

## Plan for Today

### PA6: Stack frame (Variable placement)

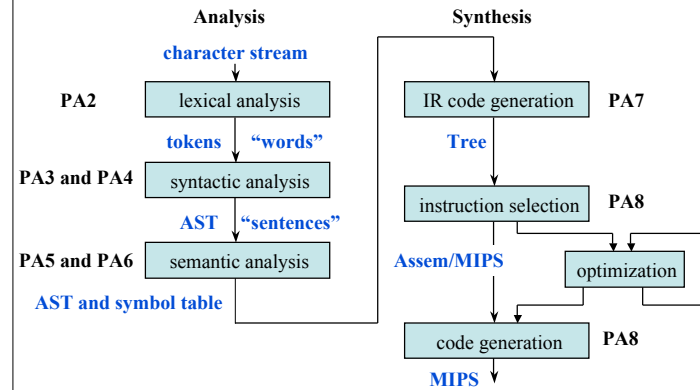
- testing and grading
- the Frame class
- the ClassSTE class
- the MethodSTE class
- the Temp package

CS453 Lecture

MiniJava Compiler - Frame class

1

## Structure of the MiniJava Compiler

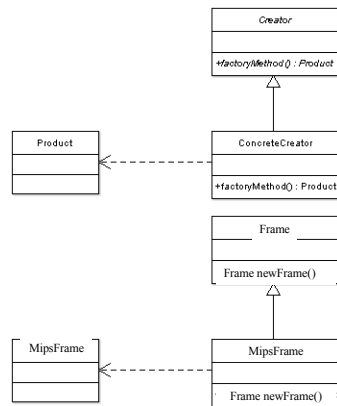


CS453 Lecture

MiniJava Compiler - Frame class

2

## Frame class implements Factory design pattern



CS453 Lecture

MiniJava Compiler - Frame class

3

## What, why, and where?

### What?

- one instance of the MipsFrame will generate other instances

### Why?

- need a Frame instance for each function and want to avoid calling the MipsFrame constructor everywhere

### Where?

```
// FrameLayout.java
Frame.Frame frame = new Mips.MipsFrame();
BuildSymTable buildSTvisitor = new BuildSymTable(linesToNodes, frame);

// BuildSymTable::inAMethodDecl
...
Frame.Frame methodFrame = mFrame.newFrame(
    new Temp.Label(mCurrentClass.getName()+"$" + node.getName().getText()),
    formalEscapeList);
```

CS453 Lecture

MiniJava Compiler - Frame class

4

## Interface to Frame

### Three main responsibilities

- provide a factory interface for generating machine-specific frames
  - Frame newFrame(Label name, List<Boolean> formals)
- answer queries that are machine-specific, but not method specific
  - int wordSize()
  - Temp FP(), coming to an interface near you in PA7
- store method-specific information about frame layout
  - Label name
  - List<Access> formals
  - Access allocLocal(boolean escape)

### Access class

```
public abstract class Access {
    public abstract String toString();
    //public abstract Tree.Exp exp(Tree.Exp e);
}
```

CS453 Lecture

MiniJava Compiler - Frame class

5

## When do parameters and/or locals escape?

### Nesting of classes and methods

```
int foo(int x)
{
    int baz(int y)
    {
        return x+y;
    }
    int bar() {
        return baz(2);
    }
    return bar();
}
int main() {
    printf("%f\n", foo(3));
}
```

### When the variable may have its address taken

```
int addressFunc(int x)
{
    int z;
    return blah(&x, &z);
}
```

### When the language uses pass-by-reference

```
FORTRAN
subroutine
foo(x,y)
double precision x
double precision y
y=x*2
end subroutine
```

```
subroutine head()
double precision a, b
call foo(a,b)
end subroutine
```

CS453 Lecture

MiniJava Compiler - Frame class

6