Padrões de Projeto de Software Orientados a Objetos

Tecnologia em Análise e Desenvolvimento de Sistemas

Paulo Mauricio Gonçalves Júnior

Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco

4 de junho de 2018

Parte I

Proxy

Introdução I

- Vamos enviar um relatório de cada máquina de chicletes.
- Vamos colocar a localização de cada máquina na classe.

```
public class GumballMachine {
  // other instance variables
  String location;
  public GumballMachine(String location, int count) {
    // other constructor code here
    this.location = location;
  }
  public String getLocation() {
    return location;
  }
  // other methods here
}

public class GumballMonitor {
  GumballMachine machine;
  public GumballMonitor(GumballMachine machine) {
```

Introdução II

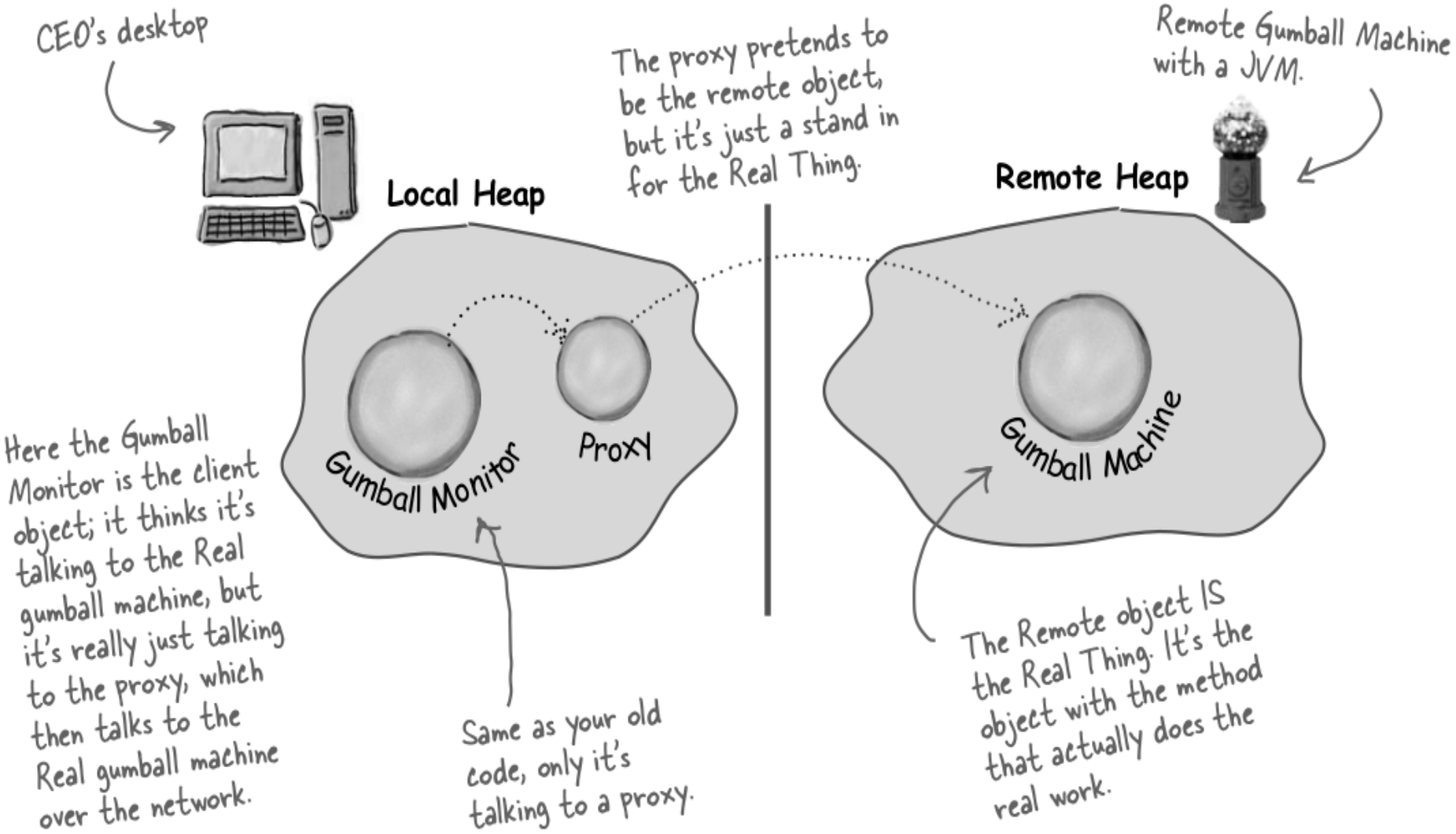
```
    this.machine = machine;
  }
  public void report() {
    System.out.println("Gumball Machine: " + machine.getLocation())
        ;
    System.out.println("Current inventory: " + machine.getCount() +
         " gumballs");
    System.out.println("Current state: " + machine.getState());
  }
}

public class GumballMachineTestDrive {
  public static void main(String[] args) {
    int count = 0;
    if (args.length < 2) {
      System.out.println("GumballMachine <name> <inventory>");
      System.exit(1);
    }
    try {
      count = Integer.parseInt(args[1]);
    } catch (Exception e) {
```

Introdução III

```
            e.printStackTrace();
            System.exit(1);
        }
        GumballMachine gumballMachine = new GumballMachine(args[0],
            count);
        GumballMonitor monitor = new GumballMonitor(gumballMachine);
        System.out.println(gumballMachine);
        // resto do código
        monitor.report();
    }
}
```

- Isso funciona localmente, mas o desejado é que funcione na rede!

Introdução IV

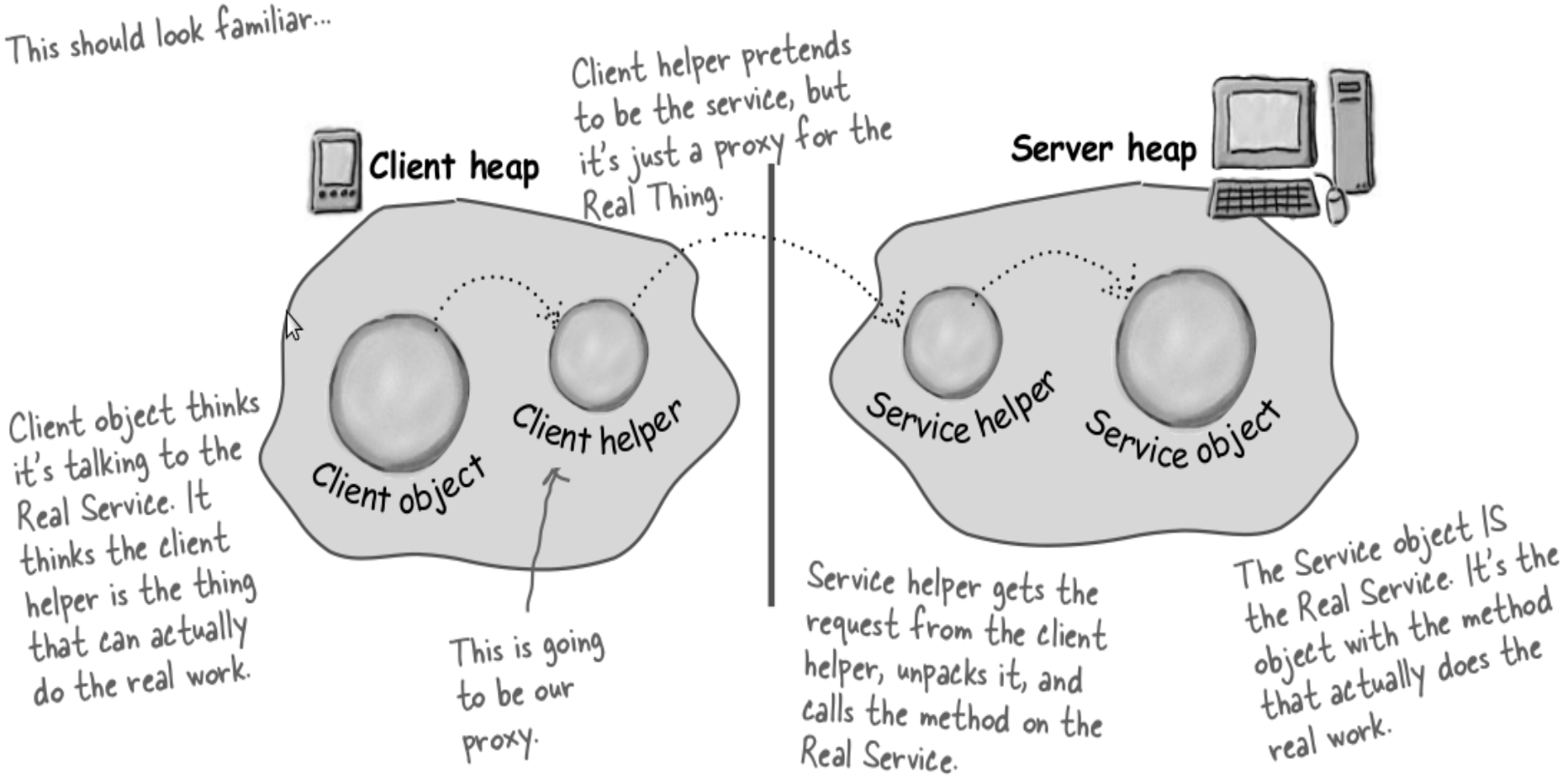


Introdução V

- Vamos usar RMI para utilizar objetos em outra JVM.

RMI I

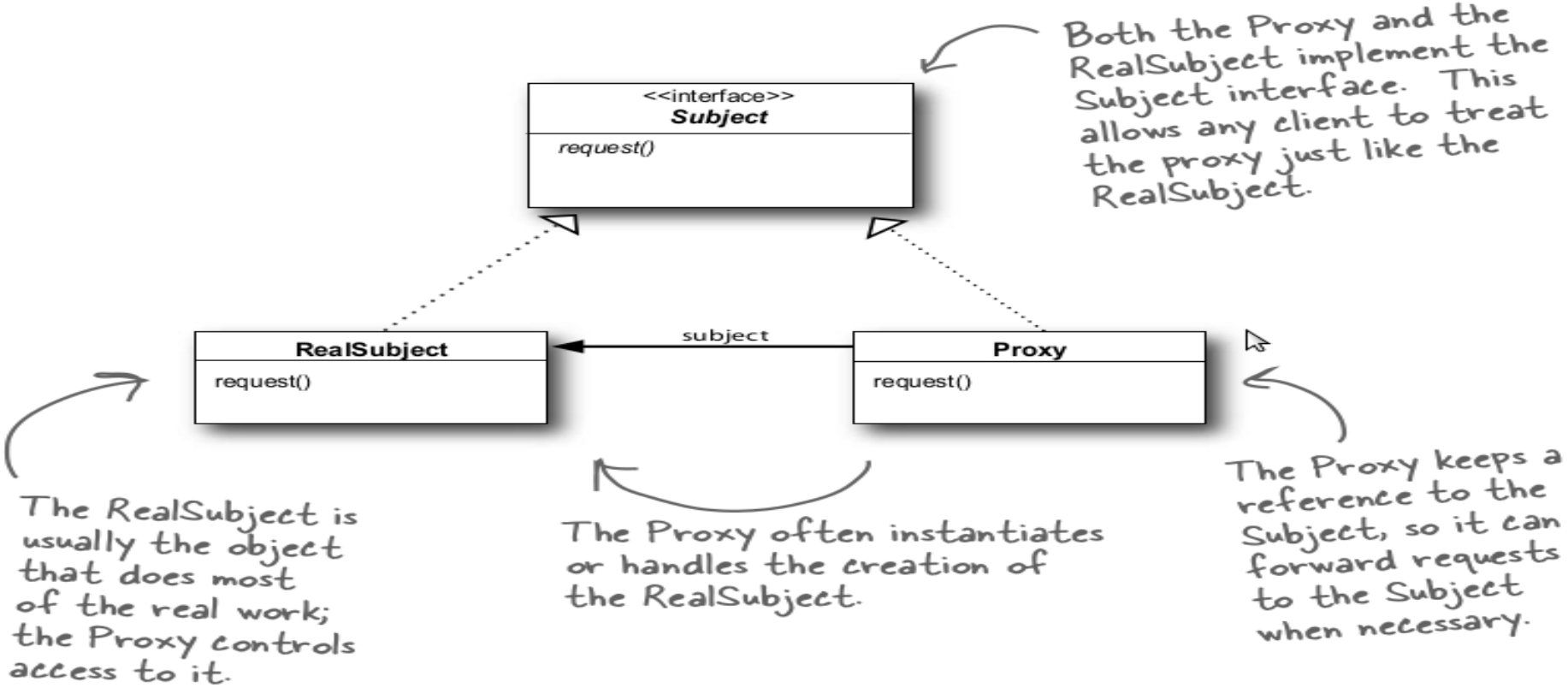
- Vamos usar classes para ajudar a realizar a comunicação pela rede.



RMI II

Definição

O padrão Proxy fornece um substituto ou espaço reservado para outro objeto para controlar o acesso a ele.

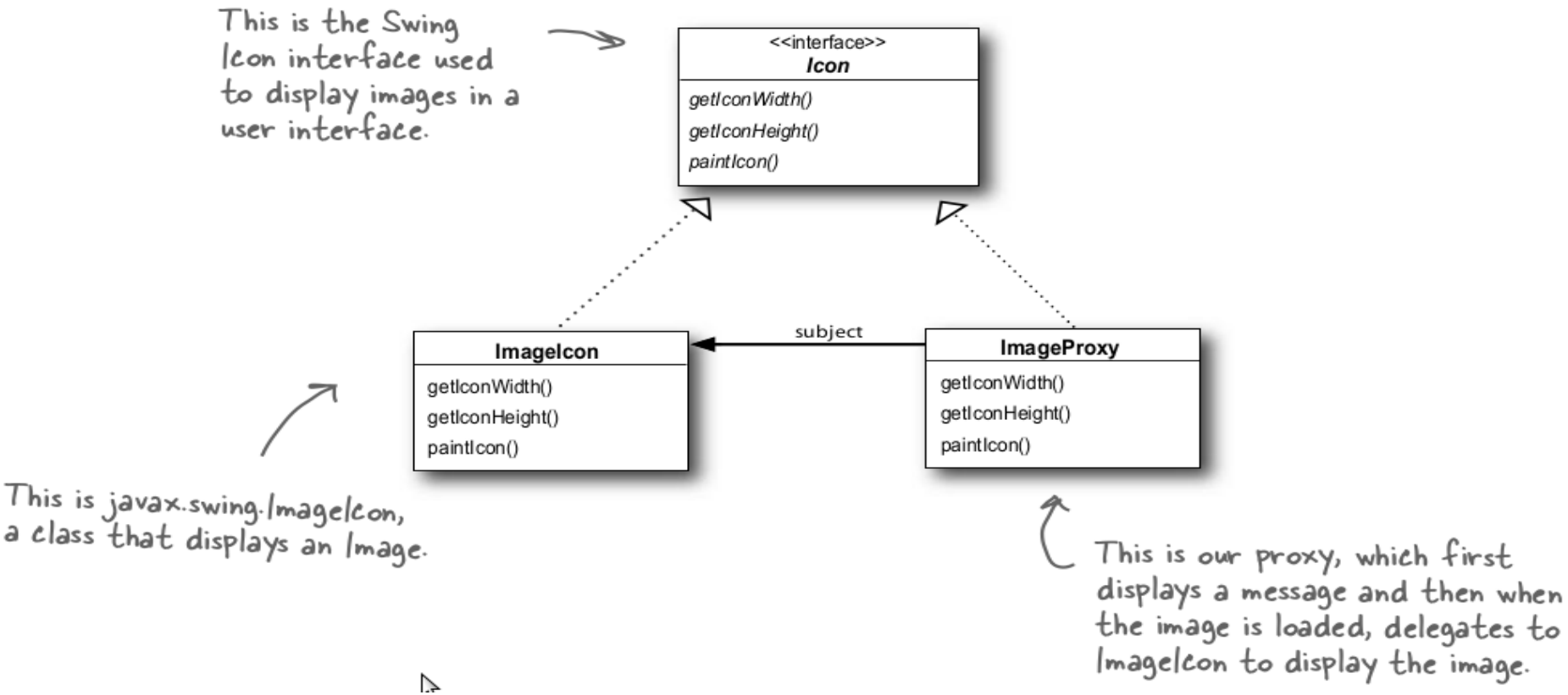


RMI III

- Existem várias formas de implementação do padrão proxy. Vejamos a implementação de outros tipos.
  - Remoto
  - Virtual
  - Proteção

Proxy Virtual I

- Mostrar a capa de álbuns obtidos pela rede e, enquanto a imagem não chega, apresentar uma mensagem.

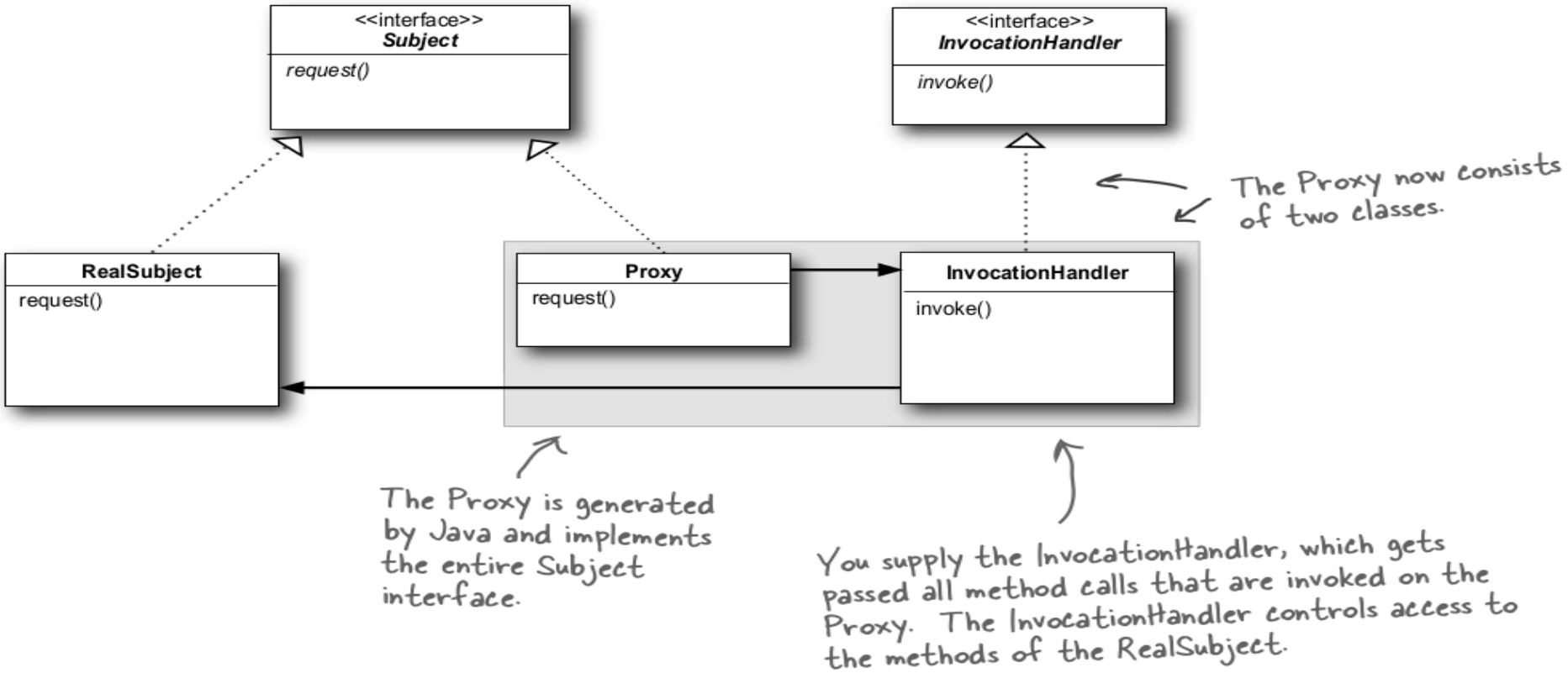


Proxy Virtual II

- ImageProxy inicialmente cria um ImageIcon e começa a carregar a partir de uma URL.
- Enquanto os bytes da imagem são obtidos, ImageProxy apresenta uma mensagem "Loading CD cover, please wait...".
- Quando a imagem for carregada, ImageProxy delega todas as chamadas de métodos para o ícone, incluindo `paintIcon()`, `getWidth()` e `getHeight()`.
- Se o usuário solicita uma nova imagem, nós criamos um novo proxy e começar o processo de novo.

Proxy de Proteção I

- Vamos usar a API Java para implementar um Proxy de Proteção (proxy dinâmico).
- Vamos implementar um sistema de encontros.



Proxy de Proteção II

```
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);
}
```

- Devemos evitar que um usuário dê notas altas a si mesmo e que outros usuários modifiquem nossos interesses.
- Vamos implementar desta forma:
  1. Criar dois InvocationHandlers
  2. Escrever o código que cria os proxies dinâmicos
  3. Encapsular qualquer objeto `PersonBean` com o proxy apropriado