//Existiam poucos casos que continha Left/Right/Base, a maioria faltava pelo menos um.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_56eb2a7_b20ae0a/rev_56eb2a7-b20ae0a/src/main/java/graphql/execution/ExecutionContextBuilder.java;<<<<<<< MINE

```
        ExecutionContext executionContext = new ExecutionContext(instrumentation);
        executionContext.setGraphQLSchema(graphQLSchema);
        executionContext.setExecutionStrategy(executionStrategy);
        executionContext.setOperationDefinition(operation);
        executionContext.setRoot(root);
        executionContext.setFragmentsByName(fragmentsByName);
        Map<String, Object> variableValues =
valuesResolver.getVariableValues(graphQLSchema, operation.getVariableDefinitions(),
args);
        executionContext.setVariables(variableValues);
        return executionContext;
||||||| BASE
        ExecutionContext executionContext = new ExecutionContext();
        executionContext.setGraphQLSchema(graphQLSchema);
        executionContext.setExecutionStrategy(executionStrategy);
        executionContext.setOperationDefinition(operation);
        executionContext.setRoot(root);
        executionContext.setFragmentsByName(fragmentsByName);
        Map<String, Object> variableValues =
valuesResolver.getVariableValues(graphQLSchema, operation.getVariableDefinitions(),
args);
        executionContext.setVariables(variableValues);
        return executionContext;
=======
        return new ExecutionContext(
                executionId,
                graphQLSchema,
                queryStrategy,
                mutationStrategy,
                fragmentsByName,
                operation,
                variableValues,
                root);
>>>>>>> YOURS
```

Mudança: Left adiciona parâmetro a ExecutionContext, enquanto Right adiciona um return.
Categoria: aFP(UN)
Justificativa: As mudanças são independentes , podendo serem mescladas através do merge automático do SS.
Como resolver: Substituir returnContext pelo do right.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_e4004d3_f7c259f/rev_e4004d3-f7c259f/src/main/java/graphql/Scalars.java;<<<<<<<
MINE
    public static GraphQLScalarType GraphQLLong = new GraphQLScalarType("Long", "Long type", new Coercing<Long>() {
||||||| BASE

    public static GraphQLScalarType GraphQLLong = new GraphQLScalarType("Long", "Long type", new Coercing<Long>() {
=======

    public static GraphQLScalarType GraphQLLong = new GraphQLScalarType("Long", "Long type", new Coercing<Long, **Long**>() {
>>>>>>> YOURS

Mudança: Right põe mais um parâmetro em new Coercing.
Categoria: aFP(UN)
Justificativa: Na hora da junção, pensando em função de como o SS resolveria, apenas o Long será acrescentado, como o Left não usa GraphQLLong em sua mudança, não devemos esperar comportamentos indesejados.
Como resolver: provavelmente, resolvendo com o yours seria a melhor solução.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_e4004d3_f7c259f/rev_e4004d3-f7c259f/src/main/java/graphql/Scalars.java;<<<<<<<
MINE

    public static GraphQLScalarType GraphQLChar = new GraphQLScalarType("Char", "Built-in Char as Character", new Coercing<Character>() {
||||||| BASE
    public static GraphQLScalarType GraphQLChar = new GraphQLScalarType("Char", "Built-in Char as Character", new Coercing<Character>() {
=======
    public static GraphQLScalarType GraphQLChar = new GraphQLScalarType("Char", "Built-in Char as Character", new Coercing<Character, **Character**>() {
>>>>>>> YOURS

Mudança: Right põe mais um parâmetro em new Coercing.
Categoria: aFP(UN)
Justificativa: Na hora da junção, apenas o Character será acrescentado, como o Left não usa GraphQLChar em sua mudança, não devemos esperar comportamentos indesejados.
Como resolver: provavelmente, resolvendo com o yours seria a melhor solução.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_e4004d3_f7c259f/rev_e4004d3-f7c259f/src/main/java/graphql/Scalars.java;<<<<<<<
MINE
    public static GraphQLScalarType **GraphQLShort** = new GraphQLScalarType("Short",

"Built-in Short as Int", new Coercing<Short>() {
||||||| BASE
    public static GraphQLScalarType GraphQLFloat = new GraphQLScalarType("Float",
"Built-in Float", new Coercing<Double>() {
=======
    public static GraphQLScalarType GraphQLFloat = new GraphQLScalarType("Float",
"Built-in Float", new Coercing<Double, **Double**>() {
>>>>>>> YOURS

Mudança: O Left passou a ter o GraphQLShort e o Right adicionou parâmetro a new Coercing.
Categoria: aFP(UN)
Justificativa: Não há problemas em manter os dois, uma vez que são apenas declarados aqui.
Como resolver: O integrador não teria problemas em unir essas partes, já que se trata de declaração/inicialização. Não teria um lugar específico para eles, apenas que esteja antes do seu primeiro uso.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_e4004d3_f7c259f/rev_e4004d3-f7c259f/src/main/java/graphql/Scalars.java;<<<<<<<
MINE
    public static GraphQLScalarType **GraphQLByte** = new GraphQLScalarType("Byte",
"Built-in Byte as Int", new Coercing<Byte>() {
||||||| BASE
    public static GraphQLScalarType GraphQLString = new GraphQLScalarType("String",
"Built-in String", new Coercing<String>() {
=======
    public static GraphQLScalarType GraphQLString = new GraphQLScalarType("String",
"Built-in String", new Coercing<String, **String**>() {
>>>>>>> YOURS

Mudança: O Left passou a ter o GraphQLByte e o Right adicionou parâmetro a new Coercing.
Categoria: aFP(UN)
Justificativa: Não há problemas em manter os dois, uma vez que são apenas declarados aqui.
Como resolver: O integrador não teria problemas em unir essas partes, já que se trata de declaração/inicialização. Não teria um lugar específico para eles, apenas que esteja antes do seu primeiro uso.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_e4004d3_f7c259f/rev_e4004d3-f7c259f/src/main/java/graphql/Scalars.java;<<<<<<<
MINE
    public static GraphQLScalarType **GraphQLFloat** = new GraphQLScalarType("Float",

"Built-in Float", new Coercing<Double>() {
||||||| BASE
    public static GraphQLScalarType GraphQLID = new GraphQLScalarType("ID", "Built-in ID", new Coercing<Object>() {
=======
    public static GraphQLScalarType GraphQLID = new GraphQLScalarType("ID", "Built-in ID", new Coercing<Object, **Object**>() {
>>>>>>> YOURS
Mudança: O Left passou a ter o GraphQLFloat e o Right adicionou parâmetro a new Coercing.
Categoria: aFP(UN)
Justificativa: Não há problemas em manter os dois, uma vez que são apenas declarados aqui.
Como resolver: O integrador não teria problemas em unir essas partes, já que se trata de declaração/inicialização. Não teria um lugar específico para eles, apenas que esteja antes do seu primeiro uso.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_e4004d3_f7c259f/rev_e4004d3-f7c259f/src/main/java/graphql/Scalars.java;<<<<<<< MINE

```
    public static GraphQLScalarType GraphQLString = new GraphQLScalarType("String", "Built-in String", new Coercing<String>() {
        @Override
        public String serialize(Object input) {
            return input == null ? null : input.toString();
        }

        @Override
        public String parseValue(Object input) {
            return serialize(input);
        }

        @Override
        public String parseLiteral(Object input) {
            if (!(input instanceof StringValue)) return null;
            return ((StringValue) input).getValue();
        }
    });


    public static GraphQLScalarType GraphQLBoolean = new GraphQLScalarType("Boolean", "Built-in Boolean", new Coercing<Boolean>() {
||||||| BASE
    public static GraphQLScalarType GraphQLByte = new GraphQLScalarType("Byte", "Built-in Byte as Int", new Coercing<Byte>() {
```

=======
    public static GraphQLScalarType GraphQLByte = new GraphQLScalarType("Byte", "Built-in Byte as Int", new Coercing<Byte, **Byte**>() {
>>>>>>> YOURS

Mudança: O Left passou a ter o GraphQLBoolean e o Right adicionou parâmetro a new Coercing.
Categoria: aFP(UN)
Justificativa: Não há problemas em manter os dois, uma vez que são apenas declarados aqui.
Como resolver: O integrador não teria problemas em unir essas partes, já que se trata de declaração/inicialização. Não teria um lugar específico para eles, apenas que esteja antes do seu primeiro uso.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_e4004d3_f7c259f/rev_e4004d3-f7c259f/src/main/java/graphql/Scalars.java;<<<<<<< MINE

    public static GraphQLScalarType **GraphQLID** = new GraphQLScalarType("ID", "Built-in ID", new Coercing<Object>() {
||||||| BASE
    public static GraphQLScalarType GraphQLShort = new GraphQLScalarType("Short", "Built-in Short as Int", new Coercing<Short>() {
=======
    public static GraphQLScalarType GraphQLShort = new GraphQLScalarType("Short", "Built-in Short as Int", new Coercing<Short, **Short**>() {
>>>>>>> YOURS

Mudança: O Left passou a ter o GraphQLID o Right adicionou parâmetro a new Coercing.
Categoria: aFP(UN)
Justificativa: Não há problemas em manter os dois, uma vez que são apenas declarados aqui.
Como resolver: O integrador não teria problemas em unir essas partes, já que se trata de declaração/inicialização. Não teria um lugar específico para eles, apenas que esteja antes do seu primeiro uso.

/home/paper219/Desktop/analysis/demonstration/projects/truth/revisions/rev_d7a2ff6_db7ed46/rev_d7a2ff6-db7ed46/src/main/java/org/junit/contrib/truth/subjects/ListSubject.java;<<<<<<< MINE
@GwtCompatible
||||||| BASE
=======
import org.junit.contrib.truth.FailureStrategy;
>>>>>>> YOURS

Mudança: Left adiciona annotation e Right adiciona um import.
Categoria: aFP(UN)
Justificativa: As mudanças não são conflitantes, pode pôr as duas sem problemas.
Como resolver: O local de se pôr o import é mais acima na classe, annotation provavelmente em cima de um método. O integrador só deveria se atentar a isso pra ver a melhor forma de integrar.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_9d94784_8f1681d/rev_9d94784-8f1681d/src/main/java/graphql/execution/ExecutionContext.java;<<<<<<< MINE
   private final ExecutionId executionId;
   private final ExecutionStrategy executionStrategy;
||||||| BASE
   private final ExecutionStrategy executionStrategy;
=======
   private final ExecutionStrategy queryStrategy;
   private final ExecutionStrategy mutationStrategy;
>>>>>>> YOURS

Mudança: Left adiciona um atributo e right adiciona/renomeia outros.
Categoria: aFP(UN)
Justificativa: As mudanças não são conflitantes, pode pôr as duas sem problemas.
Como resolver: A ordem delas não importam, seria fácil de integrar juntando-as.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_9d94784_8f1681d/rev_9d94784-8f1681d/src/main/java/graphql/execution/ExecutionContext.java;<<<<<<< MINE
   public ExecutionContext(GraphQLSchema graphQLSchema, ExecutionId executionId, ExecutionStrategy executionStrategy, Map<String, FragmentDefinition> fragmentsByName, OperationDefinition operationDefinition, Map<String, Object> variables, Object root) {
||||||| BASE
   public ExecutionContext(GraphQLSchema graphQLSchema, ExecutionStrategy executionStrategy, Map<String, FragmentDefinition> fragmentsByName, OperationDefinition operationDefinition, Map<String, Object> variables, Object root) {
=======
   public ExecutionContext(GraphQLSchema graphQLSchema, ExecutionStrategy queryStrategy, ExecutionStrategy mutationStrategy, Map<String, FragmentDefinition> fragmentsByName, OperationDefinition operationDefinition, Map<String, Object> variables, Object root) {
>>>>>>> YOURS

Mudança: Right põe mais um parâmetro no construtor (ou põe outro construtor).
Categoria: aFP(UN)
Justificativa: Aqui, o resultado seria ter mais construtores, não há problemas com isso.
Como resolver: A ordem delas não importam, seria fácil de integrar juntando-as.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_9d94784_8f1681d/rev_9d94784-8f1681d/src/main/java/graphql/execution/ExecutionContextBuilder.java;<<<<<<< MINE

```
   public ExecutionContextBuilder executionId(ExecutionId executionId) {
      this.executionId = executionId;
      return this;
   }

   public ExecutionContext build(GraphQLSchema graphQLSchema, ExecutionStrategy executionStrategy, Object root, Document document, String operationName, Map<String, Object> args) {
      // preconditions
      assertNotNull(executionId,"You must provide a query identifier");
```

||||||| BASE

```
   public ExecutionContext build(GraphQLSchema graphQLSchema, ExecutionStrategy executionStrategy, Object root, Document document, String operationName, Map<String, Object> args) {
```

=======

```
   public ExecutionContext build(GraphQLSchema graphQLSchema, ExecutionStrategy queryStrategy, ExecutionStrategy mutationStrategy, Object root, Document document, String operationName, Map<String, Object> args) {
```

>>>>>>> YOURS

Mudança: Right põe mais um parâmetro ao método build, Left adiciona comentário e provavelmente uma chamada de método.
Categoria: aFP(UN)
Justificativa: Na hora da junção, apenas o ExecutionStrategy será acrescentado, como o Left não chama build em sua mudança, não devemos esperar comportamentos indesejados aqui.
Como resolver: Adicionando o parâmetro extra ao método build e unindo as outras mudanças.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_9d94784_8f1681d/rev_9d94784-8f1681d/src/main/java/graphql/execution/Execution.java;<<<<<<< MINE

```
   private final FieldCollector fieldCollector = new FieldCollector();
   private final ExecutionStrategy strategy;
```

||||||| BASE

```
   private FieldCollector fieldCollector = new FieldCollector();
   private ExecutionStrategy strategy;
```

=======

```
   private FieldCollector fieldCollector = new FieldCollector();
   private ExecutionStrategy queryStrategy;
```

```
   private ExecutionStrategy mutationStrategy;
>>>>>>> YOURS
```

Mudança: Right inclui mutatuionStrategy, Left, provavelmente, adicionou um espaço/enter, como vi em alguns casos nesses repositórios.
Categoria: aFP(UN)
Justificativa: Como se trata de declaração, não há problemas em mesclar as mudanças de left/right.
Como resolver: Adicionando os atributos extras logo após strategy.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_9d94784_8f1681d/rev_9d94784-8f1681d/src/main/java/graphql/execution/Execution.java;

```
<<<<<<< MINE
   public Execution(ExecutionStrategy executionStrategy) {
      this.strategy = executionStrategy == null ? new SimpleExecutionStrategy() :
executionStrategy;
||||||| BASE
   public Execution(ExecutionStrategy strategy) {
      this.strategy = strategy;

      if (this.strategy == null) {
         this.strategy = new SimpleExecutionStrategy();
      }
=======
   public Execution(ExecutionStrategy queryStrategy, ExecutionStrategy mutationStrategy) {
      this.queryStrategy = queryStrategy != null ? queryStrategy : new
SimpleExecutionStrategy();
      this.mutationStrategy = mutationStrategy != null ? mutationStrategy : new
SimpleExecutionStrategy();
>>>>>>> YOURS
```

Mudança: Ambos adicionam parâmetro ao construtor e código ao corpo dele.
Categoria: aFP(UN)
Justificativa: Não teremos problemas na correção do programa caso se opte por manter os dois construtores (Left e Right) e o código adicionado após eles.
Como resolver: Manter a assinatura dos dois construtores (de Left e Right), inicializar this.strategy como tá em Left e incluir a condição.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_9d94784_8f1681d/rev_9d94784-8f1681d/src/main/java/graphql/execution/Execution.java;

```
<<<<<<< MINE
      ExecutionContext executionContext = executionContextBuilder
            .executionId(executionId)
            .build(graphQLSchema, strategy, root, document, operationName, args);

||||||| BASE
```

```
        ExecutionContext executionContext = executionContextBuilder.build(graphQLSchema, strategy, root, document, operationName, args);
=======
        ExecutionContext executionContext = executionContextBuilder.build(graphQLSchema, queryStrategy, mutationStrategy, root, document, operationName, args);
>>>>>>> YOURS
```

Mudança: Ambos inicializam executionContext.
Categoria: aFN(SS)
Justificativa: Pode ser que ao unir as mudanças de Left e Right, isto é, inicializar executionContext chamando executionId e passando os argumentos do método build do right, surja problemas de compilação.
Como resolver: Não sei bem ao certo qual seria a melhor solução, talvez renomear ou optar por um dos casos.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_d4097e9_a339c65/rev_d4097e9-a339c65/src/main/java/graphql/GraphQL.java;

```
<<<<<<< MINE
    /**
     * A GraphQL object ready to execute queries
     *
     * @param graphQLSchema     the schema to use
     * @param executionStrategy the execution strategy to use
     *
     * @deprecated use the {@link #newObject(GraphQLSchema)} builder instead.  This will be removed in a future version.
     */
    @SuppressWarnings("DeprecatedIsStillUsed")
    public GraphQL(GraphQLSchema graphQLSchema, ExecutionStrategy executionStrategy) {
||||||| BASE
    public GraphQL(GraphQLSchema graphQLSchema, ExecutionStrategy executionStrategy) {
=======
    public GraphQL(GraphQLSchema graphQLSchema, ExecutionStrategy queryStrategy) {
        this(graphQLSchema, queryStrategy, null);
    }

    public GraphQL(GraphQLSchema graphQLSchema, ExecutionStrategy queryStrategy, ExecutionStrategy mutationStrategy) {
>>>>>>> YOURS
```

Mudança: Left adiciona comentários e um annotation, enquanto right adiciona mais um construtor.
Categoria: aFP(UN)

Justificativa: Podemos ter o annotation, comentário e novo construtor juntos na nova versão, sem problemas.
Como resolver: A mesclagem é feita sem muito estresse, apenas se atentando ao annotation.

/home/paper219/Desktop/analysis/demonstration/projects/graphql-java/revisions/rev_56eb2a7_b20ae0a/rev_56eb2a7-b20ae0a/src/main/java/graphql/execution/ExecutionContextBuilder.java;<<<<<<< MINE
```
   public ExecutionContextBuilder(ValuesResolver valuesResolver, Instrumentation instrumentation) {
||||||| BASE
   public ExecutionContextBuilder(ValuesResolver valuesResolver) {
=======
   private ExecutionId executionId;

   public ExecutionContextBuilder(ValuesResolver valuesResolver) {
>>>>>>> YOURS
```
Mudança: Left introduz mais um parâmetro ao construtor, enquanto right declara uma variável.
Categoria: aFP(UN)
Justificativa: Podemos ter, como resultado, os construtores juntos e mais a declaração de right, sem bronca, ou apenas um construtor com mais um parâmetro, se este for o caso.
Como resolver: Mantendo ambos construtores ou adicionando mais um parâmetro.

/home/paper219/Desktop/analysis/demonstration/projects/dagger/revisions/rev_1bc7c83_8f2e49e/rev_1bc7c83-8f2e49e/compiler/src/main/java/dagger/internal/codegen/InjectAdapterProcessor.java;<<<<<<< MINE
```
   if (injectable.getKind() == ElementKind.FIELD
       && injectable.getModifiers().contains(FINAL)) {
     error("Can't inject a final field: " + elementToString(injectable), injectable);
     return false;
   }

   if (injectable.getKind() == ElementKind.FIELD
       && injectable.getModifiers().contains(PRIVATE)) {
     error("Can't inject a private field: " + elementToString(injectable), injectable);
     return false;
   }

   if (injectable.getKind() == ElementKind.CONSTRUCTOR
       && injectable.getModifiers().contains(PRIVATE)) {
     error("Can't inject a private constructor: " + elementToString(injectable), injectable);
||||||| BASE
   if (injectable.getModifiers().contains(Modifier.PRIVATE)) {
```

```
      error("Can't inject a private field or constructor: " + injectableType + "." + injectable,
          injectable);
=======

    if (injectable.getKind() == ElementKind.CONSTRUCTOR
        && injectable.getModifiers().contains(PRIVATE)) {
      error("Can't inject a private constructor: " + elementToString(injectable), injectable);
>>>>>>> YOURS
```

Mudança: Ambos adicionam (injectable.getKind() == ElementKind.CONSTRUCTOR.
Categoria: aFP(UN)
Justificativa: A diferença entre ambos não é problema, já que um dá toString e o outro apenas concatena.
Como resolver: Resolver com o mine.

/home/paper219/Desktop/analysis/demonstration/projects/dagger/revisions/rev_1bc7c83_8f2e49e/rev_1bc7c83-8f2e49e/compiler/src/main/java/dagger/internal/codegen/InjectAdapterProcessor.java;

```
<<<<<<< MINE
      writeSupertypeInjectorField(writer, type, supertype);
||||||| BASE
      writer.emitField(JavaWriter.type(Binding.class,
          rawTypeToString(supertype, '.')), "supertype", PRIVATE);
=======
      writeSupertypeInjectorField(writer, supertype);
>>>>>>> YOURS
```

Mudança: Ambos os lados(Left e Right) fazem chamada do método writeSupertypeInjectorField.
Categoria: aFN(SS)
Justificativa: Como a junção, em um caso de SS, resultaria em Left(ou adiciona ambas as chamadas), pode ser que dê o resultado que Right não espera.
Como resolver: Não tenho como julgar a melhor alternativa aqui.

/home/paper219/Desktop/analysis/demonstration/projects/dagger/revisions/rev_1bc7c83_8f2e49e/rev_1bc7c83-8f2e49e/compiler/src/main/java/dagger/internal/codegen/InjectAdapterProcessor.java;

```
<<<<<<< MINE
          fieldName(disambiguateFields, field), EnumSet.of(PRIVATE));
    }
  }

  private void writeParameterBindingsFields(
      JavaWriter writer, ExecutableElement constructor, boolean disambiguateFields)
      throws IOException {
    for (VariableElement parameter : constructor.getParameters()) {
      writer.emitField(JavaWriter.type(Binding.class,
          typeToString(parameter.asType())),
          parameterName(disambiguateFields, parameter), EnumSet.of(PRIVATE));
```

||||||| BASE
        fieldName(false, field), PRIVATE);
=======
        fieldName(disambiguateFields, field), EnumSet.of(PRIVATE));
>>>>>>> YOURS

Mudança: Ambos mudam os argumentos de fielName e left introduz código
Categoria: aFP(UN)
Justificativa: Ambas mudanças fazem a mesma coisa quanto a fieldName e o código extra de left pode ser unido a right sem comprometer a correção do programa;
Como resolver: Resolve com Mine.

/home/paper219/Desktop/analysis/demonstration/projects/RxNetty/revisions/rev_8530e8d_549e21f/rev_8530e8d-549e21f/rxnetty-examples/src/test/java/io/reactivex/netty/examples/http/helloworld/HelloWorldTest.java;<<<<<<< MINE
 * Copyright 2014 Netflix, Inc.
=======
 * Copyright 2015 Netflix, Inc.
>>>>>>> YOURS
Mudança: Left e Right adicionam comentário
Categoria: aFP(UN)
Justificativa: Os comentários são os mesmos.

/home/paper219/Desktop/analysis/demonstration/projects/dagger/revisions/rev_1bc7c83_8f2e49e/rev_1bc7c83-8f2e49e/core/src/main/java/dagger/internal/FailoverLoader.java;<<<<<<< MINE
        injectedClass.getName() + STATIC_INJECTION_SUFFIX,
injectedClass.getClassLoader());
=======
        injectedClass.getName().concat(STATIC_INJECTION_SUFFIX),
injectedClass.getClassLoader());
>>>>>>> YOURS

Mudança: Ambos os lados estão concatenando injectedClass.getName() + STATIC_INJECTION_SUFFIX
Categoria: aFP(UN)
Justificativa: Ambas mudanças fazem a mesma coisa;
.

/home/paper219/Desktop/analysis/demonstration/projects/RxNetty/revisions/rev_f7e49e7_153d71b/rev_f7e49e7-153d71b/rxnetty/src/main/java/io/reactivex/netty/protocol/tcp/client/ConnectionProvider.java;<<<<<<< MINE
```
   public final Observable<Void> shutdownHook() {
       return shutdownHook;
|||||||BASE
public final Observable<Void> shutdownHook() {
return shutdownHook;
}
=======

   protected Observable<Void> doShutdown() {
       return Observable.empty();
>>>>>>> YOURS
```

Mudança: O Right deletou o método shutdownHook() e adicionou um novo.
Categoria: aFP(UN)
Justificativa: Só é preciso unir ambas as partes e não causará problemas.
Como resolver: Une sem se preocupar com a ordem.

/home/paper219/Desktop/analysis/demonstration/projects/Swagger-core/revisions/rev_e661e1a_9a6cf32/rev_e661e1a-9a6cf32/modules/swagger-core/src/main/java/io/swagger/util/ReflectionUtils.java;<<<<<<< MINE
```
       Annotation[][] methodAnnotations = method.getParameterAnnotations();
       Method overriddenmethod = getOverriddenMethod(method);
||||||| BASE
          Annotation[][] methodAnnotations = method.getParameterAnnotations();
          Method overriddenmethod = getOverriddenMethod(method);
=======
       Annotation[][] methodAnnotations = method.getParameterAnnotations();
       Method overriddenmethod = getOverriddenMethod(method);

       if (overriddenmethod != null) {
          Annotation[][] overriddenAnnotations = overriddenmethod
                .getParameterAnnotations();

          for (int i = 0; i < methodAnnotations.length; i++) {
             List<Type> types = new ArrayList<Type>();
             for (int j = 0; j < methodAnnotations[i].length; j++) {
                types.add(methodAnnotations[i][j].annotationType());
             }
             for (int j = 0; j < overriddenAnnotations[i].length; j++) {
```

```
                if (!types.contains(overriddenAnnotations[i][j]
                        .annotationType())) {
                    methodAnnotations[i] = ArrayUtils.add(
                            methodAnnotations[i],
                            overriddenAnnotations[i][j]);
                }
            }
>>>>>>> YOURS
```

Mudança: Right adicionou código em relação à base e ao Left.
Categoria: aFP(UN)
Justificativa: A mudança de left(provavelmente algum espaço/Enter) não causará problemas à right e vice-versa.
Como resolver: resolve com yours.

/home/paper219/Desktop/analysis/demonstration/projects/Swagger-core/revisions/rev_f3b170d_2dcdd25/rev_f3b170d-2dcdd25/modules/swagger-annotations/src/main/java/com/wordnik/swagger/annotations/Authorization.java;<<<<<<< MINE

```
  /**
   * The scopes to be used if the authorization scheme is OAuth2.
   *
   * @see com.wordnik.swagger.annotations.AuthorizationScope
   */
||||||| BASE
  String type();
=======

  /**
   * Not in use!
   */
  String type();

  /**
   * The scopes to be used if the authorization scheme is OAuth2.
   *
   * @see com.wordnik.swagger.annotations.AuthorizationScope
   */
>>>>>>> YOURS
```

Mudança: Left e Right adicionaram comentários.
Categoria: aFP(UN)
Justificativa: É possível mesclar ambas as partes já que os comentários não vão causar problemas.
Como resolver: resolve com yours.

/home/paper219/Desktop/analysis/demonstration/projects/Swagger-core/revisions/rev_e199bff_cc4e4f4/rev_e199bff-cc4e4f4/modules/swagger-models/src/main/java/io/swagger/models/properties/PropertyBuilder.java;<<<<<<< MINE
	FORMAT("format"),
	VENDOR_EXTENSIONS("vendorExtensions");
||||||| BASE
	FORMAT("format");
=======
	FORMAT("format"),
	READ_ONLY("readOnly");
>>>>>>> YOURS

Mudança: Left e Right fazem chamada de métodos a mais.
Categoria: aFN(SS)
Justificativa: Pode ser que unindo as chamadas, dê resultados indesejados, logo, sendo conservadores, atribuímos como aFN(SS).
Como resolver: Não tenho certeza de como resolver.