프록시 패턴

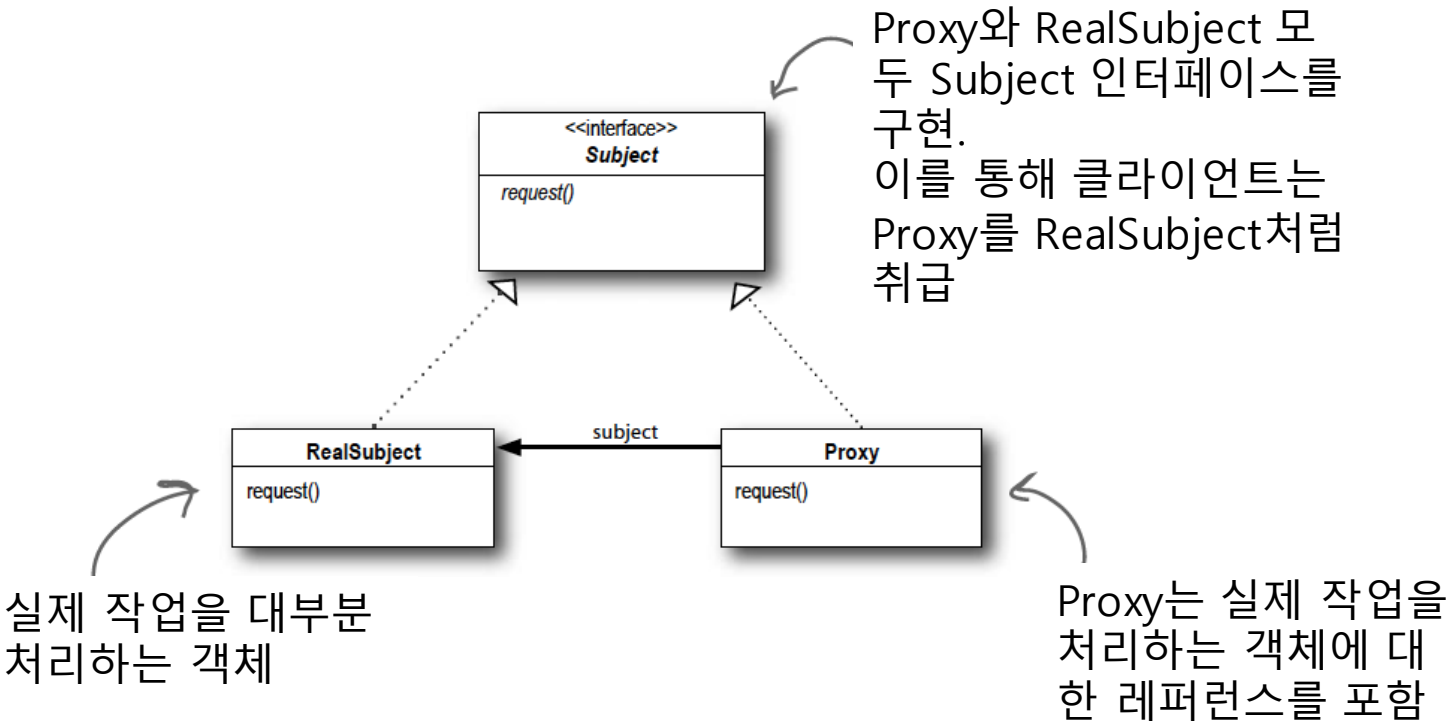
이관우
kwlee@hansung.ac.kr

학습 목표

- 원격 프록시에 대한 이해
- 가상 프록시에 대한 이해
- 보호 프록시에 대한 이해

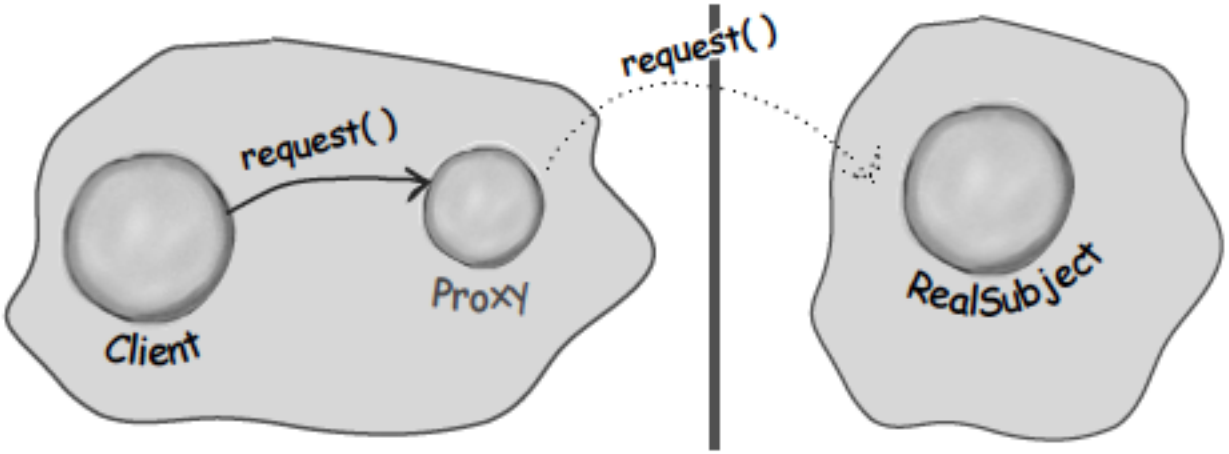
프록시 (Proxy) 패턴

- 정의
  - 어떤 객체에 대한 접근을 제어하기 위한 용도로 대리인이나 대변인에 해당하는 객체를 제공하는 패턴



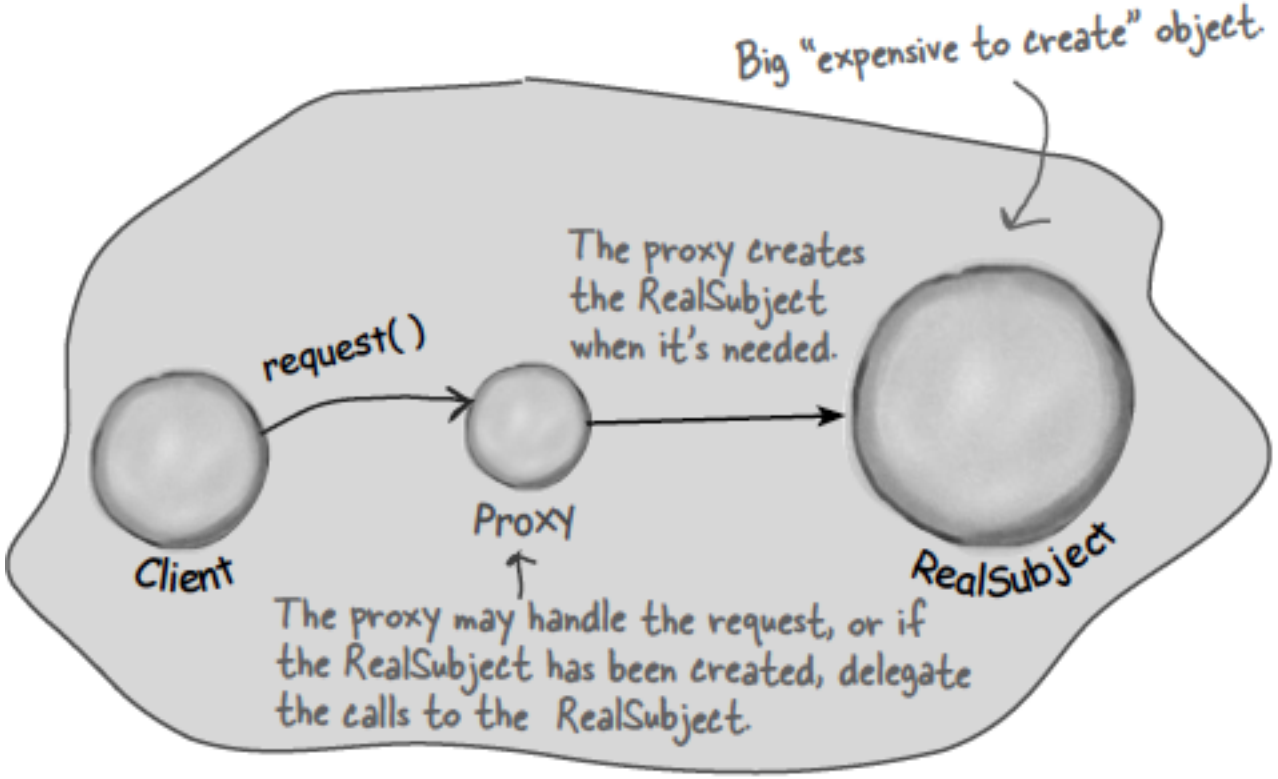
원격 프록시

- 원격 프록시는 다른 주소 공간에 존재하는 객체에 대한 로컬 대변인 역할을 수행



가상 프록시

- 가상 프록시는 생성하는 데 많은 비용이 드는 객체를 대신하는 역할

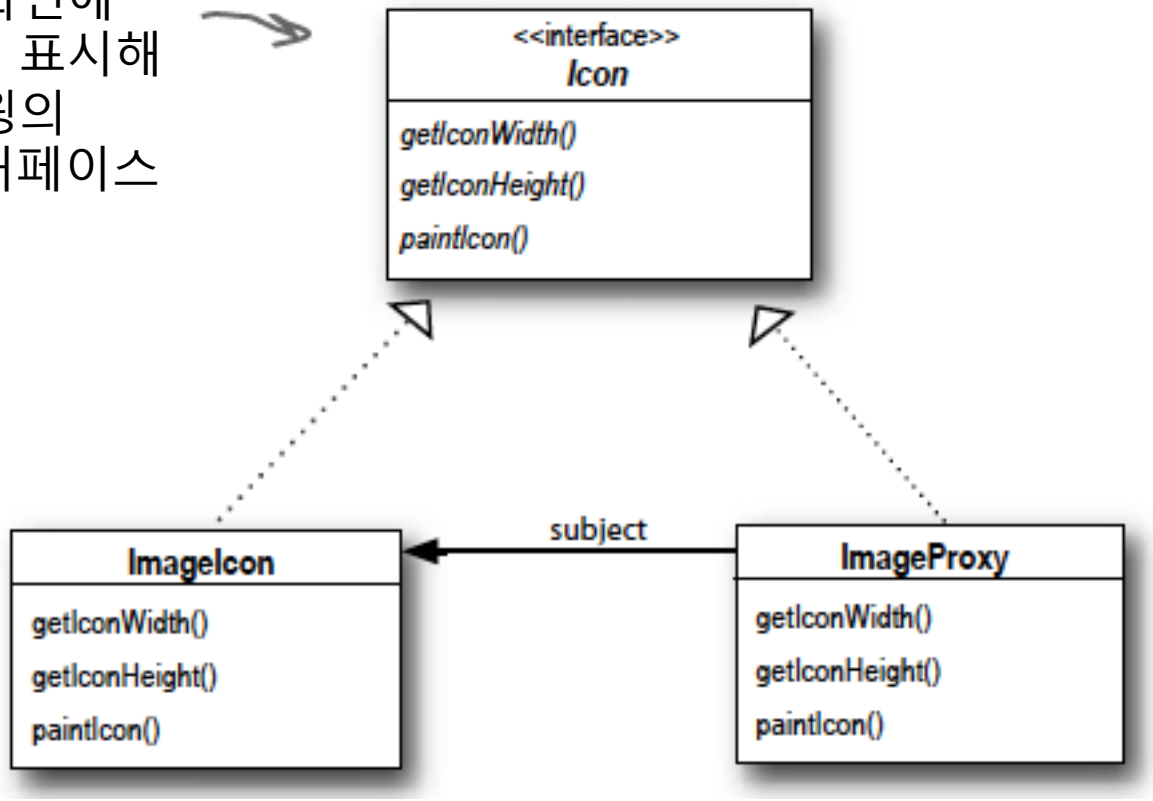


CD 커버 표시하기

- CD 타이틀 메뉴를 만든 다음 이미지를 온라인 서비스로부터 가져옴
  - 스윙의 Icon객체를 사용하여, Icon객체가 네트워크를 통해 이미지를 불러옴
  - 네트워크의 상태나 인터넷 연결 속도에 따라 CD 커버 이미지 로딩이 시간이 걸릴 수 있으므로, 이미지 로딩동안 화면에 다른 메시지를 표시하고 어플리케이션이 전체 동작을 멈추지 않도록 함
- 가상 프록시가 Icon 객체 대신에 백그라운드에서 이미지를 로딩하는 작업을 처리
  - 이미지가 완전이 로딩되기 전에는 "Loading CD cover, please wait.." 메시지를 표시
  - 이미지 로딩이 끝나면 프록시에서는 Icon 객체한테 모든 작업을 맡김

CD 커버 가상 프록시 설계

사용자 화면에
이미지를 표시해
주는 스윙의
Icon 인터페이스



이미지를 화면에 표시해주는
javax.swing.ImageIcon 클래스

이미지 로딩 전에는 간단한 메시지 표시, 이미지 로딩 후에는 ImageIcon에 이미지 표시 작업 위임

ImageProxy 코드

```
class ImageProxy implements Icon {
        ImageIcon imageIcon;
        final URL imageURL;
        Thread retrievalThread;
        boolean retrieving = false;

        public ImageProxy(URL url) { imageURL = url; }

        public int getIconWidth() {
                if (imageIcon != null) {
                        return imageIcon.getIconWidth();
                } else
                        return 800;
        }

        public int getIconHeight() {
                if (imageIcon != null) {
                        return imageIcon.getIconHeight();
                } else
                        return 600;
        }
```

Icon 인터페이스 구현

실제로 표시될 이미지를 가진 Icon

이미지의 URL을 생성자 파라미터로 넘김

imageIcon 로딩전에는 디폴트 값 반환, 이후에는 IconImage 객체 사용

ImageProxy 코드

화면에 Icon을 그릴 때 호출되는 메소드

```
public void paintIcon(final Component c, Graphics  g, int x,  int y) {
        if (imageIcon != null) {
                imageIcon.paintIcon(c,  g, x, y);
        } else {
                g.drawString("Loading  CD cover, please  wait...", x+300, y+190);
                if (!retrieving) {
                        retrieving  =  true;

                        retrievalThread  =  new Thread(new Runnable() {
                                public void run() {
                                        try {
                                                setImageIcon(new ImageIcon(imageURL, "CD Cover"));
                                                c.repaint();
                                        } catch (Exception e) {
                                                e.printStackTrace();
                                        }
                                }
                        });
                        retrievalThread.start();
                }
        }
}
```

ImageIcon이 준비되었으며, 자체 그리기

그렇지 않으면, "Loading..." 메시지 출력

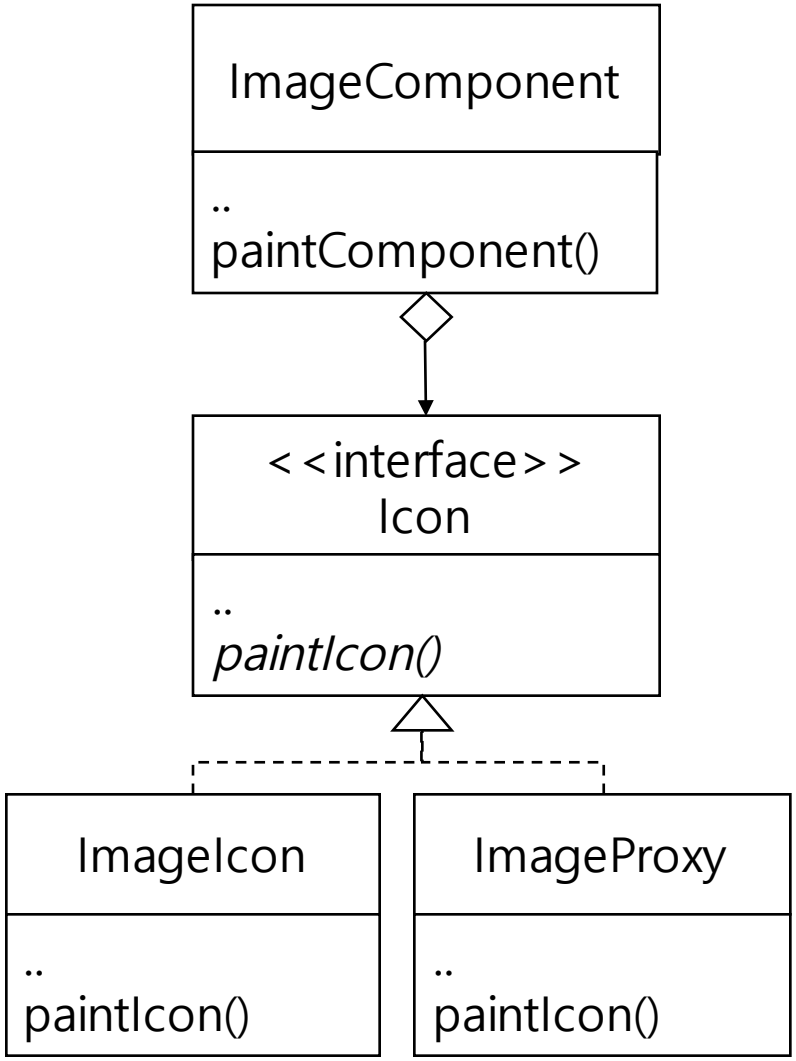
이미지 로딩 중이 아니라면..

이미지 로딩이 끝나기 전에는 ImageIcon 생성자에서 어떠한 것도 반환하지 않음

ImageComponent 코드

```
class ImageComponent extends JComponent {
    private static final long serialVersionUID = 1L;
    private Icon icon;

    public ImageComponent(Icon icon) {
        this.icon = icon;
    }

    public void setIcon(Icon icon) {
        this.icon = icon;
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        int w = icon.getIconWidth();
        int h = icon.getIconHeight();
        int x = (800 - w)/2;
        int y = (600 - h)/2;
        icon.paintIcon(this, g, x, y);
    }
}
```

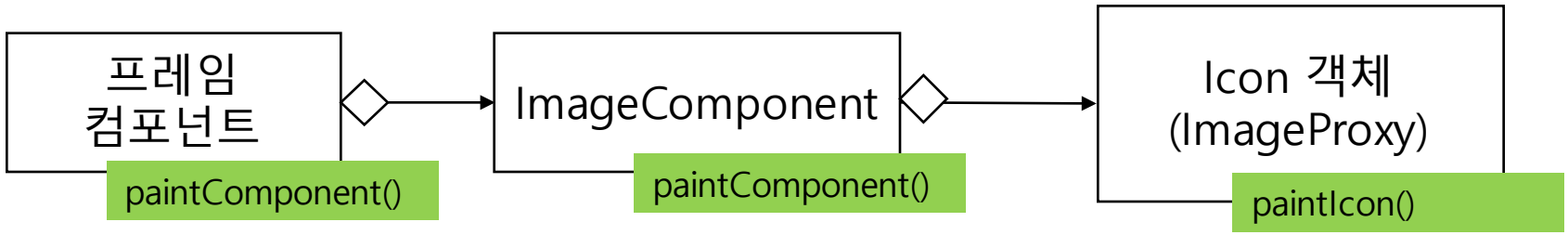
Icon 객체의 폭과 높이를 구해서, paintIcon() 메소드 호출



CD 커버 표시하기 테스트

```
public class ImageProxyTestDrive {
        ImageComponent imageComponent;
        ...
        public static void main (String[] args) throws Exception {
                ...
                // set up frame and menus

                Icon icon = new ImageProxy(initialURL);
                imageComponent = new ImageComponent(icon);
                frame.getContentPane().add(imageComponent);
                ...
        }
```

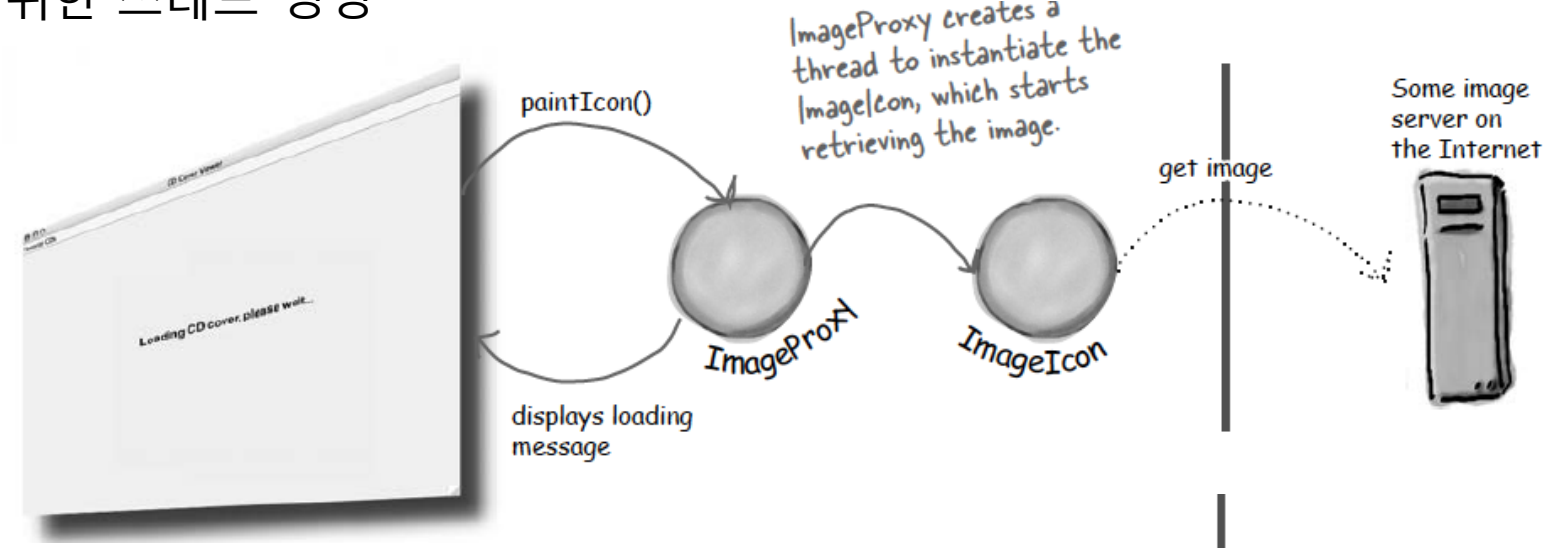
프레임에 추가할 수 있도록 프록시를 ImageComponent로 감쌈.



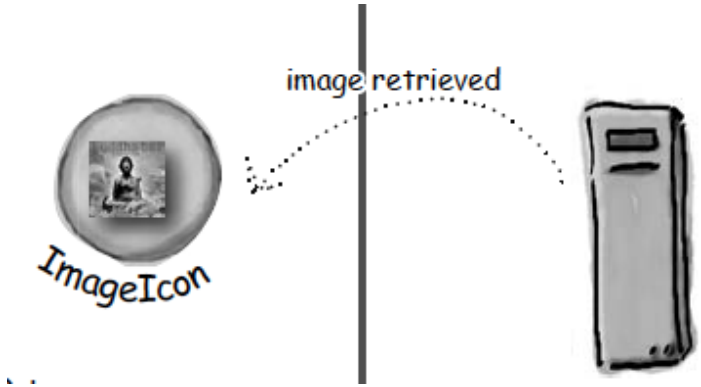
https://github.com/kwanulee/PatternExample/tree/master/proxy/virtualproxy

정리

1. 화면에 이미지를 표시하기 위한 ImageProxy 생성, paintIcon() 메소드가 호출되면, 이미지를 가져오는 ImageIcon을 생성하기 위한 쓰레드 생성

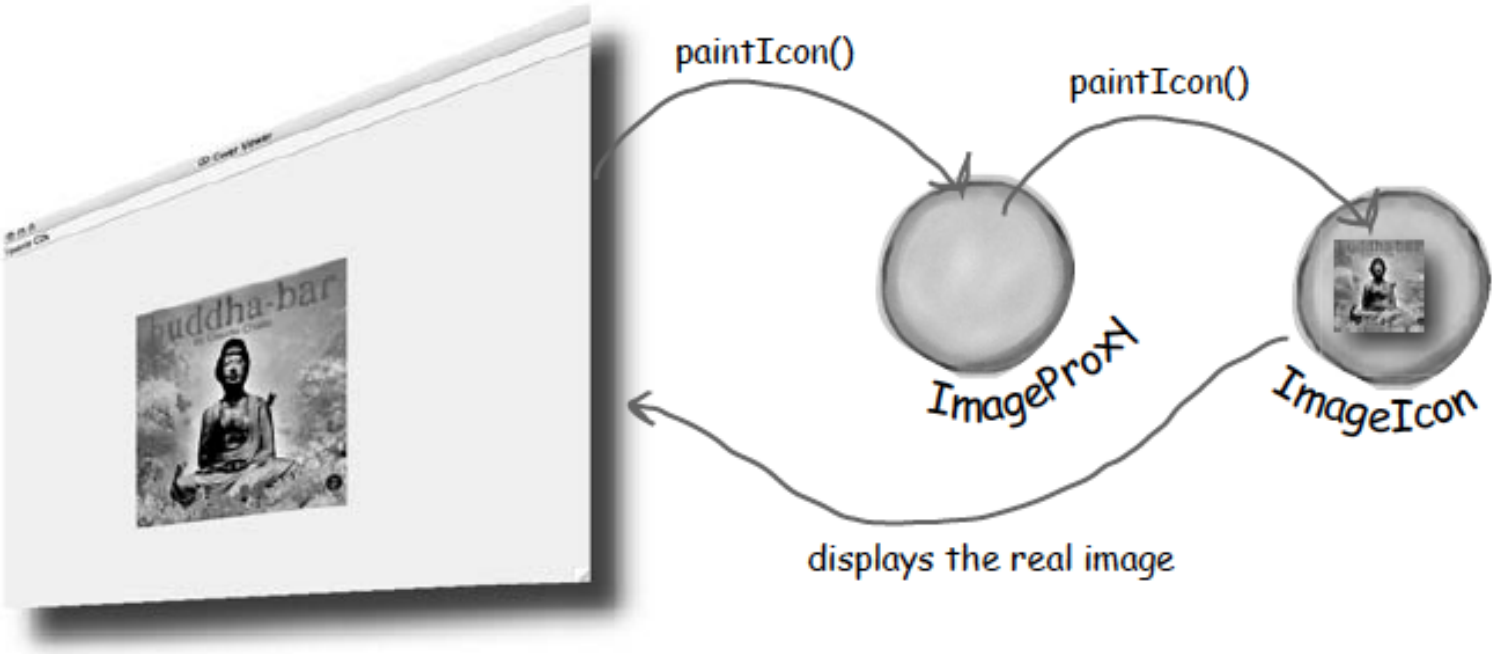


2. 얼마 후에 이미지가 리턴되고, ImageIcon 인스턴스를 만드는 작업이 완료



정리

3. ImageIcon 생성이 완료된 후에 paintIcon()이 호출되면, 프록시에서 그 호출을 곧바로 ImageIcon 객체한테 넘김



결혼 정보 시스템

- 고객 정보를 검색하고, 서로 상대방을 평가할 수 있는 기능

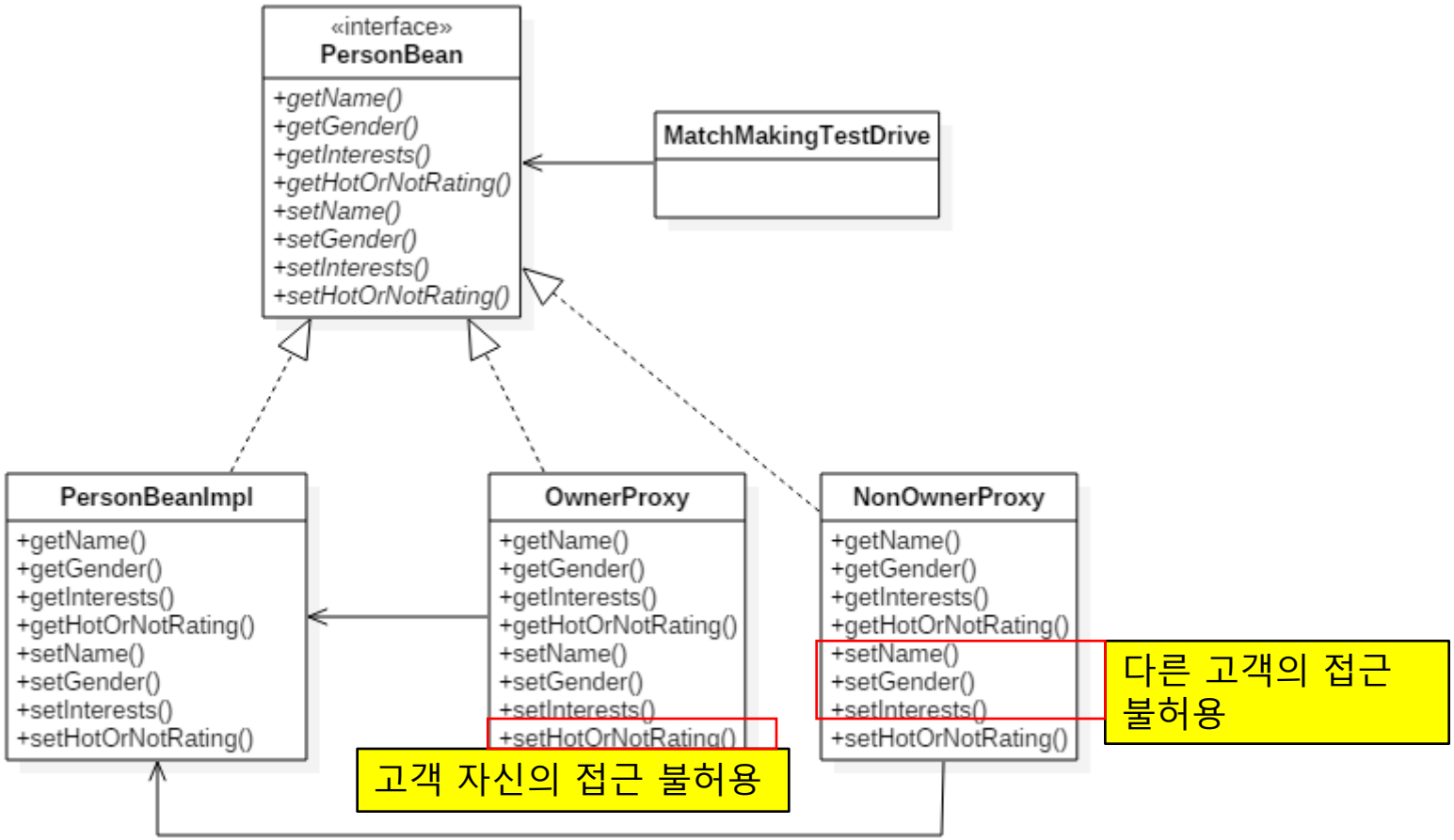
```
public interface PersonBean {

        String getName();
        String getGender();
        String getInterests();
        int getHotOrNotRating();

        void setName(String name);
        void setGender(String gender);
        void setInterests(String interests);
        void setHotOrNotRating(int rating);

}
```

결혼 정보 시스템 보호기능

- 고객이 자신의 정보 (이름, 성, 취미)를 바꾸는 것은 허용
- 다른 고객의 정보 (이름, 성, 취미)를 바꾸는 것을 불허용
- 고객이 자신의 점수를 매기는 것은 불허용
- 다른 고객의 점수를 매기는 것은 허용

- 보호 프록시 (Protection Proxy)를 사용
  - 접근 권한을 바탕으로 객체에 대한 접근을 제어하는 프록시
  - 고객 자신의 접근 권한과 다른 고객의 접근 권한을 구분

결혼 정보 시스템 보호기능



OwnerProxy 코드

```
public class OwnerProxy implements PersonBean {
    private PersonBean person;

    public OwnerProxy(PersonBean person){
        this.person = person;
    }

    @Override
    public String getName() {
        return person.getName();
    }

    // 다른 메소드 구현은 모두 person객체의 동일 메소드 호출로 구현

    @Override
    public void setHotOrNotRating(int rating) throws IllegalAccessException {
        throw new IllegalAccessException();
    }
}
```

https://github.com/kwanulee/PatternExample/tree/master/proxy/protectionProxy

테스트 코드

```
public class MatchMakingTestDrive {
        HashMap<String, PersonBean> datingDB = new HashMap<String, PersonBean>();

        public static void main(String[] args) {
                MatchMakingTestDrive test = new MatchMakingTestDrive();
                test.drive();
        }
        ...

        public void drive() {
                PersonBean joe = getPersonFromDatabase("Joe Javabean");
                PersonBean ownerProxy = getOwnerProxy(joe);
                System.out.println("Name is " + ownerProxy.getName());
                ownerProxy.setInterests("bowling, Go");
                System.out.println("Interests set from owner proxy");
                try {
                        ownerProxy.setHotOrNotRating(10);
                } catch (Exception e) {
                        System.out.println("Can't set rating from owner proxy");
                }
                System.out.println("Rating is " + ownerProxy.getHotOrNotRating());

                ...
        }
```

https://github.com/kwanulee/PatternExample/tree/master/proxy/protectionProxy

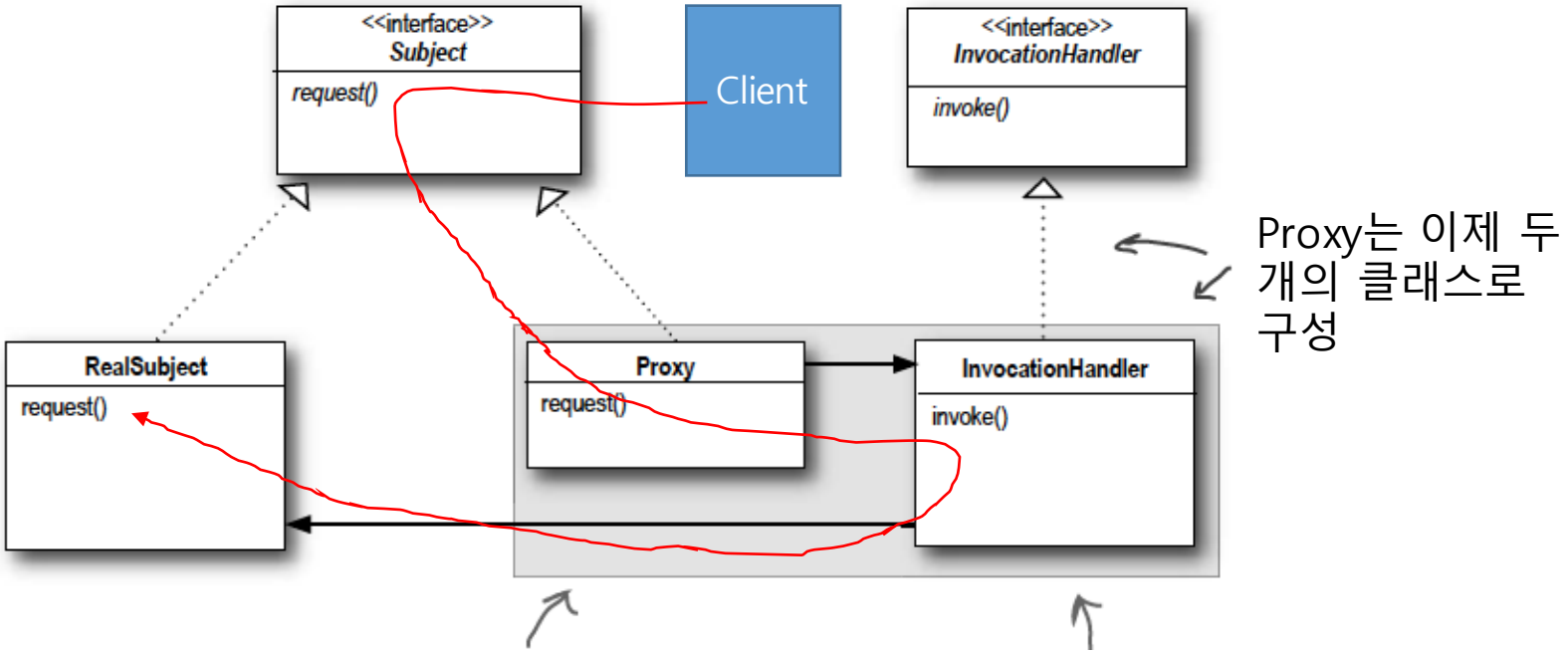
결혼 정보 시스템 보호기능 설계의 분석

- 현 설계의 단점
  - Proxy의 보호기능이 들어가지 않는 다른 메소드들도 모두 구현해 줘야 함
  - PersonBean의 인터페이스가 변경된다면, 이를 구현한 모든 클래스 (OwnerProxy, NonOwnerProxy 등)가 같이 변경이 되어야 함.

- 단점 해결을 위한 방안?
  - Proxy의 역할인 보호기능이 필요한 메소드만 프록시에 구현하고, 나머지 메소드들을 자동완성해준다면 ..
    - Java의 동적 프록시 기능 사용

Java API를 이용해서 보호 프록시 생성

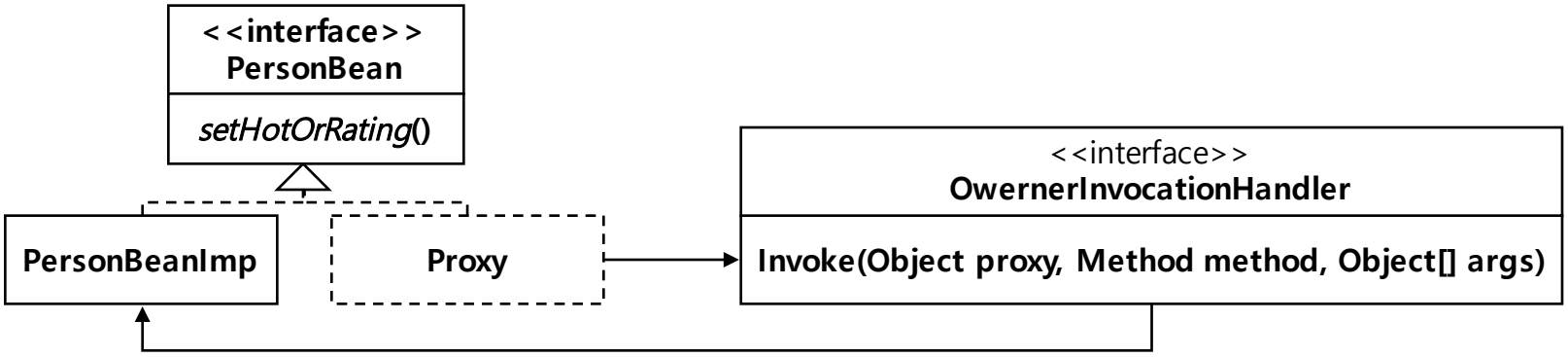
- 자바에는 프록시 기능이 java.lang.reflect 패키지에 있음
- 동적 프록시 (Dynamic Proxy)
  - Subject 인터페이스를 구현하고 메소드 호출을 사용자가 지정한 클래스(RealSubject)의 메소드 호출로 전환하는 **프록시 클래스를 자바가 동적으로 생성**



PersonBean 용 동적 프록시 만들기

- 첫번째 단계: InvocationHandler 만들기
  - 프록시의 메소드가 호출되었을 때, 어떠한 메소드의 접근을 제한할지를 결정하는 핸들러 작성
    - OwnerInvationHandler
    - NonOwnerInvationHandler
- 두번째 단계: 동적 프록시 생성
  - java.lang.reflect 패키지를 이용하여, 프록시 클래스를 동적으로 생성하고 그 인스턴스를 만듬
- 세번째 단계: PersonBean 객체를 적절한 프록시로 감쌈
  - PersonBean 객체를 소유자가 사용하고자 한다면 OwnerInvationHandler로, 비소유자가 사용하고자 한다면 NonOwnerInvationHandler로 감싼다.

첫번째 단계: InvocationHandler 만들기



1 `proxy.setHotOrNotRating(9);`

2 `invoke(Object proxy, Method method, Object[] args)`

주어진 요청을 어떻게 처리할지를 결정한 다음,
RealSubject (person)에 요청을 전달

3 `return method.invoke(person, args);`

OwenerInvocationHandler

InvocationHandler 구현

RealSubject 레퍼런스

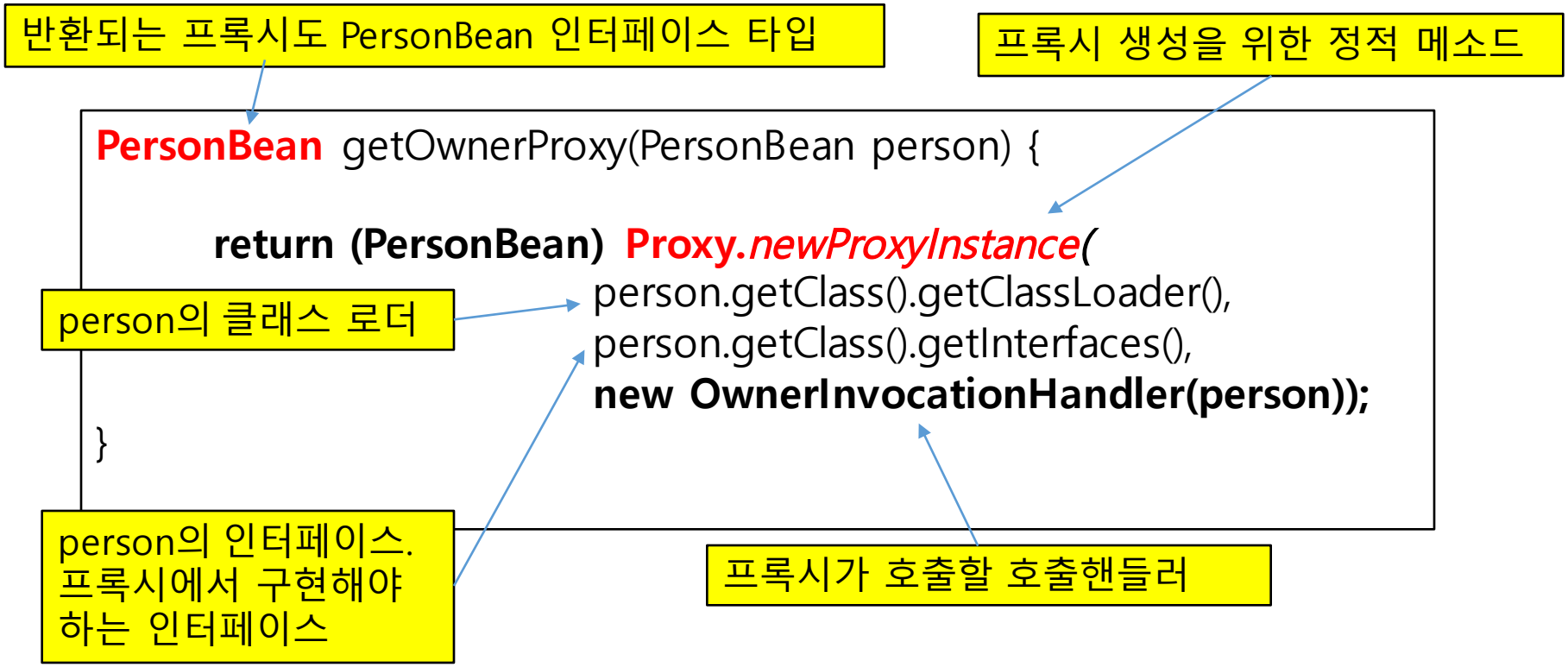
프록시의 메소드가 호출될 때마다 호출되는 메소드

주어진 요청을 어떻게 처리할지를 결정한 다음, RealSubject (person)에 요청을 전달

```
public class OwnerInvocationHandler implements InvocationHandler {
        PersonBean person;

        public OwnerInvocationHandler(PersonBean person) {
                this.person = person;
        }

        public Object invoke(Object proxy, Method method, Object[] args)
                                                        throws IllegalAccessException {

                try {
                        if (method.getName().startsWith("get")) {
                                return method.invoke(person, args);
                        } else if (method.getName().equals("setHotOrNotRating")) {
                                throw new IllegalAccessException();
                        } else if (method.getName().startsWith("set")) {
                                return method.invoke(person, args);
                        }
                } catch (InvocationTargetException e) {
                        e.printStackTrace();
                }
                return null;
        }
}
```

두번째 단계: 동적 프록시 생성

동적으로 프록시 클래스를 생성하고 프록시 객체 인스턴스를 생성함



세번째 단계: PersonBean 객체를 적절한 프록시로 감쌈

```
public class MatchMakingTestDrive {
        HashMap<String, PersonBean> datingDB = new HashMap<String, PersonBean>();

        public static void main(String[] args) {
                MatchMakingTestDrive test = new MatchMakingTestDrive();
                test.drive();
        }
        ...

        public void drive() {
                PersonBean joe = getPersonFromDatabase("Joe Javabean");
                PersonBean ownerProxy = getOwnerProxy(joe);
                System.out.println("Name is " + ownerProxy.getName());
                ownerProxy.setInterests("bowling, Go");
                System.out.println("Interests set from owner proxy");
                try {
                        ownerProxy.setHotOrNotRating(10);
                } catch (Exception e) {
                        System.out.println("Can't set rating from owner proxy");
                }
                System.out.println("Rating is " + ownerProxy.getHotOrNotRating());

                ...
        }
```

https://github.com/kwanulee/PatternExample/tree/master/proxy/javaprotectionproxy

핵심 정리

- 프록시 패턴을 이용하면 어떤 객체에 대한 대변인을 내세워서 클라이언트의 접근을 제어할 수 있습니다.
- 원격 프록시는 클라이언트와 원격 객체 사이의 데이터 전달을 관리해 줍니다.
- 가상 프록시는 인스턴스를 만드는 데 많은 비용이 드는 객체에 대한 접근을 제어합니다.
- 보호 프록시는 호출하는 쪽의 권한에 따라서 객체에 있는 메소드에 대한 접근을 제어합니다.