Annotation processing

Any interface annotate with @Mixin is processed in the following way:

Examining the code of the interface and the code of the implemented interface(transitivelly) we get the following data for all the methods in the interface:

- a) MethodName,
- b) isAbstract/isDefaultImplemented
- c) number of parameters and their types
- d) return type

Example

```
@Mixin
public interface Point {
    int x();
    int y();
    Point withX(int x);
    void x(int x);
    void y(int x);
    default int distance(){
        return (int) Math.sqrt(x()*x()+y()*y());
    }
}
```

should (conceptually) generate:

```
public interface Point {
    int x();
    int y();
    Point withX(int x);
    void x(int x);
    void y(int x);
    default int distance(){
        return (int) Math.sqrt(x()*x()+y()*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; }
    };}
}
```

processing: Generate the `of` method

Generate `of` method, serves as the constructor of the annotated interface, using an anonymous inner class implementing all the methods in the annotated interface. Detailed procedure of how to generate these methods are described below:

Generate get method: `Tx x()`:

- `x` is the getter method, with return type `Tx`. Conceptually, it is a member field with name "x" and type `Tx`.
- generate member filed `_x` of type `Tx`, initialized with `x`
- generate method

```
public Tx x() { return _x; }
```

Generate set method: `void x(Tx x)`:

- check if exist method `Tx x()`
- inside the inner class, generate

```
pubic void x(Tx x) { this.x = x; }
```

Generate `T withX(Tx _)` method:

- if there's no `x` field, or `Tx` doesn't match, then generate error. else:
- implement `withX` using the `of` method.

Generate `T clone()` method: TODO

Generate `T withX()` method: TODO