```
@startuml FJ-lam
'Show access modifiers as characters
skinparam classAttributeIconSize 0

'page 2x2

'Number of pixels of the resulting image
'scale 2000 width

'Group inheritance arrow heads
'skinparam groupInheritance 2

'Make diagram top to bottom
'left to right direction

package imt.fil.a3.recherche.fj.model {
  class TypeCheckingContext <<record>> {
    + {static} Logger logger
    + TypeTable typeTable
    - Map<String, String> context
    + Optional<String> typeName(String variableName)
    + TypeCheckingContext copy()
    + TypeCheckingContext with(List<FJField> fields)
    + void add(List<FJField> fields)
    + TypeCheckingContext with(String variableName, String variableType)
    + void add(String variableName, String variableType)
  }

  class TypeTable <<record>> {
  }
}

package imt.fil.a3.recherche.fj.model.error {
  abstract class TypeError {
    + String getMessage()
  }

  class ArgsTypesMismatch {
    + List<String> expected
    + List<String> actual
  }
  ArgsTypesMismatch -down-|> TypeError

  class ArgTypeMismatch {
    + String expected
    + String actual
  }
  ArgTypeMismatch -down-|> TypeError

  class ClassNotFound {
    + String name
  }
```

```
    ClassNotFound -down-|> TypeError

    class FieldNotFound {
      + String name
    }
    FieldNotFound -right-|> TypeError

    class MethodNotFound {
      + String name
      + String returnTypeName
    }
    MethodNotFound -up-|> TypeError

    class VariableNotFound {
      + String name
    }
    VariableNotFound -up-|> TypeError

    class WrongCast {
      + String castType
      + FJExpr expression
    }
    WrongCast -up-|> TypeError

    class WrongLambdaType {
      + String targetTypeName
      + FJExpr lambda
    }
    WrongLambdaType -left-|> TypeError
  }

  package imt.fil.a3.recherche.fj.model.java.expression {
    interface FJExpr {
      + TypeAnnotatedExpression getTypeApproach1(\n\tTypeCheckingContext
context\n) throws TypeError
      + String getTypeNameApproach2(\n\tTypeCheckingContext context\n)
throws TypeError
      + default FJExpr lambdaMark(String typeName)
      + FJExpr removingRuntimeAnnotation()
      + default FJExpr evalApproach2(TypeTable typeTable)
      + Boolean isValue()
      + Optional<FJExpr> _evalApproach2(TypeTable typeTable)
      + Optional<FJExpr> substituteApproach2(\n\tList<String>
parameterNames,\n\tList<FJExpr> args\n)
      + default Optional<FJExpr> evalMethodInvocationApproach2(\n\tTypeTable
typeTable, \n\tFJMethodInvocation invocation\n)
    }

    class FJCast implements FJExpr {
      + String typeName
      + FJExpr body
    }
    class FJCast <<record>>
    class FJCreateObject implements FJExpr {
      + String className
      + List<FJExpr> args
    }
    class FJCreateObject <<record>>

    class FJFieldAccess implements FJExpr {
      + FJExpr object
      + String fieldName
    }
    class FJFieldAccess <<record>>

    class FJLambda implements FJExpr {
      + List<FJField> args
      + FJExpr body
      + TypeAnnotatedExpression getTypeApproach1(\n\tTypeCheckingContext
  context,\n\tString returnTypeName\n) throws TypeError
      + String getTypeNameApproach2(\n\tTypeCheckingContext
  context,\n\tString returnTypeName\n) throws TypeError
    }
    class FJLambda <<record>>

    class FJMethodInvocation implements FJExpr {
      + FJExpr source
      + String methodName
      + List<FJExpr> args
    }
    class FJMethodInvocation <<record>>

    class FJVariable implements FJExpr {
      + String name
    }
    class FJVariable <<record>>
  }

  package imt.fil.a3.recherche.fj.model.java.misc {
    class FJConstructor {
      + String name
      + List<FJField> args
      + List<String> superArgs
      + List<FieldInit> fieldInits
    }
    class FJConstructor <<record>>

    class FJField {
      + String type
      + String name
    }
    class FJField <<record>>

    class FJMethod {
      + FJSignature signature
      + FJExpr body
      + Optional<FJMethod> typeCheckApproach1(\n\tTypeCheckingContext
  context,\n\tString className\n)
      + Boolean typeCheckApproach2(\n\tTypeCheckingContext
  context,\n\tString className\n)
      + Boolean signatureEquals(FJMethod other)
    }
    class FJMethod <<record>>

    class FJProgram {
      + List<FJType> types
      + Boolean typeCheckApproach2(TypeCheckingContext context)
      + TypeTable getTypeTable()
    }
    class FJProgram <<record>>

    class FJSignature {
      + String returnTypeName
      + String name
      + List<FJField> args
      + MethodTypeSignature getTypeSignature()
    }
    class FJSignature <<record>>
  }

  package imt.fil.a3.recherche.fj.model.java.type {
    interface FJType {
      + Optional<? extends FJType> typeCheckApproach1(TypeCheckingContext
  context)
      + Boolean typeCheckApproach2(TypeCheckingContext context)
      + Boolean isSubtype(TypeTable typeTable, String otherTypeName)
      + default Optional<List<FJField>> classFields(TypeTable typeTable)
      + Optional<List<FJSignature>> abstractMethods(TypeTable typeTable)
      + Optional<List<FJMethod>> methods(TypeTable typeTable)
    }

    class FJClass implements FJType {
      + String name
      + String extendsName
      + List<String> implementsNames
      + List<FJField> fields
      + List<FJMethod> methods
      + FJConstructor constructor
    }
    class FJClass <<record>>

    class FJInterface implements FJType {
      + String name
      + List<String> extendsNames
      + List<FJSignature> signatures
      + List<FJMethod> defaultMethods
    }
    class FJInterface <<record>>
  }

  package imt.fil.a3.recherche.fj.model.misc {
    class FieldInit <<record>> {
      + String fieldName
      + String argumentName
    }
    class MethodBodySignature <<record>> {
      + List<String> argumentNames
      + FJExpr body
    }
    class MethodTypeSignature <<record>> {
      + List<String> parameterTypeNames
      + String returnTypeName
    }
    class TypeAnnotatedExpression <<record>> {
      + String typeName
      + FJExpr expression
    }
    class TypeMismatch <<record>> {
      + FJExpr expression
      + String expectedTypeName
    }
  }

  package imt.fil.a3.recherche.fj.util.haskell {
    class Haskell {
      + {static} <E> List<E> union(List<E> a, List<E> b)
      + {static} <E> List<E> union(List<E> a, List<E> b, BiPredicate<E, E>
  predicate)
      + {static} <E> List<E> difference(List<E> a, List<E> b)
    }
  }

  'Relationships
  FJCast --> FJExpr
  FJCreateObject "0..*" o--> FJExpr
  FJFieldAccess --> FJExpr
  FJLambda --> FJExpr
  FJLambda "0..*" o--> FJField
  FJMethodInvocation --> FJExpr
  FJMethodInvocation "0..*" o--> FJExpr

  MethodBodySignature --> FJExpr
  TypeAnnotatedExpression --> FJExpr
  TypeMismatch --> FJExpr

  TypeCheckingContext --> TypeTable

  FJClass "0..*" o--> FJField
  FJClass "0..*" o--> FJMethod
  FJClass --> FJConstructor
  FJInterface "0..*" o--> FJSignature
  FJInterface "0..*" o--> FJMethod

  FJProgram "0..*" o--> FJType
  FJConstructor "0..*" o--> FJField
  FJConstructor "0..*" o--> FieldInit
  FJSignature "0..*" o--> FJField
  FJMethod "0..*" o--> FJSignature
  FJMethod --> FJExpr

  'Placement
  imt.fil.a3.recherche.fj.model.error -up[hidden]->
  imt.fil.a3.recherche.fj.model
  imt.fil.a3.recherche.fj.model.misc -up[hidden]->
  imt.fil.a3.recherche.fj.model
  imt.fil.a3.recherche.fj.util.haskell -up[hidden]->
  imt.fil.a3.recherche.fj.model
  @enduml```
```