Plan for Today

PA6: Stack frame (Variable placement)
- testing and grading
- the Frame class
- the ClassSTE class
- the MethodSTE class
- the Temp package

Frame class implements Factory design pattern

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);}
**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**

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

---

**When do parameters and/or locals escape?**

**Nesting of classes and methods**

When the variable may have its address taken

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

When the language uses pass-by-reference

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

```java
int foo(int x) {
    int bar(int y) {
        return x+y;
    }
    int bar() {
        return bar(2);
    }
    return bar();
}
```

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

```java
int main() {
    printf("%f\n", foo(3));
}
```