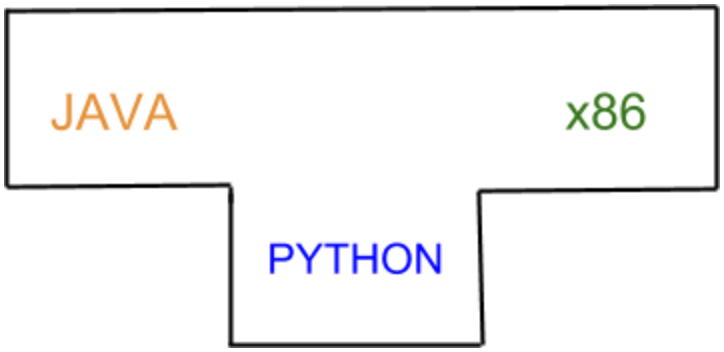
COMPILERS ASSIGNMENT 0

Group Members:

1. Abhisek Panda - 150026 - CSE - apanda@iitk.ac.in
2. Raktim Mitra - 150562 - CSE - raktim@iitk.ac.in
3. Sanket - 150634 - CSE - sanketyd@iitk.ac.in

T-Diagram:



Tools Used for Implementation:

For lexing and parsing, we will use **ply**.
For debugging, we will use **pdb**.
For visualization, we will use **pydot**.

EBNF for source language(JAVA):

The grammar below uses the following BNF - style conventions:

[x] denotes zero or one occurrences of x.
{x} denotes zero or more occurrences of x.
(x | y) means one of either x or y.

NOTE:

- ➔ All features highlighted in RED have been removed from our implementation.
- ➔ Features highlighted in BLUE have been added to enhance our language. This includes the introduction of “lambda” .

Identifier:
    IDENTIFIER

QualifiedIdentifier:
    Identifier { . Identifier }

QualifiedIdentifierList:
    QualifiedIdentifier { , QualifiedIdentifier }

CompilationUnit:
    [[Annotations] package QualifiedIdentifier ;]
        {ImportDeclaration} {TypeDeclaration}

ImportDeclaration:
    import [static] Identifier { . Identifier } [. *] ;

TypeDeclaration:
    ClassOrInterfaceDeclaration
    ;

ClassOrInterfaceDeclaration:
    {Modifier} (ClassDeclaration | InterfaceDeclaration)

ClassDeclaration:
    NormalClassDeclaration
    EnumDeclaration

InterfaceDeclaration:
    NormalInterfaceDeclaration
    AnnotationTypeDeclaration

NormalClassDeclaration:
    class Identifier [TypeParameters]
        [extends Type] [implements TypeList] ClassBody

EnumDeclaration:
    enum Identifier [implements TypeList] EnumBody

NormalInterfaceDeclaration:
    interface Identifier [TypeParameters] [extends TypeList] InterfaceBody

AnnotationTypeDeclaration:
    @ interface Identifier AnnotationTypeBody

Type:
    BasicType {[]}
    ReferenceType {[]}

BasicType:
    byte
    short
    char
    int
    long
    float
    double
    boolean

ReferenceType:
    Identifier [TypeArguments] { . Identifier [TypeArguments] }

TypeArguments:
    < TypeArgument { , TypeArgument } >

TypeArgument:
    ReferenceType
    ? [(extends | super) ReferenceType]

NonWildcardTypeArguments:
    < TypeList >

TypeList:
    ReferenceType { , ReferenceType }

TypeArgumentsOrDiamond:
    < >
    TypeArguments

NonWildcardTypeArgumentsOrDiamond:
    < >
    NonWildcardTypeArguments

TypeParameters:
    < TypeParameter { , TypeParameter } >

TypeParameter:
    Identifier [extends Bound]

Bound:
    ReferenceType { & ReferenceType }

Modifier:
    Annotation
    public
    protected
    private
    static
    abstract
    final
    native
    synchronized
    transient
    volatile
    strictfp

Annotations:
    Annotation {Annotation}

Annotation:
    @ QualifiedIdentifier [([AnnotationElement])]

AnnotationElement:
    ElementValuePairs
    ElementValue

ElementValuePairs:
    ElementValuePair { , ElementValuePair }

ElementValuePair:
    Identifier = ElementValue

ElementValue:
    Annotation
    Expression1
    ElementValueArrayInitializer

ElementValueArrayInitializer:
    { [ElementValues] [,] }

ElementValues:
    ElementValue { , ElementValue }

ClassBody:
    { { ClassBodyDeclaration } }

ClassBodyDeclaration:
    ;
    {Modifier} MemberDecl
    [static] Block

MemberDecl:
    MethodOrFieldDecl
    void Identifier VoidMethodDeclaratorRest
    Identifier ConstructorDeclaratorRest
    GenericMethodOrConstructorDecl
    ClassDeclaration
    InterfaceDeclaration

MethodOrFieldDecl:
    Type Identifier MethodOrFieldRest

MethodOrFieldRest:
    FieldDeclaratorsRest ;
    MethodDeclaratorRest

FieldDeclaratorsRest:
    VariableDeclaratorRest { , VariableDeclarator }

MethodDeclaratorRest:
    FormalParameters {[]} [throws QualifiedIdentifierList] (Block | ;)

VoidMethodDeclaratorRest:
    FormalParameters [throws QualifiedIdentifierList] (Block | ;)

ConstructorDeclaratorRest:
FormalParameters [throws QualifiedIdentifierList] Block

GenericMethodOrConstructorDecl:
TypeParameters GenericMethodOrConstructorRest

GenericMethodOrConstructorRest:
(Type | void) Identifier MethodDeclaratorRest
Identifier ConstructorDeclaratorRest

InterfaceBody:
{ { InterfaceBodyDeclaration } }

InterfaceBodyDeclaration:
;
{Modifier} InterfaceMemberDecl

InterfaceMemberDecl:
InterfaceMethodOrFieldDecl
void Identifier VoidInterfaceMethodDeclaratorRest
InterfaceGenericMethodDecl
ClassDeclaration
InterfaceDeclaration

InterfaceMethodOrFieldDecl:
Type Identifier InterfaceMethodOrFieldRest

InterfaceMethodOrFieldRest:
ConstantDeclaratorsRest ;
InterfaceMethodDeclaratorRest

ConstantDeclaratorsRest:
ConstantDeclaratorRest { , ConstantDeclarator }

ConstantDeclaratorRest:
{[]} = VariableInitializer

ConstantDeclarator:
Identifier ConstantDeclaratorRest

InterfaceMethodDeclaratorRest:
    FormalParameters {[]} [throws QualifiedIdentifierList] ;

VoidInterfaceMethodDeclaratorRest:
    FormalParameters [throws QualifiedIdentifierList] ;

InterfaceGenericMethodDecl:
    TypeParameters (Type | void) Identifier InterfaceMethodDeclaratorRest

FormalParameters:
    ([FormalParameterDecls])

FormalParameterDecls:
    {VariableModifier} Type FormalParameterDeclsRest

VariableModifier:
    final
    Annotation

FormalParameterDeclsRest:
    VariableDeclaratorId [, FormalParameterDecls]
    ... VariableDeclaratorId

VariableDeclaratorId:
    Identifier {[]}

VariableDeclarators:
    VariableDeclarator { , VariableDeclarator }

VariableDeclarator:
    Identifier VariableDeclaratorRest

VariableDeclaratorRest:
    {[]} [= VariableInitializer]

VariableInitializer:
    ArrayInitializer
    Expression

ArrayInitializer:
    { [VariableInitializer { , VariableInitializer } [,]] }

Block:
{ BlockStatements }

BlockStatements:
{ BlockStatement }

BlockStatement:
LocalVariableDeclarationStatement
ClassOrInterfaceDeclaration
[Identifier :] Statement

LocalVariableDeclarationStatement:
{ VariableModifier } Type VariableDeclarators ;

Statement:
Block
;

Identifier : Statement
StatementExpression ;
if ParExpression Statement [else Statement]
assert Expression [: Expression] ;
switch ParExpression { SwitchBlockStatementGroups }
while ParExpression Statement
do Statement while ParExpression ;
for (ForControl) Statement
break [Identifier] ;
continue [Identifier] ;
return [Expression] ;
throw Expression ;

synchronized ParExpression Block
try Block (Catches | [Catches] Finally)
try ResourceSpecification Block [Catches] [Finally]

StatementExpression:
Expression

Catches:
CatchClause { CatchClause }

CatchClause:
    catch ({VariableModifier} CatchType Identifier) Block

CatchType:
    QualifiedIdentifier { | QualifiedIdentifier }

Finally:
    finally Block

ResourceSpecification:
    (Resources [;])

Resources:
    Resource { ; Resource }

Resource:
    {VariableModifier} ReferenceType VariableDeclaratorId = Expression

SwitchBlockStatementGroups:
    { SwitchBlockStatementGroup }

SwitchBlockStatementGroup:
    SwitchLabels BlockStatements

SwitchLabels:
    SwitchLabel { SwitchLabel }

SwitchLabel:
    case Expression :
    case EnumConstantName :
    default :

EnumConstantName:
    Identifier

ForControl:
    ForVarControl
    ForInit ; [Expression] ; [ForUpdate]

ForVarControl:
    {VariableModifier} Type VariableDeclaratorId ForVarControlRest

ForVarControlRest:
    ForVariableDeclaratorsRest ; [Expression] ; [ForUpdate]
    : Expression

ForVariableDeclaratorsRest:
    [= VariableInitializer] { , VariableDeclarator }

ForInit:
ForUpdate:
    StatementExpression { , StatementExpression }

Expression:
    Expression1 [AssignmentOperator Expression1]
    LambdaExpression

LambdaExpression:
    LambdaParameters -> LambdaBody

LambdaParameters:
    Identifier
    ([FormalParameters])
    (QualifiedIdentifierList)

LambdaBody:
    Expression
    Block

AssignmentOperator:
    =
    +=
    -=
    *=
    /=
    &=
    |=
    ^=
    %=
    <<=
    >>=
    >>>=

Expression1:
    Expression2 [Expression1Rest]

Expression1Rest:
    ? Expression : Expression1

Expression2:
    Expression3 [Expression2Rest]

Expression2Rest:
    { InfixOp Expression3 }
    instanceof Type

InfixOp:
    ||
    &&
    |
    ^
    &
    ==
    !=
    <
    >
    <=
    >=
    <<
    >>
    >>>
    +
    -
    *
    /
    %

Expression3:
    PrefixOp Expression3
    ((Expression | Type)) Expression3
    Primary { Selector } { PostfixOp }

PrefixOp:

++
--
!
~
+
-

PostfixOp:

++
--

Primary:

Literal
ParExpression
this [Arguments]
super SuperSuffix
new Creator
NonWildcardTypeArguments (ExplicitGenericInvocationSuffix | this Arguments)
Identifier { . Identifier } [IdentifierSuffix]
BasicType {[]} . class
void . class

Literal:

IntegerLiteral
FloatingPointLiteral
CharacterLiteral
StringLiteral
BooleanLiteral
NullLiteral

ParExpression:

(Expression)

Arguments:

([Expression { , Expression }])

SuperSuffix:
    Arguments
    . Identifier [Arguments]

ExplicitGenericInvocationSuffix:
    super SuperSuffix
    Identifier Arguments

Creator:
    NonWildcardTypeArguments CreatedName ClassCreatorRest
    CreatedName (ClassCreatorRest | ArrayCreatorRest)

CreatedName:
    Identifier [TypeArgumentsOrDiamond] { . Identifier [TypeArgumentsOrDiamond] }

ClassCreatorRest:
    Arguments [ClassBody]

ArrayCreatorRest:
    [(] {[]} ArrayInitializer | Expression] {[Expression]} {[]})

IdentifierSuffix:
    [({[]} . class | Expression)]
    Arguments
    . (class | ExplicitGenericInvocation | this | super Arguments |
                new [NonWildcardTypeArguments] InnerCreator)

ExplicitGenericInvocation:
    NonWildcardTypeArguments ExplicitGenericInvocationSuffix

InnerCreator:
    Identifier [NonWildcardTypeArgumentsOrDiamond] ClassCreatorRest

Selector:
    . Identifier [Arguments]
    . ExplicitGenericInvocation
    . this
    . super SuperSuffix
    . new [NonWildcardTypeArguments] InnerCreator
    [Expression]

EnumBody:
{ [EnumConstants] [,] [EnumBodyDeclarations] }

EnumConstants:
EnumConstant
EnumConstants , EnumConstant

EnumConstant:
[Annotations] Identifier [Arguments] [ClassBody]

EnumBodyDeclarations:
; {ClassBodyDeclaration}

AnnotationTypeBody:
{ [AnnotationTypeElementDeclarations] }

AnnotationTypeElementDeclarations:
AnnotationTypeElementDeclaration
AnnotationTypeElementDeclarations AnnotationTypeElementDeclaration

AnnotationTypeElementDeclaration:
{Modifier} AnnotationTypeElementRest

AnnotationTypeElementRest:
Type Identifier AnnotationMethodOrConstantRest ;
ClassDeclaration
InterfaceDeclaration
EnumDeclaration
AnnotationTypeDeclaration

AnnotationMethodOrConstantRest:
AnnotationMethodRest
ConstantDeclaratorsRest

AnnotationMethodRest:
( ) [[]] [default ElementValue]