

# AnonymousPrincipal - Duplicated Abstraction

---

## Chat 1: Before being told that it was a duplicated abstraction smell:

[Link to chat](#)

Here, we simply gave the code to GPT and it was unable to identify the duplicated abstraction design smell, only suggesting minor changes.

## Chat 2: After being told that it was a duplicated abstraction smell:

[Link to chat](#)

Certainly! To address the duplicated abstraction and improve the design, you can introduce a single **Principal** class that can represent both anonymous and authenticated users. Here's how you can refactor the code:

```
package com.sismics.security;

import org.joda.time.DateTimeZone;

import java.util.Locale;
import java.util.Set;

/**
 * Principal representing both anonymous and authenticated users.
 *
 * @author jtremeaux
 */
public class Principal implements IPrincipal {
    /**
     * ID of the user.
     */
    private String id;

    /**
     * Username of the user.
     */
    private String name;

    /**
     * Locale of the principal.
     */
    private Locale locale;

    /**
     * Timezone of the principal.
     */
    private DateTimeZone dateTimeZone;
```

```

/**
 * Email of the principal.
 */
private String email;

/**
 * Flag indicating if the principal is anonymous.
 */
private boolean anonymous;

/**
 * User base functions.
 */
private Set<String> baseFunctionSet;

/**
 * Constructor of Principal for an anonymous user.
 */
public Principal() {
    this.anonymous = true;
}

/**
 * Constructor of Principal for an authenticated user.
 *
 * @param id ID of the user
 * @param name Username of the user
 */
public Principal(String id, String name) {
    this.id = id;
    this.name = name;
    this.anonymous = false;
}

@Override
public boolean isAnonymous() {
    return anonymous;
}

@Override
public String getId() {
    return id;
}

/**
 * Setter of id.
 *
 * @param id id
 */
public void setId(String id) {
    this.id = id;
}

```

```
@Override
public String getName() {
    return name;
}

/**
 * Setter of name.
 *
 * @param name name
 */
public void setName(String name) {
    this.name = name;
}

@Override
public Locale getLocale() {
    return locale;
}

/**
 * Setter of locale.
 *
 * @param locale locale
 */
public void setLocale(Locale locale) {
    this.locale = locale;
}

@Override
public DateTimeZone getDateTimeZone() {
    return dateTimeZone;
}

/**
 * Setter of dateTimeZone.
 *
 * @param dateTimeZone dateTimeZone
 */
public void setDateTimeZone(DateTimeZone dateTimeZone) {
    this.dateTimeZone = dateTimeZone;
}

@Override
public String getEmail() {
    return email;
}

/**
 * Setter of email.
 *
 * @param email email
 */
public void setEmail(String email) {
    this.email = email;
}
```

```

    }

    /**
     * Getter of baseFunctionSet.
     *
     * @return baseFunctionSet
     */
    public Set<String> getBaseFunctionSet() {
        return baseFunctionSet;
    }

    /**
     * Setter of baseFunctionSet.
     *
     * @param baseFunctionSet baseFunctionSet
     */
    public void setBaseFunctionSet(Set<String> baseFunctionSet) {
        this.baseFunctionSet = baseFunctionSet;
    }
}

```

With this refactored code, you have a single `Principal` class that can represent both anonymous and authenticated users based on the value of the `anonymous` flag. This approach eliminates the need for separate `AnonymousPrincipal` and `UserPrincipal` classes, reducing duplication and improving code maintainability.

This was a very simple fix, so we attempted explaining the issue to GPT and obtained near-accurate promising results - closely resembling what we did for refactoring manually.