Reified Type Parameters Using Java Annotations

a proposal for full featured generics without code duplication

Prodromos Gerakios, **Aggelos Biboudis**, Yannis Smaragdakis

University of Athens

Motivation: Generics & Type Erasure

```
class ArrayList<X>{
  X [] arr …
}
```

↓

```
class ArrayList {
  Object [] arr
}
```

No information for generic parameters after erasure: no new T, extends T, T.class where T generic parameter :-(

New Code Patterns

```
class ReifiedGeneric <@reify X,Y> {
  Class classOfX = X.class;
  Y id(Y y) { return y; }
  X newInstance() { return new X(); }
}
```

X is reified

```
class Serial <@reify T> extends T {
  public long getSerialNumber() { … }
}
```

mixin pattern

```
Serial <Customer> customer =
      new Serial <Customer>();

customer.getSerialNumber();
```

JSR 308 & Checker Framework

- What we will use:
  - A new location for @reify brought by JSR 308
  - The Checker Framework for plugging into the Java Compiler


```
class C<@reify X,Y> {}
```

Case A: Plain Generation

```
class Foo<@reify X> {
  void meth() { X local = new X(); }
}
```

The interface contains
a) methods implied
b) operations with T

(Shared part)

```
interface iface$Foo<X> {
  X new$X();
  void meth();
}

class Foo<X> implements iface$Foo<X> {

  X new$X() { return null; }
  void meth() { X local = new$X(); }
}
```

Case B: Mixin Generation

What if decode exists in X?

```
class Foo<@reify X> extends X {
  Integer decode(String nm){ ... }
}
```

Generation as in Case A, but we also need to constraint X:

```
interface Constraint<X> {
  Integer decode(String nm);
}

class Foo<@reify(Constraint.class) X>
extends X {
  Integer decode(String nm){ ... }
}
```

Generation at Instantiation Point

```
foo = new Foo<Integer>();
```

If Foo has a reifiable param then:

```
class Foo$Integer
extends Foo<Integer> {
  Integer new$X(){
     return new Integer(); }
}
```

Translating by expansion

If Foo is a mixin, then also:

```
class Foo$$Integer extends Integer
implements iface$Foo<Integer> {
  iface$Foo<Integer> mixin =
           new Foo$Integer();

  Integer decode(String nm) {
    return mixin.decode(nm);
  }
}
```

Delegation under the hood!