CLEAN CODE

CODE SMELLS EN LOS NOMBRES

Daniel Blanco Calviño

N1: NOMBRES NO DESCRIPTIVOS

- Utiliza nombres descriptivos.

```
int d;
int m;
int y;
```

```
int dayOfBirth;
int monthOfBirth;
int yearOfBirth;
```

N2: NOMBRES EN NIVEL DE ABSTRACCIÓN INCORRECTO

```
public interface FileDownloader {
    File download(String webUrl);
}
```

- Si tenemos una clase FtpFileDownloader que implementa la interfaz, ya no recibiría una web url.

N3: NO USAR NOMENCLATURA ESTÁNDAR

- **Usar la nomenclatura estándar** cuando sea posible.
- Usar la palabra *Singleton* para clases que usen dicho patrón, la palabra *Controller* para las clases de tu capa controlador etc.
- Sobrescribir métodos *toString* que ya tienen las clases java en lugar de crear un método propio...

N4: NOMBRES AMBIGUOS

- Evitar la ambiguedad en los nombres de las variables y de las funciones.

```
private String lstUsedName;
```

```
private String leastUsedName;
```

N5: NO USAR NOMBRES LARGOS PARA LARGOS ALCANCES

- Se pueden usar nombres de pocos caracteres para situaciones de alcance muy acotado.
- Usar nombres largos en situaciones de mayor alcance.

```
for (int i = 0; i < numberOfRequests; i++) {
    processRequest(requests.get(i));
}
```

```
public class Car {
    private int hp;
}
```

```
public class Car {
    private int horsePower;
}
```

N6: USAR CODIFICACIONES

- Evita los nombres con codificaciones que ensucien y distraigan al lector.
- Variables String strName o Integer intVariable.

N7: OCULTAR EFECTOS SECUNDARIOS

- Si existen efectos secundarios, los nombres de variables y funciones deben describirlos.

```
public Boolean login(User user) {
    User databaseUser = userRepository.findUserByUsername(user.getUsername());

    if(validCredentials(user, databaseUser)) {
        return true;
    }
    if(checkNumberOfTries(user.getIp()) > MAXIMUM_TRIES) {
        blockIpAddress(user.getIp());
    }

    return false;
}
```