Iterator in Java

Iterator is a behavioral design pattern that allows sequential traversal through a complex data structure without exposing its internal details.

Thanks to the Iterator, clients can go over elements of different collections in a similar fashion using a single iterator interface.

Learn more about Iterator →

Navigation

Popularity: ★★★

Usage examples: The pattern is very common in Java code. Many frameworks and libraries use it to provide a standard way for traversing their collections.

Here are some examples from core Java libraries:

- All implementations of `java.util.Iterator` (also `java.util.Scanner`).
- All implementations of `java.util.Enumeration`.

Identification: Iterator is easy to recognize by the navigation methods (such as `next`, `previous` and others). Client code that uses iterators might not have direct access to the collection being traversed.

Iterating over social network profiles

In this example, the Iterator pattern is used to go over social profiles of a remote social network collection without exposing any of the communication details to the client code.

iterators

iterators/ProfileIterator.java: Defines profile interface

```
package refactoring_guru.iterator.example.iterators;

import refactoring_guru.iterator.example.profile.Profile;

public interface ProfileIterator {
    boolean hasNext();

    Profile getNext();

    void reset();
}
```

iterators/FacebookIterator.java: Implements iteration over Facebook profiles

```
import refactoring_guru.iterator.example.profile.Profile;
import refactoring_guru.iterator.example.social_networks.Facebook;

import java.util.ArrayList;
import java.util.List;

public class FacebookIterator implements ProfileIterator {
    private Facebook facebook;
    private String type;
    private String email;
    private int currentPosition = 0;
    private List<String> emails = new ArrayList<>();
    private List<Profile> profiles = new ArrayList<>();

    public FacebookIterator(Facebook facebook, String type, String email) {
        this.facebook = facebook;
        this.type = type;
        this.email = email;
    }

    private void lazyLoad() {
        if (emails.size() == 0) {
            List<String> profiles = facebook.requestProfileFriendsFromFacebook(this.email
            for (String profile : profiles) {
                this.emails.add(profile);
                this.profiles.add(null);
            }
        }
    }

    @Override
    public boolean hasNext() {
        lazyLoad();
        return currentPosition < emails.size();
    }

    @Override
    public Profile getNext() {
        if (!hasNext()) {
            return null;
        }

        String friendEmail = emails.get(currentPosition);
        Profile friendProfile = profiles.get(currentPosition);
        if (friendProfile == null) {
            friendProfile = facebook.requestProfileFromFacebook(friendEmail);
            profiles.set(currentPosition, friendProfile);
        }
        currentPosition++;
```

```
    @Override
    public void reset() {
        currentPosition = 0;
    }
}
```

iterators/LinkedInIterator.java: Implements iteration over LinkedIn profiles

```
package refactoring_guru.iterator.example.iterators;

import refactoring_guru.iterator.example.profile.Profile;
import refactoring_guru.iterator.example.social_networks.LinkedIn;

import java.util.ArrayList;
import java.util.List;

public class LinkedInIterator implements ProfileIterator {
    private LinkedIn linkedIn;
    private String type;
    private String email;
    private int currentPosition = 0;
    private List<String> emails = new ArrayList<>();
    private List<Profile> contacts = new ArrayList<>();

    public LinkedInIterator(LinkedIn linkedIn, String type, String email) {
        this.linkedIn = linkedIn;
        this.type = type;
        this.email = email;
    }

    private void lazyLoad() {
        if (emails.size() == 0) {
            List<String> profiles = linkedIn.requestRelatedContactsFromLinkedInAPI(this.e
            for (String profile : profiles) {
                this.emails.add(profile);
                this.contacts.add(null);
            }
        }
    }

    @Override
    public boolean hasNext() {
        lazyLoad();
        return currentPosition < emails.size();
    }
```

```
    public Profile getNext() {
        if (!hasNext()) {
            return null;
        }

        String friendEmail = emails.get(currentPosition);
        Profile friendContact = contacts.get(currentPosition);
        if (friendContact == null) {
            friendContact = linkedIn.requestContactInfoFromLinkedInAPI(friendEmail);
            contacts.set(currentPosition, friendContact);
        }
        currentPosition++;
        return friendContact;
    }

    @Override
    public void reset() {
        currentPosition = 0;
    }
}
```

social_networks

social_networks/SocialNetwork.java: Defines common social network interface

```
package refactoring_guru.iterator.example.social_networks;

import refactoring_guru.iterator.example.iterators.ProfileIterator;

public interface SocialNetwork {
    ProfileIterator createFriendsIterator(String profileEmail);

    ProfileIterator createCoworkersIterator(String profileEmail);
}
```

social_networks/Facebook.java: Facebook

```
package refactoring_guru.iterator.example.social_networks;

import refactoring_guru.iterator.example.iterators.FacebookIterator;
import refactoring_guru.iterator.example.iterators.ProfileIterator;
import refactoring_guru.iterator.example.profile.Profile;
```

```
import java.util.List;

public class Facebook implements SocialNetwork {
    private List<Profile> profiles;

    public Facebook(List<Profile> cache) {
        if (cache != null) {
            this.profiles = cache;
        } else {
            this.profiles = new ArrayList<>();
        }
    }

    public Profile requestProfileFromFacebook(String profileEmail) {
        // Here would be a POST request to one of the Facebook API endpoints.
        // Instead, we emulates long network connection, which you would expect
        // in the real life...
        simulateNetworkLatency();
        System.out.println("Facebook: Loading profile '" + profileEmail + "' over the net

        // ...and return test data.
        return findProfile(profileEmail);
    }

    public List<String> requestProfileFriendsFromFacebook(String profileEmail, String cor
        // Here would be a POST request to one of the Facebook API endpoints.
        // Instead, we emulates long network connection, which you would expect
        // in the real life...
        simulateNetworkLatency();
        System.out.println("Facebook: Loading '" + contactType + "' list of '" + profileE

        // ...and return test data.
        Profile profile = findProfile(profileEmail);
        if (profile != null) {
            return profile.getContacts(contactType);
        }
        return null;
    }

    private Profile findProfile(String profileEmail) {
        for (Profile profile : profiles) {
            if (profile.getEmail().equals(profileEmail)) {
                return profile;
            }
        }
        return null;
    }

    private void simulateNetworkLatency() {
        try {
```

```
            ex.printStackTrace();
        }
    }

    @Override
    public ProfileIterator createFriendsIterator(String profileEmail) {
        return new FacebookIterator(this, "friends", profileEmail);
    }

    @Override
    public ProfileIterator createCoworkersIterator(String profileEmail) {
        return new FacebookIterator(this, "coworkers", profileEmail);
    }

}
```

social_networks/LinkedIn.java: LinkedIn

```
package refactoring_guru.iterator.example.social_networks;

import refactoring_guru.iterator.example.iterators.LinkedInIterator;
import refactoring_guru.iterator.example.iterators.ProfileIterator;
import refactoring_guru.iterator.example.profile.Profile;

import java.util.ArrayList;
import java.util.List;

public class LinkedIn implements SocialNetwork {
    private List<Profile> contacts;

    public LinkedIn(List<Profile> cache) {
        if (cache != null) {
            this.contacts = cache;
        } else {
            this.contacts = new ArrayList<>();
        }
    }

    public Profile requestContactInfoFromLinkedInAPI(String profileEmail) {
        // Here would be a POST request to one of the LinkedIn API endpoints.
        // Instead, we emulates long network connection, which you would expect
        // in the real life...
        simulateNetworkLatency();
        System.out.println("LinkedIn: Loading profile '" + profileEmail + "' over the net

        // ...and return test data.
```

```
    public List<String> requestRelatedContactsFromLinkedInAPI(String profileEmail, String
        // Here would be a POST request to one of the LinkedIn API endpoints.
        // Instead, we emulates long network connection, which you would expect
        // in the real life.
        simulateNetworkLatency();
        System.out.println("LinkedIn: Loading '" + contactType + "' list of '" + profileE

        // ...and return test data.
        Profile profile = findContact(profileEmail);
        if (profile != null) {
            return profile.getContacts(contactType);
        }
        return null;
    }

    private Profile findContact(String profileEmail) {
        for (Profile profile : contacts) {
            if (profile.getEmail().equals(profileEmail)) {
                return profile;
            }
        }
        return null;
    }

    private void simulateNetworkLatency() {
        try {
            Thread.sleep(2500);
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }

    @Override
    public ProfileIterator createFriendsIterator(String profileEmail) {
        return new LinkedInIterator(this, "friends", profileEmail);
    }

    @Override
    public ProfileIterator createCoworkersIterator(String profileEmail) {
        return new LinkedInIterator(this, "coworkers", profileEmail);
    }
}
```

profile

```
package refactoring_guru.iterator.example.profile;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Profile {
    private String name;
    private String email;
    private Map<String, List<String>> contacts = new HashMap<>();

    public Profile(String email, String name, String... contacts) {
        this.email = email;
        this.name = name;

        // Parse contact list from a set of "friend:email@gmail.com" pairs.
        for (String contact : contacts) {
            String[] parts = contact.split(":");
            String contactType = "friend", contactEmail;
            if (parts.length == 1) {
                contactEmail = parts[0];
            }
            else {
                contactType = parts[0];
                contactEmail = parts[1];
            }
            if (!this.contacts.containsKey(contactType)) {
                this.contacts.put(contactType, new ArrayList<>());
            }
            this.contacts.get(contactType).add(contactEmail);
        }
    }

    public String getEmail() {
        return email;
    }

    public String getName() {
        return name;
    }

    public List<String> getContacts(String contactType) {
        if (!this.contacts.containsKey(contactType)) {
            this.contacts.put(contactType, new ArrayList<>());
        }
        return contacts.get(contactType);
```

spammer

spammer/SocialSpammer.java: Message sending app

```
package refactoring_guru.iterator.example.spammer;

import refactoring_guru.iterator.example.iterators.ProfileIterator;
import refactoring_guru.iterator.example.profile.Profile;
import refactoring_guru.iterator.example.social_networks.SocialNetwork;

public class SocialSpammer {
    public SocialNetwork network;
    public ProfileIterator iterator;

    public SocialSpammer(SocialNetwork network) {
        this.network = network;
    }

    public void sendSpamToFriends(String profileEmail, String message) {
        System.out.println("\nIterating over friends...\n");
        iterator = network.createFriendsIterator(profileEmail);
        while (iterator.hasNext()) {
            Profile profile = iterator.getNext();
            sendMessage(profile.getEmail(), message);
        }
    }

    public void sendSpamToCoworkers(String profileEmail, String message) {
        System.out.println("\nIterating over coworkers...\n");
        iterator = network.createCoworkersIterator(profileEmail);
        while (iterator.hasNext()) {
            Profile profile = iterator.getNext();
            sendMessage(profile.getEmail(), message);
        }
    }

    public void sendMessage(String email, String message) {
        System.out.println("Sent message to: '" + email + "'. Message body: '" + message
    }
}
```

Demo.java: Client code

```
import refactoring_guru.iterator.example.profile.Profile;
import refactoring_guru.iterator.example.social_networks.Facebook;
import refactoring_guru.iterator.example.social_networks.LinkedIn;
import refactoring_guru.iterator.example.social_networks.SocialNetwork;
import refactoring_guru.iterator.example.spammer.SocialSpammer;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Demo class. Everything comes together here.
 */
public class Demo {
    public static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("Please specify social network to target spam tool (default:Fa
        System.out.println("1. Facebook");
        System.out.println("2. LinkedIn");
        String choice = scanner.nextLine();

        SocialNetwork network;
        if (choice.equals("2")) {
            network = new LinkedIn(createTestProfiles());
        }
        else {
            network = new Facebook(createTestProfiles());
        }

        SocialSpammer spammer = new SocialSpammer(network);
        spammer.sendSpamToFriends("anna.smith@bing.com",
                "Hey! This is Anna's friend Josh. Can you do me a favor and like this pos
        spammer.sendSpamToCoworkers("anna.smith@bing.com",
                "Hey! This is Anna's boss Jason. Anna told me you would be interested in
    }

    public static List<Profile> createTestProfiles() {
        List<Profile> data = new ArrayList<Profile>();
        data.add(new Profile("anna.smith@bing.com", "Anna Smith", "friends:mad_max@ya.com
        data.add(new Profile("mad_max@ya.com", "Maximilian", "friends:anna.smith@bing.com
        data.add(new Profile("bill@microsoft.eu", "Billie", "coworkers:avanger@ukr.net"))
        data.add(new Profile("avanger@ukr.net", "John Day", "coworkers:bill@microsoft.eu"
        data.add(new Profile("sam@amazon.com", "Sam Kitting", "coworkers:anna.smith@bing.
        data.add(new Profile("catwoman@yahoo.com", "Liza", "friends:anna.smith@bing.com",
        return data;
    }
}
```