
 public void showImage() {
 System.out.println("Real Image: " + imageURL);
 }
}

public class ImageProxy implements Image {

 private HighResolutionImage highResolutionImage;

 public ImageProxy(HighResolutionImage highResolutionImage) {
 this.highResolutionImage = highResolutionImage;
 }

 @Override
 public void showImage() {
 while (!highResolutionImage.isLoad()) {
 try {
 System.out.println("Temp Image: " + highResolutionImage.getWidth() + " " + highResolutionImage.getHeight());
 Thread.sleep(100);
 } catch (InterruptedException e) {
 e.printStackTrace();
 }
 }
 highResolutionImage.showImage();
 }
}

public class ImageViewer {

 public static void main(String[] args) throws Exception {
 String image = "http://image.jpg";
 URL url = new URL(image);
 HighResolutionImage highResolutionImage = new HighResolutionImage(url);
 ImageProxy imageProxy = new ImageProxy(highResolutionImage);
 imageProxy.showImage();
 }
}

### JDK

* java.lang.reflect.Proxy
* RMI