CS100J 11 September 2003

The "at" sign, @, was made famous by Ray Tomlinson, a researcher at BBN in Boston. In 1971, he selected @ as the separator between an email name and location.
Here are names for @ in other languages:

Italian: *chiocciolina* = little snail
French: *petit escargot* = little snail
German: *klammeraffe* = spider monkey
Dutch: *api* = short for apestaart (monkey's tail)
Norwegian: *kanel-bolle* = spiral-shaped cinnamon cake
Danish: *snabel* = an "A" with a trunk
Israeli: *strudel* = a pastry
Finnish: *miau* = cat tail
Spanish: *un arroba* = a unit of about 25 pounds

For more info: **http://www.mailmsg.com/history.htm**

How do you deal with the partner frame in FramingFrame?

Explain through another example.

```
public class Person {
    Date birthdate;
    String name;
    Address address;



}
```

A person has a mother. Suppose we want to include in each manilla folder of class Person the name on the folder of their mother. How do we do it?

How do you deal with the partner frame in FramingFrame?

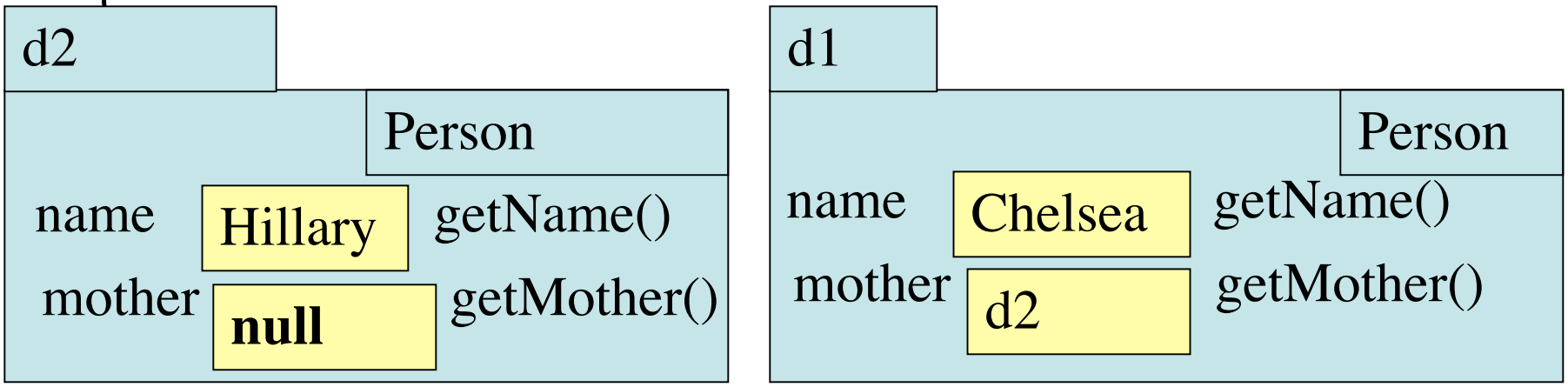
Explain through another example.

```
public class Person {
      private Date birthdate;
      private String name;
      private Address address;
      private Person mother;

     // = name of this person's mother
     public Person getMother() {
         return mother;
     }
}
```

A person has a mother. Suppose we want to include in each manilla folder of class Person the name on the folder of their mother. Here's how we do it.

```
public class Person {
    private String name;
    private Person mother;

   // = name of this person's mother
   public Person getMother() { return mother; }
   // = name of this person's name
   public String getName() { return name; }
}
```

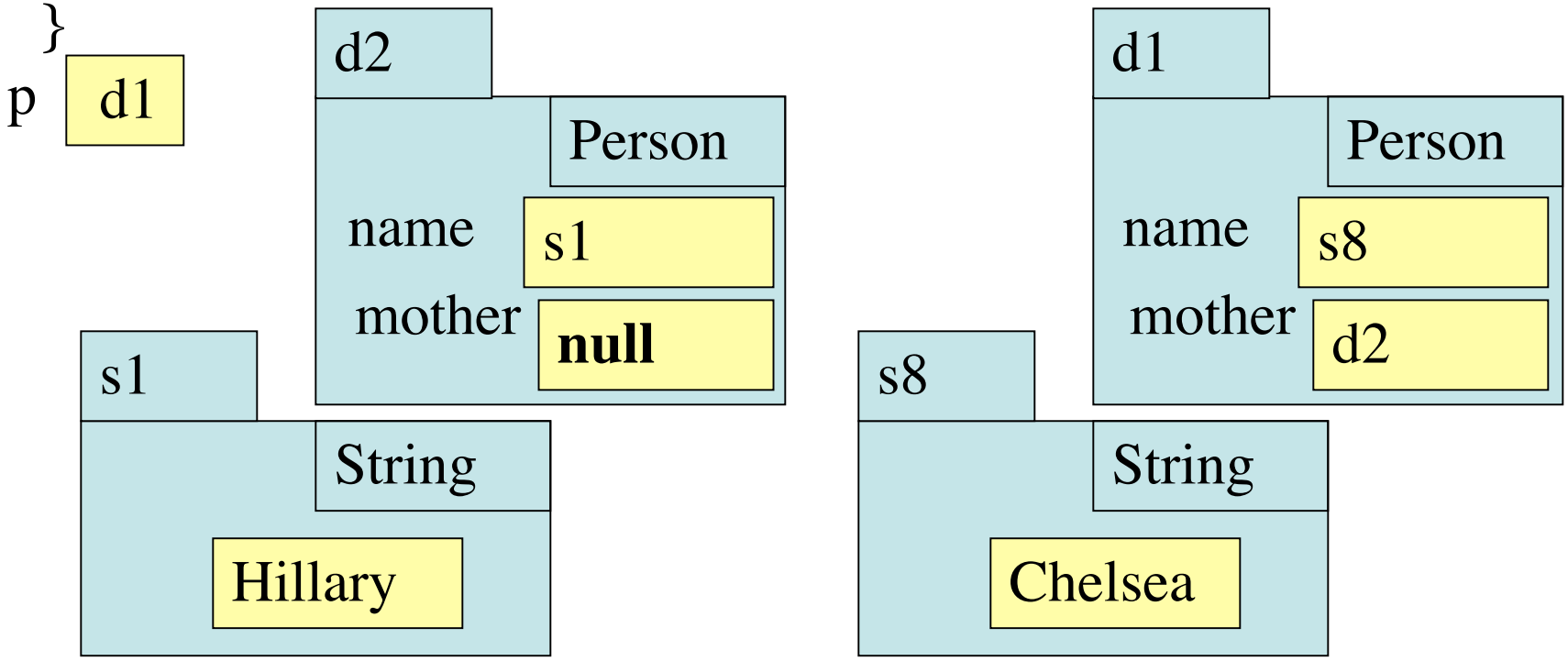
p d1



To get the name of p's mother:

p.getMother().getName()

```
public class Person {
    private String name;
    private Person mother;

   // = name of this person's mother
   public Person getMother() { return mother; }
   // = name of this person's name
   public String getName() { return name; }
}
```



A Person has a mother.
Therefore, a folder for a Person has a field, mother.
Since a mother is a Person, the type of the field is Person. If the mother is not known, this field contains **null.**

A folder of class FramingFrame is supposed to have an associated window, a JFrame, which surrounds it.
Therefore, …

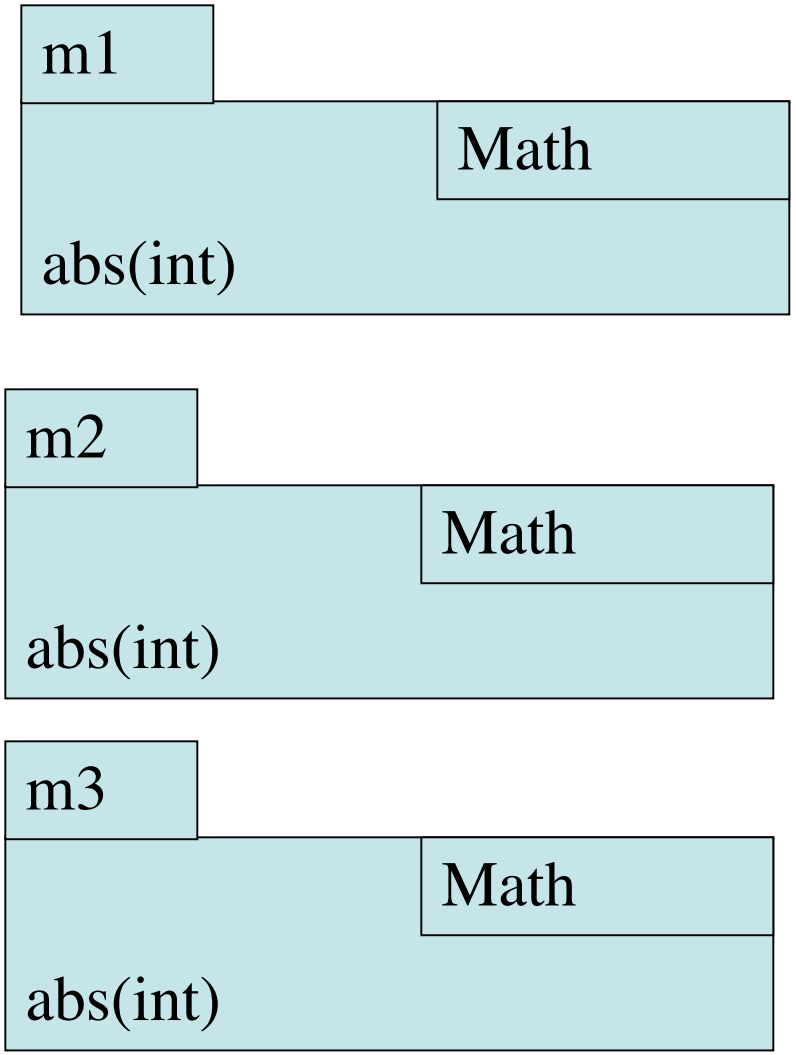
Static variables

```
public class Math {
    /** = the absolute value of x
    public int abs(int x) {
        if (x >= 0) return x;
        return – x;
    }
```

Function abs belongs in every folder of class Math

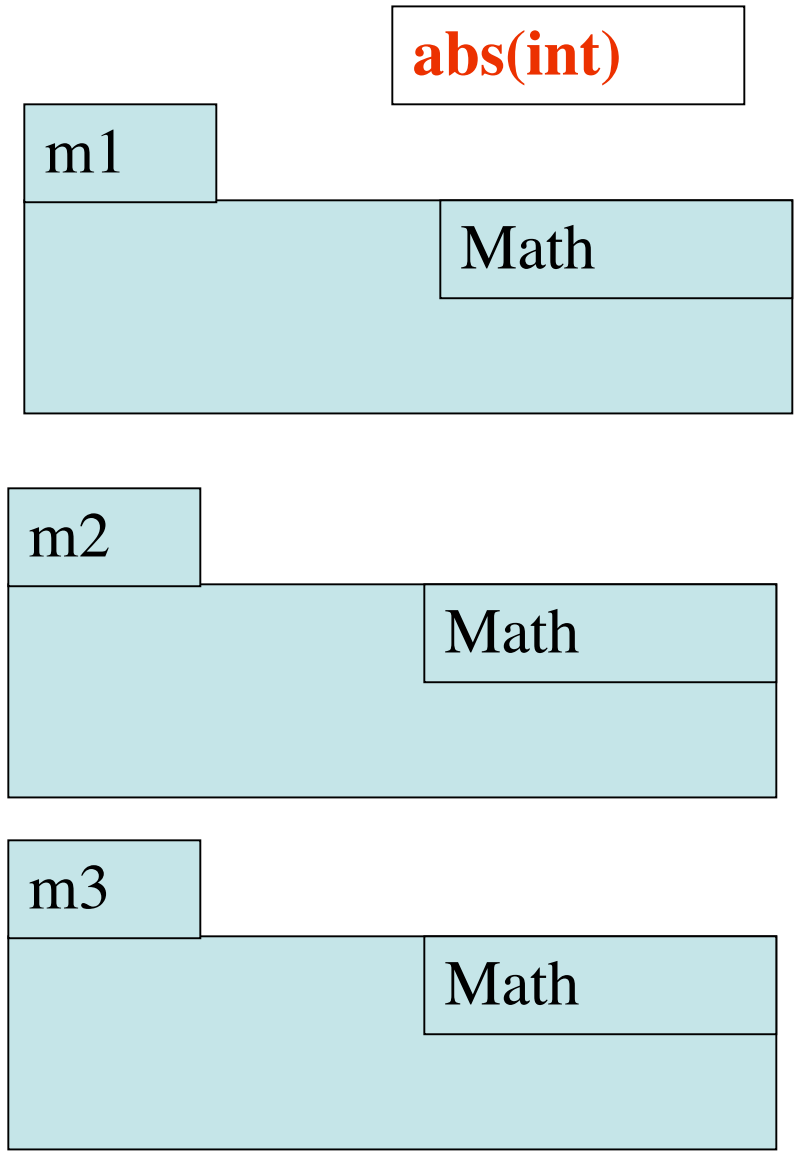
Need only ONE function abs

```
}
```



Four things in file drawer Math:

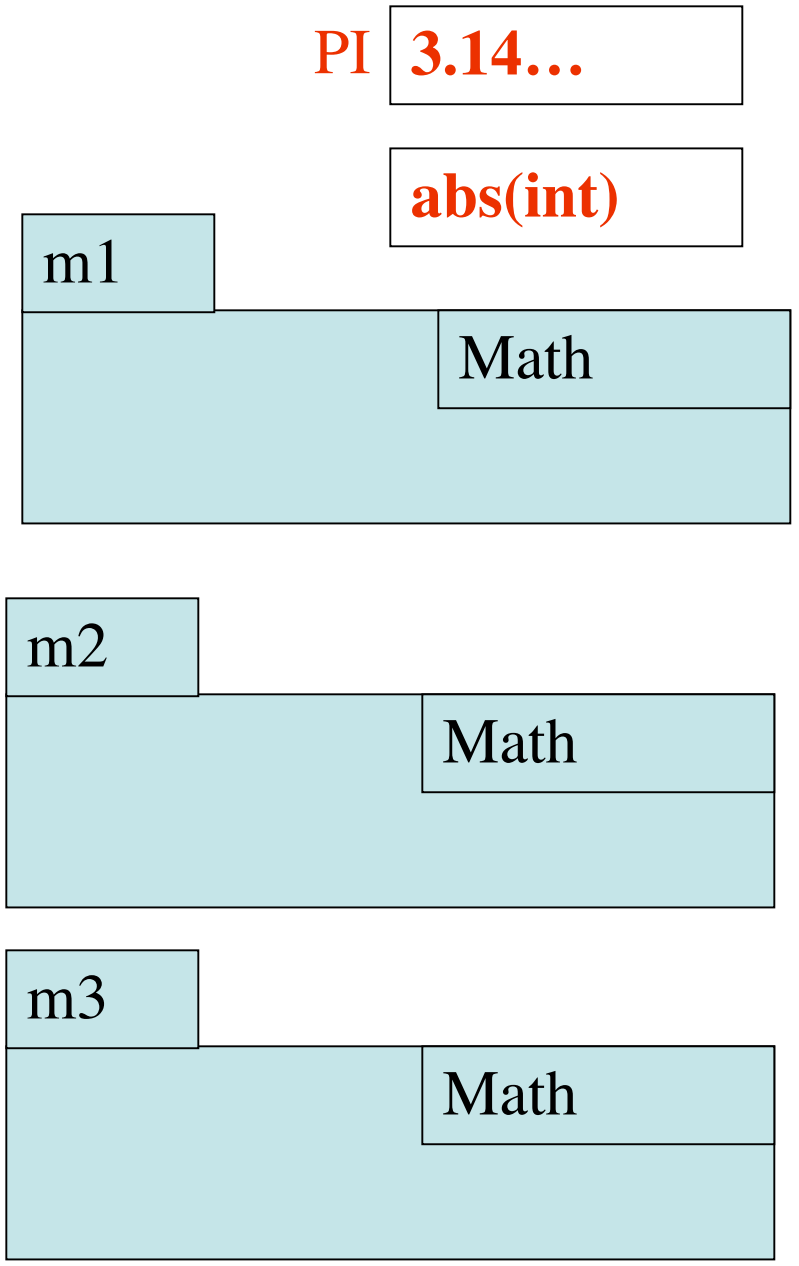
```
public class Math {
    /** = the absolute value of x
    public static int abs(int x) {
        if (x >= 0) return x;
        return – x;
    }




}
```



Five things in file drawer Math:

```
public class Math {
    /** = the absolute value of x
    public static int abs(int x) {
        if (x >= 0) return x;
        return – x;
    }
    // =  2* PI
    public double twoPi() {
         return 2*PI;
    }

    public static final double PI=
…;
}
```

means that PI can't be changed



Math is in package java.lang.

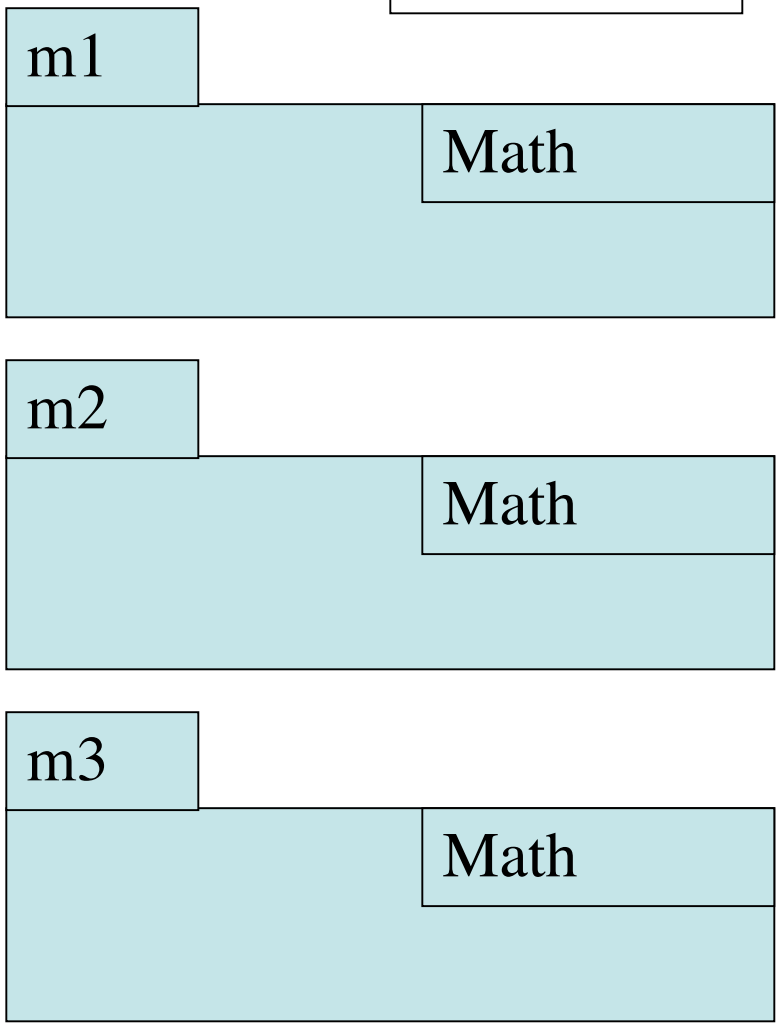
Referencing static entities

```
public class C {
    public void meth () {
        …
        // set x to absolute value of -3
        x= Math.abs(-3);

        // set y to PI
        y= Math.PI;

    }
}
```

Five things in file drawer Math:

PI **3.14…**

abs(int)



Methods: procedures, functions, constructors
Read: Section 2.1, 2.2

```
/** Print a, b, and their sum on one line */
public static void print(int a, int b) {
    System.out.println(a + "  "  + b + "  " + (a+b));

}
```

Definition: a ***parameter*** is a variable that is declared within the parentheses of the method header.

Parameters: a and b.

The comment is a ***specification*** of the method. It says WHAT the method does.

Method body: the "block" { … }

Methods: procedures, functions, constructors
Read: Section 2.1, 2.2

/** Print a, b, and their sum on one line */
public static void print(**int** a, **int** b) { …} ← header

When writing or understanding a call on a method, look only at the specification and not the method body.

What does this call do?

print(3+4, 6); ← **call, with arguments 3+4 and 6**

Print 3+4, 6, and their sum on one line.

Methods: procedures, functions, constructors
Read: Section 2.1, 2.2

/** Print a, b, and their sum on one line */
public static void print(**int** a, **int** b) { …} ← header

When writing or understanding a call on a method, look only at the specification and not the method body.

What does this call do?

print(3+4, 6);

Print 3+4, 6, and their sum on one line.

Parameters of the method: **a** and **b**

Arguments of the call: **3+4** and **6**