

# DESCLASIFICACIÓN BASADA EN TIPOS EN DART

## IMPLEMENTACIÓN Y ELABORACIÓN DE HERRAMIENTAS DE INFERENCIA

---

Matías Meneses Cortés

1. Control de flujo de información
2. Inferencia de facetas públicas en Dart
3. Validación
4. Conclusiones y trabajo futuro

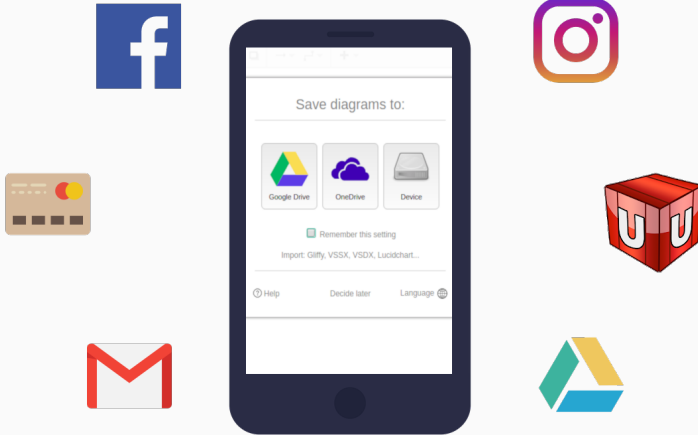
# Control de flujo de información

---

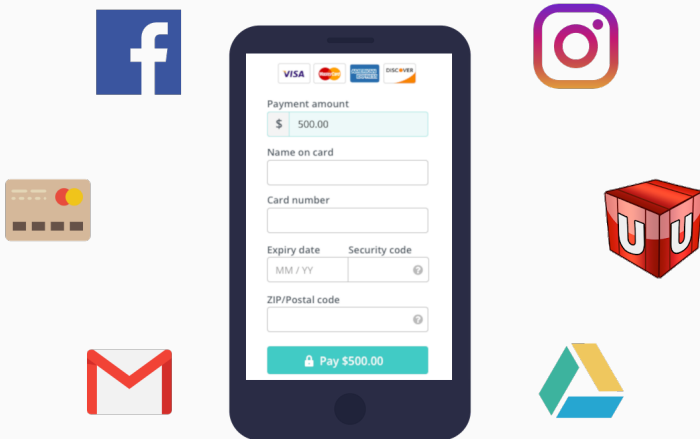


# Protección de confidencialidad





# Protección de confidencialidad



# Protección de confidencialidad

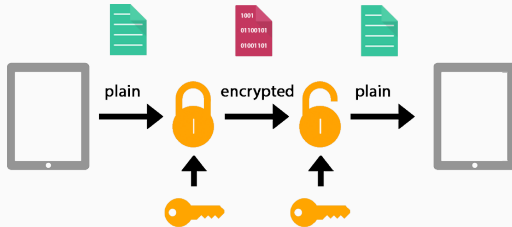




Distintas técnicas de seguridad en distintas capas de comunicación.

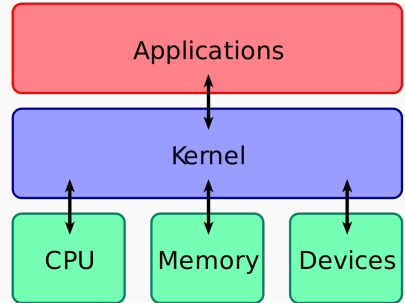
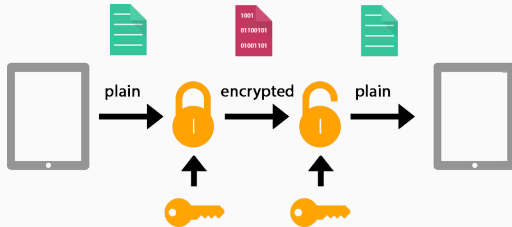
# Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.



# Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.





H  
|  
L

Orden parcial de dos niveles

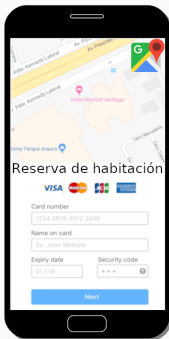
# Control de flujo de información



```
String book(String username, int date, int cardNumber) {  
    return sendToHotel(username, date, cardNumber);  
}
```

```
String sendToHotel(String username, int date, int cardNumber);  
String sendToGoogle(String token, int xCoord, int yCoord);
```

# Control de flujo de información



```
String book(String username, int date, int cardNumber) {  
    return sendToGoogle (username, date, cardNumber);  
}
```

```
String sendToHotel(String username, int date, int cardNumber);  
String sendToGoogle(String token, int xCoord, int yCoord);
```

# Tipado de seguridad para el control de flujo de información

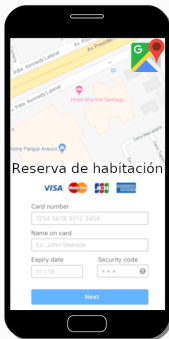


```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToGoogle(username, date, cardNumber);  
}
```

```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```



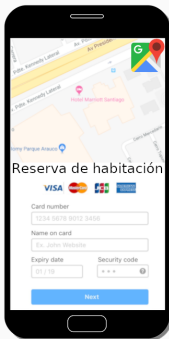
# Tipado de seguridad para el control de flujo de información



```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToGoogle(username, date, cardNumber);  
}
```

```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```

# Tipado de seguridad para el control de flujo de información

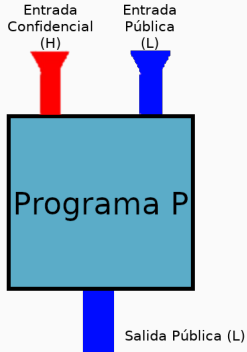


```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToHotel(username, date, cardNumber);  
}
```

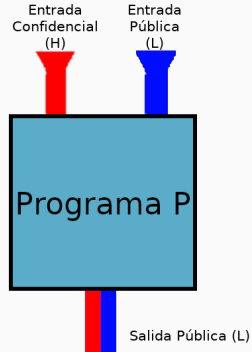
```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```

# No-interferencia

Propiedad fundamental del control de flujo de información.



No-interferencia



Fuga de información

```
String@L login(String@L guess, String@H password) {  
    if (password == guess) return "Login successful";  
    else return "Login failed";  
}
```

## Problema con no-interferencia

```
String@L login(String@L guess, String@H password) {  
    if (password == guess) return "Login successful";  
    else return "Login failed";  
}
```

¡No cumple con no-interferencia!

```
String@L login(String@L guess, String@H password) {  
    if (declassify(password == guess)) return "Login successful";  
    else return "Login failed";  
}
```

```
declassify(password)
```

```
declassify(password)
```

¡Grave fuga de información!



```
String<String login(String<String guess, String<StringEq password) {  
  if (password.eq(guess)) return "Login successful";  
  else return "Login failed";  
}
```

```
String<String login(String<String guess, String<StringEq password) {  
    if (password.eq(guess)) return "Login successful";  
    else return "Login failed";  
}
```

- Tipos de dos facetas `String<StringEq`

## Desclasificación basada en tipos

```
String<String login(String<String guess, String<StringEq password) {  
  if (password.eq(guess)) return "Login successful";  
  else return "Login failed";  
}
```

- Tipos de dos facetas `String<StringEq`
- `StringEq = [eq: String<String -> Bool<Bool]`

## Desclasificación basada en tipos

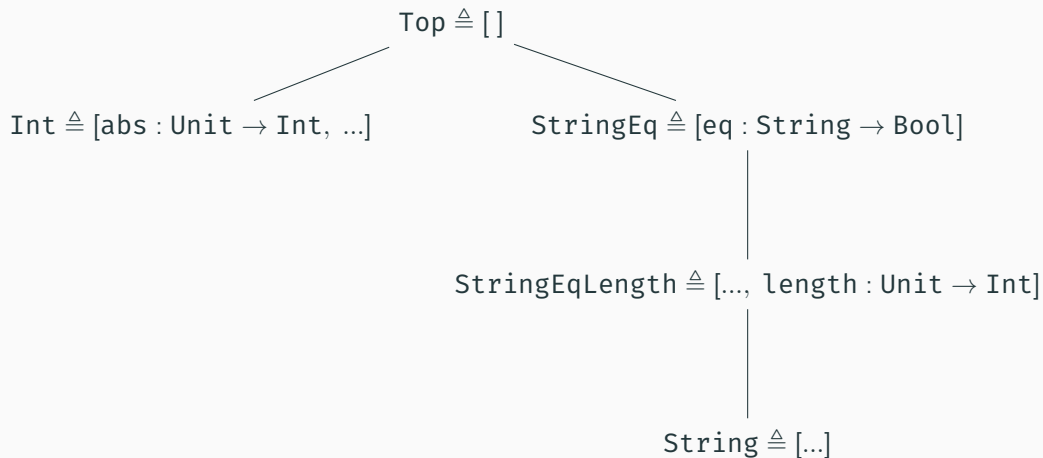
```
String<String login(String<String guess, String<StringEq password) {  
  if (password.eq(guess)) return "Login successful";  
  else return "Login failed";  
}
```

- Tipos de dos facetas `String<StringEq`
- `StringEq = [eq: String<String -> Bool<Bool]`
- `String <: StringEq` (Tipo bien formado)

## Desclasificación basada en tipos

```
String<String login(String<String guess, String<StringEq password) {  
  if (password.eq(guess)) return "Login successful";  
  else return "Login failed";  
}
```

- Tipos de dos facetas `String<StringEq`
- `StringEq = [eq: String<String -> Bool<Bool]`
- `String <: StringEq` (Tipo bien formado)
- No-interferencia relajada



## Problemas con la desclasificación basada en tipos

- Propuesta sin implementación práctica.



## Problemas con la desclasificación basada en tipos

- Propuesta sin implementación práctica.
- Anotación completa de facetas para realizar análisis.

# Problemas con la desclasificación basada en tipos

- Propuesta sin implementación práctica.
- Anotación completa de facetas para realizar análisis.

```
String login(String guess, String<StringEq password) {  
    if (password.eq(guess)) return "Login successful";  
    else return "Login failed";  
}
```

## Objetivo de la memoria

Implementar un sistema de inferencia de facetas públicas para la desclasificación basada en tipos, en conjunto con una extensión para ambientes de desarrollo.

## Inferencia de facetas públicas en Dart

---

## Ejemplo 1

```
bool login(String guess, String password) {  
    return password.eq(guess);  
}
```

## Ejemplo 1

```
bool login(String guess, String<StringEq password) {  
    return password.eq(guess);  
}
```

```
StringEq = [eq: String<String -> bool<bool]
```

## Ejemplo 1

```
bool login(String<String guess, String<StringEq password) {  
    return password.eq(guess);  
}
```

Faceta pública de métodos que pertenecen a los tipos por defecto: **Bot** → **Bot**.

# Problema de inferencia a resolver

## Ejemplo 1

```
bool<X login(String<String guess, String<StringEq password) {  
    return password.eq(guess);  
}
```

Faceta pública de métodos que pertenecen a los tipos por defecto: **Bot** → **Bot**.



## Ejemplo 2

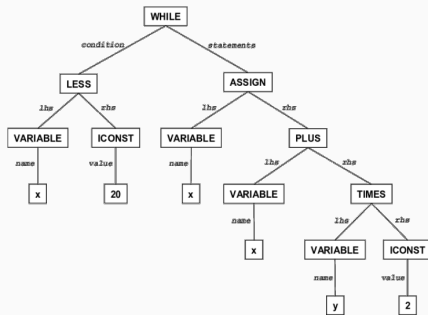
```
bool login(String<String guess, String<Top password) {  
    return password.eq(guess);  
}
```

## Ejemplo 2

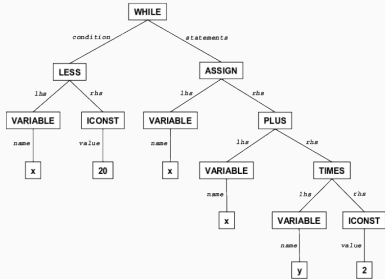
```
bool<Top login(String<String guess, String<Top password) {  
    return password.eq(guess);  
}
```



# Dart



Abstract Syntax Tree



Abstract Syntax Tree



Errores y sugerencias de edición

```
class Foo {
```

```
class Foo {  
  String foo(String a, String b) {
```

```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
  }  
}
```



```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
    if (a == b)
```

```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
    if (a == b) return a.concat(b);  
    return s;  
  }  
}
```

Uso de anotaciones de Dart para declarar las facetas públicas.

Uso de anotaciones de Dart para declarar las facetas públicas.

```
bool login(String guess, @S("StringEq") String password) {  
  return password.eq(guess);  
}
```

## Definición de facetas públicas

Uso de clases abstractas de Dart para definir las facetas públicas.

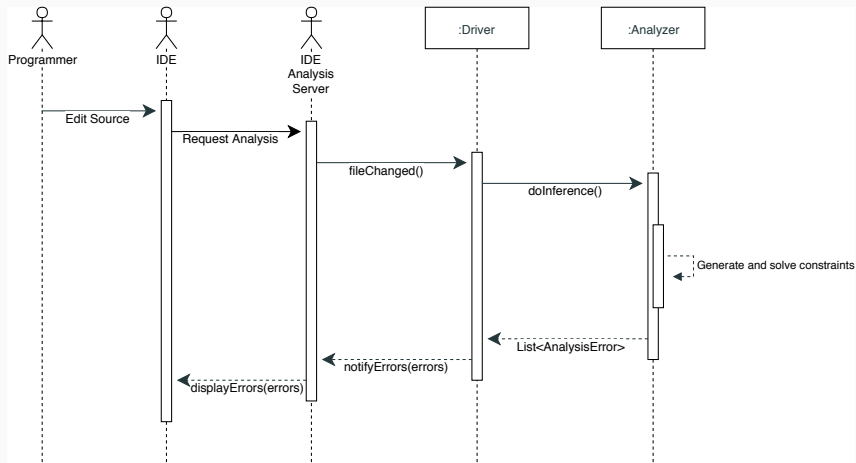
# Definición de facetas públicas

Uso de clases abstractas de Dart para definir las facetas públicas.

```
bool login(String guess, @S("StringEq") String password) {  
    return password.eq(guess);  
}
```

```
abstract class StringEq {  
    bool eq(String other);  
}
```

# Extensión para ambientes de desarrollo



Errores de seguridad



## Errores de seguridad

```
bool<bool login(String guess, @S("Top") String password) {  
    return password.eq(guess);  
}
```

Errores de tipo mal formado

Errores de tipo mal formado

```
bool<Top login(String guess, @S("Abs") String password) {  
    return password.eq(guess);  
}  
  
abstract class Abs {  
    int abs();  
}
```

Warning de faceta pública no definida

Warning de faceta pública no definida

```
bool<bool login(String guess, @S("StringEq") String password) {  
    return password.eq(guess);  
}
```

Información de faceta pública inferida

Información de faceta pública inferida

```
bool<bool login(String<String guess, String password) {  
    return password.eq(guess);  
}
```

Información de faceta pública inferida

```
bool<bool login(String<String guess, String password) {  
    return password.eq(guess);  
}
```

```
password: [eq: String<String -> bool<bool]
```



**Tabla 1:** Métricas de la implementación

Líneas de código	Clases
2866	42

# Validación

---

# Ejemplo: Sistema de autenticación web

## Secure Login

Enter your credentials

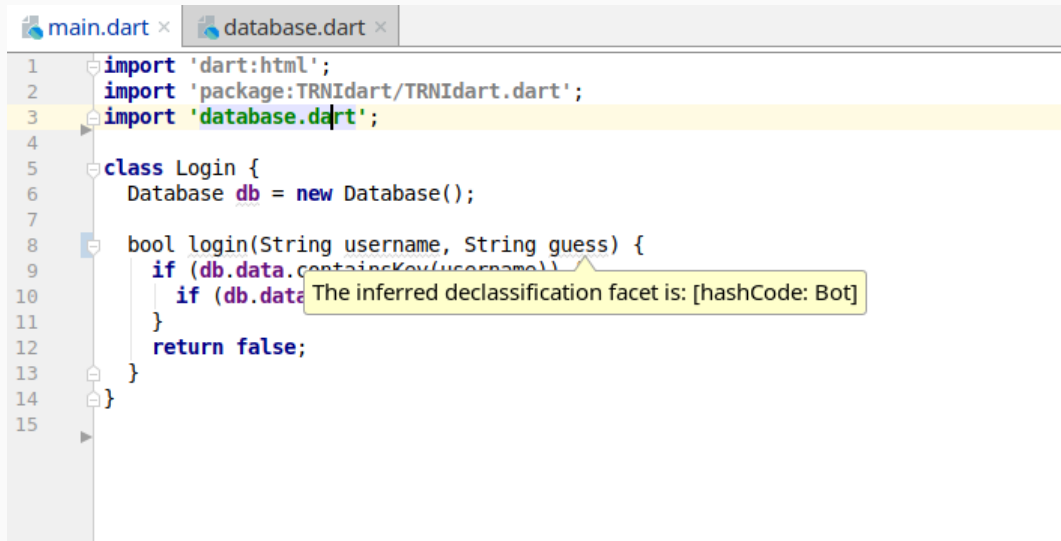
Enter your email

Enter your password

[Forgot Password?](#)

LOGIN

## Ejemplo: Sistema de autenticación web



```
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Login {
6   Database db = new Database();
7
8   bool login(String username, String guess) {
9     if (db.data.containsKey(username)) {
10       if (db.data.containsKey(guess)) {
11         return true;
12       }
13     }
14     return false;
15   }
16 }
```

The inferred declassification facet is: [hashCode: Bot]

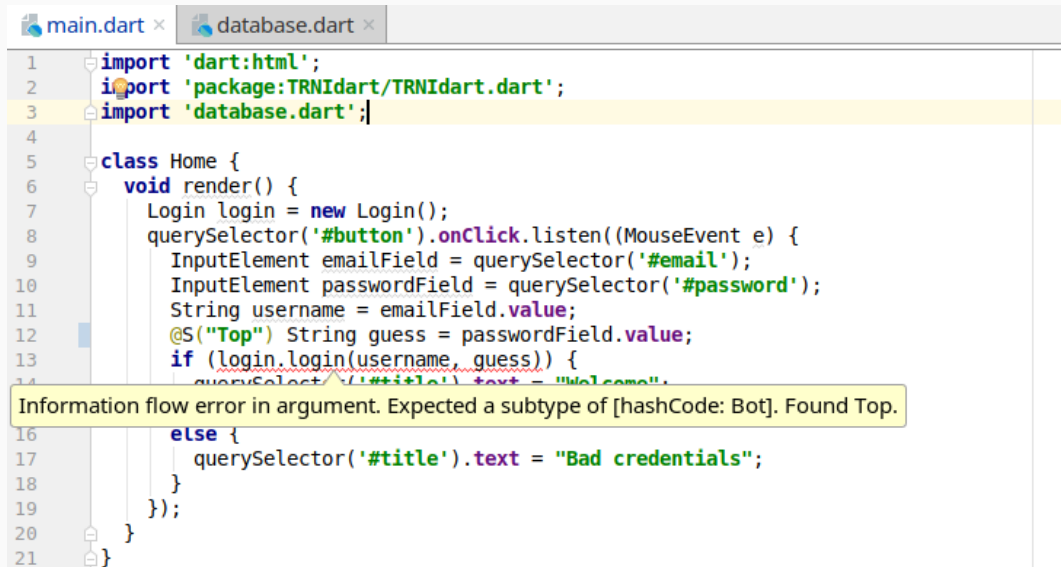
## Ejemplo: Sistema de autenticación web



```
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Login {
6   Database db = new Database();
7
8   bool login(String username, @S("Hash") String guess) {
9     if (db.data.containsKey(username)) {
10       if (db.data[username] == guess) {
11         return true;
12       }
13     }
14     return false;
15   }
16 }
```

The declared facet Hash is not defined.

## Ejemplo: Sistema de autenticación web



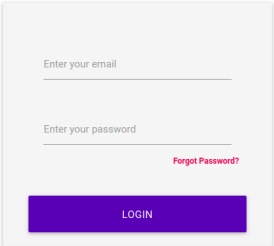
The screenshot shows an IDE with two tabs: `main.dart` and `database.dart`. The `main.dart` file contains the following Dart code:

```
1 import 'dart:html';
2 import 'package:TRNIdart/TRNIdart.dart';
3 import 'database.dart';
4
5 class Home {
6   void render() {
7     Login login = new Login();
8     querySelector('#button').onClick.listen((MouseEvent e) {
9       InputElement emailField = querySelector('#email');
10      InputElement passwordField = querySelector('#password');
11      String username = emailField.value;
12      @S("Top") String guess = passwordField.value;
13      if (login.login(username, guess)) {
14        querySelector('#title').text = "Welcome";
15      } else {
16        querySelector('#title').text = "Bad credentials";
17      }
18    });
19  }
20 }
21 }
```

A yellow tooltip is displayed over the code, indicating a type error:

Information flow error in argument. Expected a subtype of [hashCode: Bot]. Found Top.

# Ejemplo: Sistema de autenticación web



A login form titled "Secure Login" with the subtitle "Enter your credentials". It contains two input fields: "Enter your email" and "Enter your password". Below the password field is a red link labeled "Forgot Password?". At the bottom is a blue button labeled "LOGIN".

Secure Login  
Enter your credentials

Enter your email

Enter your password

[Forgot Password?](#)

LOGIN

Código en repositorio GitHub [1]

**Tabla 2:** Anotación de facetas públicas en identificadores

Con inferencia	Sin inferencia
7	22



14

Test unitarios en el repositorio del proyecto [2].

## Conclusiones y trabajo futuro

---

- Conexión entre abstracciones de tipo y relaciones de orden de etiquetas de seguridad.

- Conexión entre abstracciones de tipo y relaciones de orden de etiquetas de seguridad.
- Integración de conceptos de control de flujo de información con infraestructuras existentes.

- Conexión entre abstracciones de tipo y relaciones de orden de etiquetas de seguridad.
- Integración de conceptos de control de flujo de información con infraestructuras existentes.
- Implementación para un subconjunto del lenguaje Dart que demuestra la utilidad de desclasificación basada en tipos.

Formalización de inferencia.

Extensión al subconjunto soportado de Dart.

Características de la extensión para ambientes de desarrollo.





M. M. C.

Secure login screen, programmed with type-based security types.

<https://github.com/matiasimc/secure-login-test>.



M. M. C.

Type based declassification inference for dart.

<https://github.com/matiasimc/TRNIdart>.

# Preguntas

# Inferencia de tipos en Scala

```
def mathFunction(n1: Int, n2: Float) = {  
  val n = fact(n1);  
  n + n2;  
}
```

```
def fact(n: Int) : Int = {  
  if (n == 0) return 1  
  else n * fact(n-1)  
}
```

- Inferencia de tipos local
- Soporte de overloading y conversiones implícitas de tipos

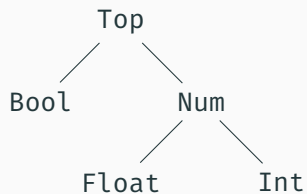
# Inferencia de tipos en OCaml

```
# let average a b =  
    (a +. b) /. 2.0;;  
  
val average : float -> float -> float
```

- Inferencia de tipos global
- No soporta overloading y conversiones implícitas

# Inferencia de tipos

```
calculate(c, Int n) {  
  if (c) return n*2;  
  else return n*0.5;  
}
```



## Variables de tipo

```
X calculate(Y c, Int n) {  
  if (c) return n*2;  
  else return n*0.5;  
}
```

## Generación de restricciones

```
X calculate(Y c, Int n) {  
  if (c) return n*2;  
  else return n*0.5;  
}
```

1.  $Y <: \text{Bool}$
2.  $\text{Int} <: X$
3.  $\text{Float} <: X$

# Encadenamiento de invocaciones a métodos

## Ejemplo 3

```
bool login(int guess, String password) {  
    return password.hash().eq(guess);  
}
```



# Encadenamiento de invocaciones a métodos

## Ejemplo 3

```
bool<bool login(String<String guess, String<StringHash password) {  
    return password.hash().eq(guess);  
}
```

```
StringHash = [hash: () -> int< -]
```

# Encadenamiento de invocaciones a métodos

## Ejemplo 3

```
bool<bool login(String<String guess, String<StringHash password) {  
    return password.hash().eq(guess);  
}
```

```
StringHash = [hash: () -> int<IntEq]
```

```
IntEq = [eq: int<int -> bool<bool]
```