CSE 142

Scope

Introduction to Scope

- The *scope* of a name is the region of the program in which that name is defined, or bound. (A scope corresponds to a piece of scratch paper.)
- Method definitions define a scope: names defined *inside* of a method (either parameters or local names) are defined *only* within the body of that method.
- Same with constructors.
- Jeva bindings are in their own scope.
- Scoping is good: it frees programmers from worrying about name collisions. Temporary scratch names can be used freely.
- Ref: Niño & Hosch: sec. 14.7

Scope in Pictures

```
public class House {
    private Rectangle frame; ...
    public House( ) {
        int leftX = 100;
        int topY = 75;
        this.frame = new Rectangle(leftX, topY, ...);
        ...
    }
    public House(int leftX, int bottomY, int width, int height, ...) {
        int topY = bottomY – height;
        this.frame = new Rectangle(leftX, topY, ...);
        ...
    }
}
```

- No confusion about “leftX” or “bottomY”

Nesting Scopes

- Scopes can be *nested*, one within another. A nested scope can refer to names declare in itself *or in any enclosing scope*. The outer scope cannot refer to names in one of its nested scopes.
- A class definition defines a scope, with method definition scopes nested inside it.
  - Instance variables and methods are names that are bound in the class scope. They can be referenced directly by any nested scope, without using "this." !

Nested Scopes in Pictures

```
public class House {
    private Rectangle frame; ...
    public House( ) {
        int leftX = 100;
        int topY = 75;
        frame = new Rectangle(leftX, topY, ...);
        ...
    }
    public House(int leftX, int bottomY, int width, int height, ...) {
        int topY = bottomY – height;
        frame = new Rectangle(leftX, topY, ...);
        ...
    }
}
```

- "frame" means the same as "this.frame" because of nested scoping

Navigating Nested Scopes

- *Name resolution*: figuring out which declaration a name at a given spot in the program refers to.
- Proceed as follows:
  - Start in the scope where the name is referenced.
    (Exception: if name is preceded by “this.”, start in class scope)
  - If if a declaration is found there, then use that one.
  - Otherwise go to the next enclosing scope, and repeat the search.
  - If run out of scopes, search the imported packages (just for class names).
  - If still can't find it, report a "unresolved symbol" error.

A Buggy Program

- What's wrong with this constructor?

```
public class House {
    private Rectangle frame; ...

    public House( ) {
        int leftX = 100;
        int topY = 75;
        Rectangle frame = new Rectangle(leftX, topY, ...);
        ...
    }
}
```


Using "this" to Resolve Scopes

```
public class BankAccount {
    private double balance;
    private String accountName;
    public BankAccount(double balance, String accountName) {
        this.balance = balance;
        this.accountName = accountName;
        ...
    }
}
```

- Question: What happens if we omit "this." in the assignment statements?

Nested Scopes in { ... }

- In an if statement, the true and false statement blocks { ... } each introduce their own nested scopes

```
void askForBeer(Person customer) {
    if (customer.getApparentAge( ) >= 30) {
        int numberOfBeers = ...;
        ...
    } else if (customer.getAgeOnID( ) >= 21) {
        int numberOfBeers = ...;
        ...
    } else {
        int policePhoneNumber = ...;
        ...
    }
}
```


Computing Results from If Statements

- Let's say we wanted to create a name recording whether or not the customer was allowed to buy beer.
- What type of value will we compute?

Computing Results from If Statements

- Let's try to do it this way:

```
if (customer.getApparentAge( ) >= 30 || customer.getAgeOnID( ) >= 21) {
    boolean canBuyBeer = true;
} else {
    boolean canBuyBeer = false;
}

if (canBuyBeer) {
    sellBeer( );                    // or this.sellBeer( );  etc.
} else {
    callCops( );
}
```

- It doesn't work: "symbol canBuyBeer unresolved". What's wrong?

Fixed Program

- Solution: declare name in enclosing scope

```
boolean canBuyBeer;
if (customer.getApparentAge( ) >= 30 || customer.getAgeOnID( ) >= 21) {
    canBuyBeer = true;   // initialization, not declaration (no "boolean")
} else {
    canBuyBeer = false;   // initialization, not declaration (no "boolean")
}

if (canBuyBeer) {
    sellBeer( );
} else {
    callCops( );
}
```