Classless Java

No Author Given

No Institute Given

Abstract.

1 Introduction

- Using annotattions to implement a rich notion of traits with a mechanism to instantiate objects (the of method). Goal 1: is to reduce the amount of code that is required to program with interfaces and default methods. Goal 2: To provide a convenient means to do multiple inheritance in Java.
- Specify the system more formally.
- Show that we can model all trait operations
- Implementation using Lombok
- Case studies: The expression problem, Trivially Case Studies from traits paper.

2 Overview

Yanlin

* Explain what the Mixin annotations do using examples.

* Motivate the use of multiple inheritance in Java.

* Maybe use Marco's example (Point example).

We provide a Java annotation **@Mixin**to provide default implementations for various methods and a mechanism to instantiate objects. **@Mixin**annotation helps programmers to write less cumbersome code and instantiate interfaces in Java.

For example, interface `Point` annotated with **@Mixin**:

```
@Mixin
interface Point {
   int X();
   int Y();
   void X(int x);
   void Y(int y);
   Point withX(int x);
   Point withY(int y);
   Point clone();
}
```

`Point` has two (conceptually) member fields `X` and `Y`. Methods `int X()` and `int Y()` serve as *getter* methods. Methods `void X(int X)` and `void Y(int Y)` serve as *setter* methods. Method `Point withX(int X)` updates field `X` and returns **this**.

A typical and trivial implementation that programmers usually do is:

```
class PointImpl implements Point {
   private int _X;
   private int _Y;
   public PointImpl(int X, int Y) {
      this._X = X;
      this._Y = Y;
   }
   public int X() {
      return _X;
   }
   public int Y() {
      return _Y;
   }
   public Point withX(int X) {
      X(X);
      return this;
   }
   public void X(int X) {
      _X = X;
   }
   public void Y(int Y) {
      _Y = Y;
   }
   public Point withY(int Y) {
      Y(Y);
      return this;
   }
   public Point clone() {
      return new PointImpl(_X, _Y);
   }
}
```

```
   static Point of(int X, int Y) {
      return new Point() {
         int _X = X;
         public int X() {
            return _X;
         }
         int _Y = Y;
         public int Y() {
            return _Y;
         }
         public Point withX(int X) {
            return of(X, Y());
         }
         public void X(int X) {
            _X = X;
         }
```

```
        public void Y(int Y) {
           _Y = Y;
        }
        public Point withY(int Y) {
           return of(X(), Y);
        }
        public Point clone() {
           return of(X(), Y());
        }
     };
  }
```

3 Comparing to traits and mixins

Haoyuan

- vs both: we do automatic return type refinement, which has useful applications (example: Expression Problem)

- vs traits: we support of methods to create new objects (a replacement to constructors); Moreover we have the with and clone methods (we miss more applications for those). Show how to model the operations on traits; discuss operations that we cannot model (example: renaming).

- vs mixins: we use the trait model of explicitly resolving conflicts. This is arguably better for reasoning.

4 Formal Semantics

Yanlin and Haoyuan

We need to show 2 things:

1) The dynamic semantics: what's the code that gets generated by a mixin annotation;

2) The type system: what programs to reject; properties: generation of type-safe/checkable code.

BRUNO: The implementation is still missing the type system (rejecting some programs)!

5 Implementation

Haoyuan

discuss implementation in lombok; and limitations.

BRUNO: The implementation does not support separate compilation yet. Can we fix this?

6 Case studies

Haoyuan and Yanlin

6.1 A Trivial Solution to the Expression Problem

6.2 Other case studies

BRUNO: The case studies still need to be implemented!

Collections example from traits paper? (YANLIN: couldn't found source code)

YANLIN: We're modeling "provided methods" by default methods in Java 8 interfaces. How to model "required methods" in traits. Example: in traits paper (Traits: Composable Units of Behaviour), the TCircle, TDrawing example.

Other case studies using multiple inheritance?

7 Related Work

- traits (original, variations, scala) - mixins (original, scala) - multiple inheritance - expression problem - ...

8 Conclusion