Announcements

- Midterm: Those who have a conflict please e-mail me at eposse@cs.mcgill.ca with
- your name
- the exam it conflicts with

Statements

- Variable declaration

  ```
  type variable;
  ```

- Assignment

  ```
  variable = expression;
  ```

- Method invocation

  ```
  objectreference.methodname(parameters);
  ```

  or

  ```
  classname.methodname(parameters);
  ```

- Conditional

- Loop

Conditionals

- The conditional statement is a statement used to make decisions: take different courses of action depending on some given condition
- There are several (syntactic) forms of conditionals
- The simplest form is:

```
if (boolean_expression) {
    list_of_statements;
}
```

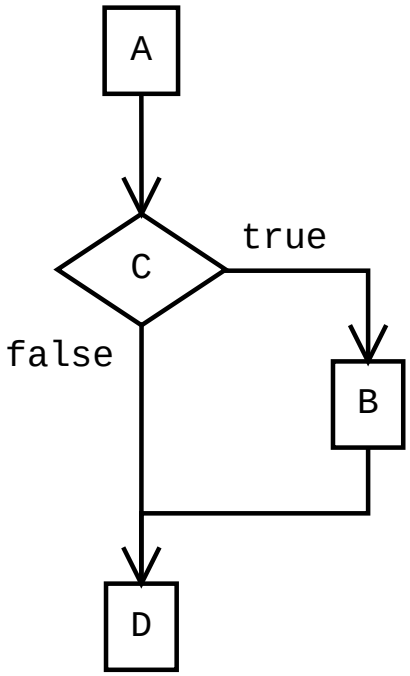
- For example:

```
if (winter && temperature >= -30.0f) {
    System.out.print("I'm going ");
    System.out.println("skiing!");
}
```

Conditionals

```
A;
if (C) {
    B;
}
D;
```

- Control flow diagram



Conditionals

- Conditionals with alternatives

```
if (boolean_expression) {
    list_of_statements_1;
}
else {
    list_of_statements_2;
}
```

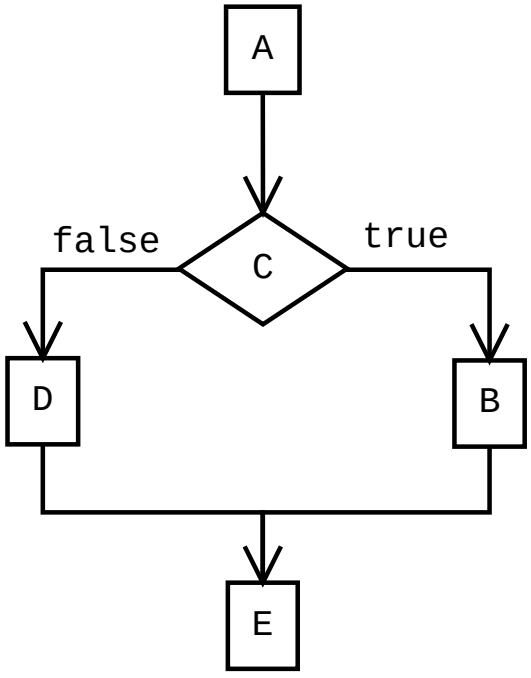
- For example:

```
if (!raining) {
    System.out.println("Go out");
}
else {
    System.out.println("Stay in");
}
```

Conditionals

```
A;
if (C) {
    B;
}
else {
    D;
}
E;
```

- Control flow diagram



Conditionals

- A conditional can take the form

```
if (condition)
    statement;
```

without braces, but only when a unique statement depends on the condition

```
if (a == 2)
  b = 4.0;
  c = true;
```

is the same as

```
if (a == 2) {
  b = 4.0;
}
c = true;
```

and not the same as

```
if (a == 2) {
  b = 4.0;
  c = true;
}
```

Conditionals

- The "dangling else" problem:

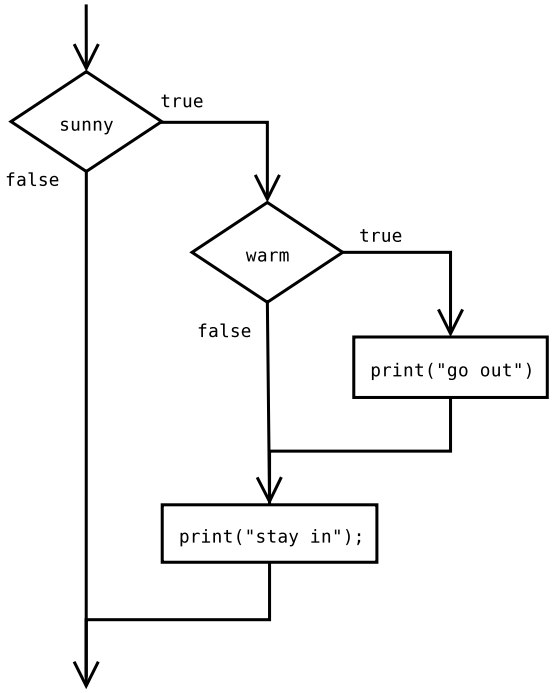
```
if (sunny)
    if (warm)
        System.out.print("go out");
else
    System.out.print("stay in");
```

If `sunny` is `false`, it will not print anything!

Conditionals

- It actually means:

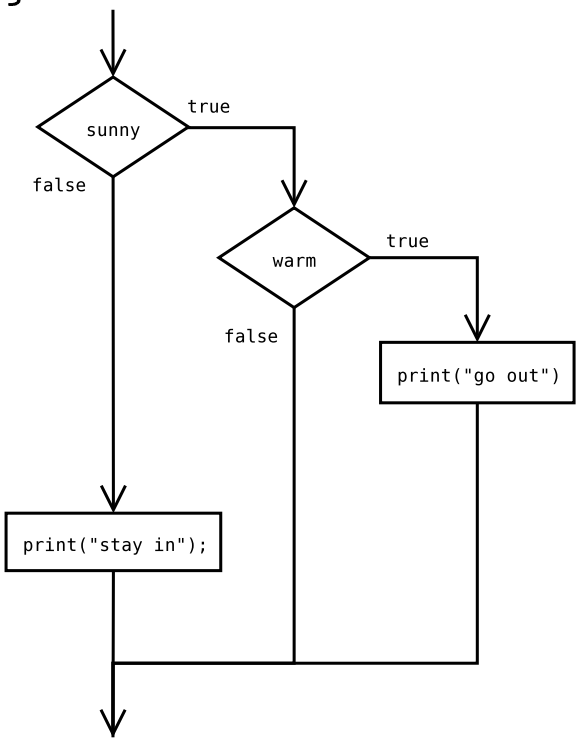
```
if (sunny) {
    if (warm) {
        System.out.print(“go out”);
    }
    else {
        System.out.print(“stay in”);
    }
}
```



Conditionals

and not

```
if (sunny) {
    if (warm) {
        System.out.print(“go out”);
    }
}
else {
    System.out.print(“stay in”);
}
```



Conditionals

- Order matters: if the value of a variable changes, then it will only affect conditionals which appear after the assignment

```
int a = 2;
String s = “hello”;

if (a == 3) {
    s = “bye”;
}
a = 3;
```

Conditionals

- Order matters: if the value of a variable changes, then it will only affect conditionals which appear after the assignment

```
int a = 2;
String s = “hello”;

if (a == 2) {
    s = “bye”;
    a = 5;
}
else {
    s = “again”;
}
```

Conditionals

- Order matters: if the value of a variable changes, then it will only affect conditionals which appear after the assignment

```
int a = 2;
String s = “start”;

if (a == 1) {
    s = “ready”;
}
if (a == 2) {
    s = “set”;
}
if (a == 3) {
    s = “go”;
}
a = 3;
```

Conditionals

- Order matters

```
boolean sunny = true, snow = false;
float temperature = -20.0f, windchill = -25.0f;

if (!sunny && temperature > -10.0f && !snow) {
    snow = true;
}
if (!snow && temperature - windchill < 10.0f) {
    sunny = false;
}
if (sunny && snow) {
    sunny = snow;
}
```

Conditionals

```
boolean x;

if (false) {
    x = true;
}
```

Conditionals

```
boolean x;

x = false;
if (false) {
    x = true;
}
```

Conditionals

```
boolean x;

x = true;
if (false) {
    x = true;
}
```

Conditionals

```
boolean x;

x = false;
if (x == false) {
    x = true;
}
```

Conditionals

```
boolean x;

x = false;
if (!x) {
    x = true;
}
```

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

```
P;
if (false) {
    Q;
}
R;
```

  is equivalent to

```
P;
R;
```

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

```
P;
if (C) {
    Q;
}
R;
```

  is equivalent to

```
P;
R;
```

  if the value of `C` is always `false`

Conditionals

- A few properties of conditionals:

```
System.out.println(“A”);
if (x == x + 1) {
    System.out.println(“B”);
}
System.out.println(“C”);
```

  is equivalent to

```
System.out.println("A");
System.out.println("C");
```

  because `x==x+1` is always `false`

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

```
P;
if (C) {
    Q;
}
R;
```

  is equivalent to

```
P;
Q;
R;
```

  if the value of `C` is always `true`

Conditionals

- A few properties of conditionals:

```
System.out.println(“A”);
if (x + 1 == x + 1) {
    System.out.println(“B”);
}
System.out.println(“C”);
```

  is equivalent to

```
System.out.println("A");
System.out.println("B");
System.out.println("C");
```

  because `x+1==x+1` is always `true`

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

```
P;
if (C == true) {
    Q;
}
R;
```

is equivalent to

```
P;
if (C) {
    Q;
}
R;
```

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

```
P;
if (C == false) {
    Q;
}
R;
```

is equivalent to

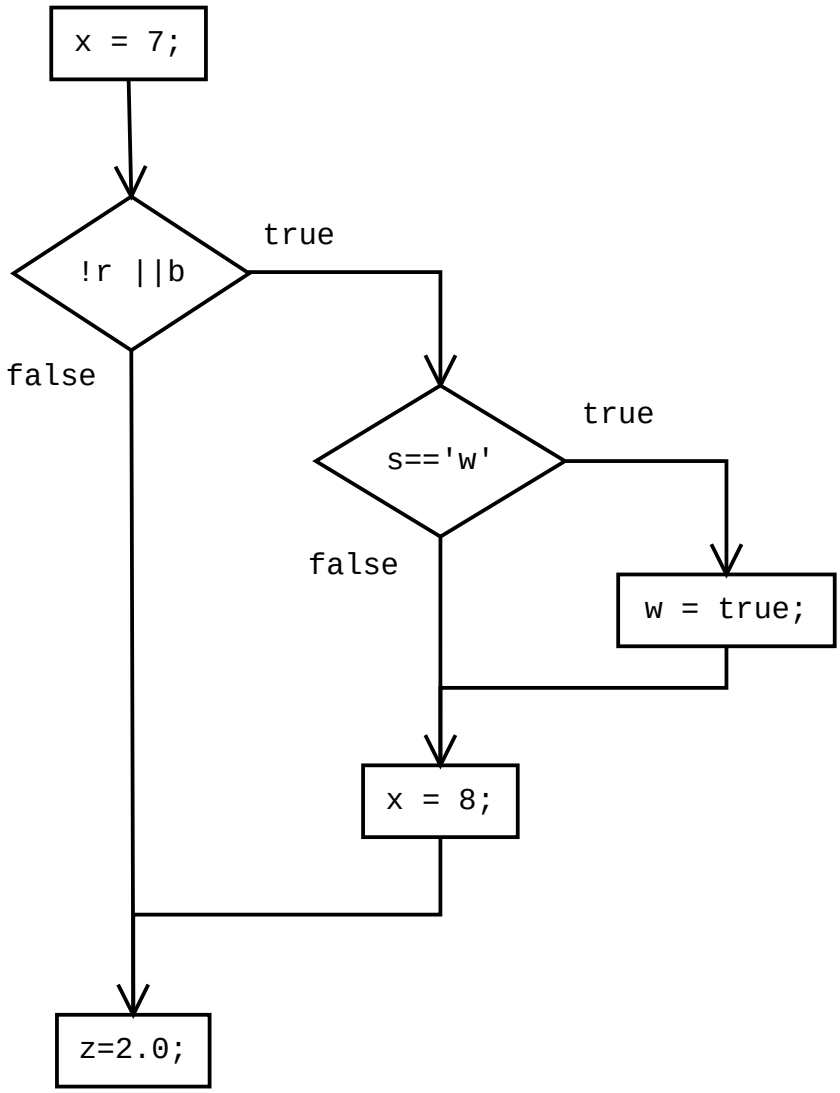
```
P;
if (!C) {
    Q;
}
R;
```

Conditionals

- Nested conditionals: Conditionals are statements and therefore they can go inside other conditionals

```
x = 7;
if (!r || b) {
    if (s ==‘w’) {
        w = true;
    }
    x = 8;
}
z = 2.0;
```

Conditionals

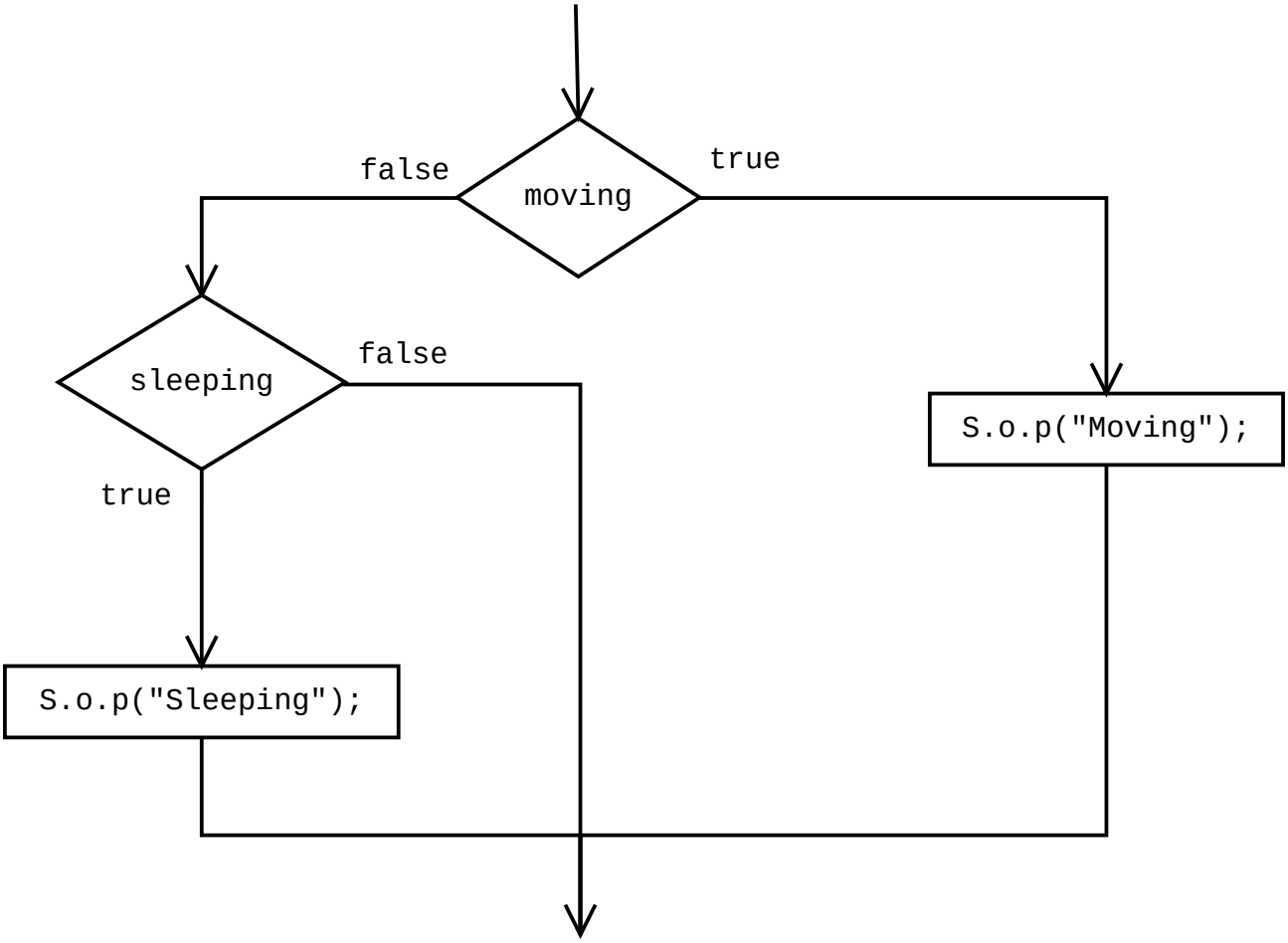


Conditionals

- Nested conditionals

```
if (moving) {
    System.out.println(“Moving”);
}
else {
    if (sleeping) {
        System.out.println(“Sleeping”);
    }
}
```

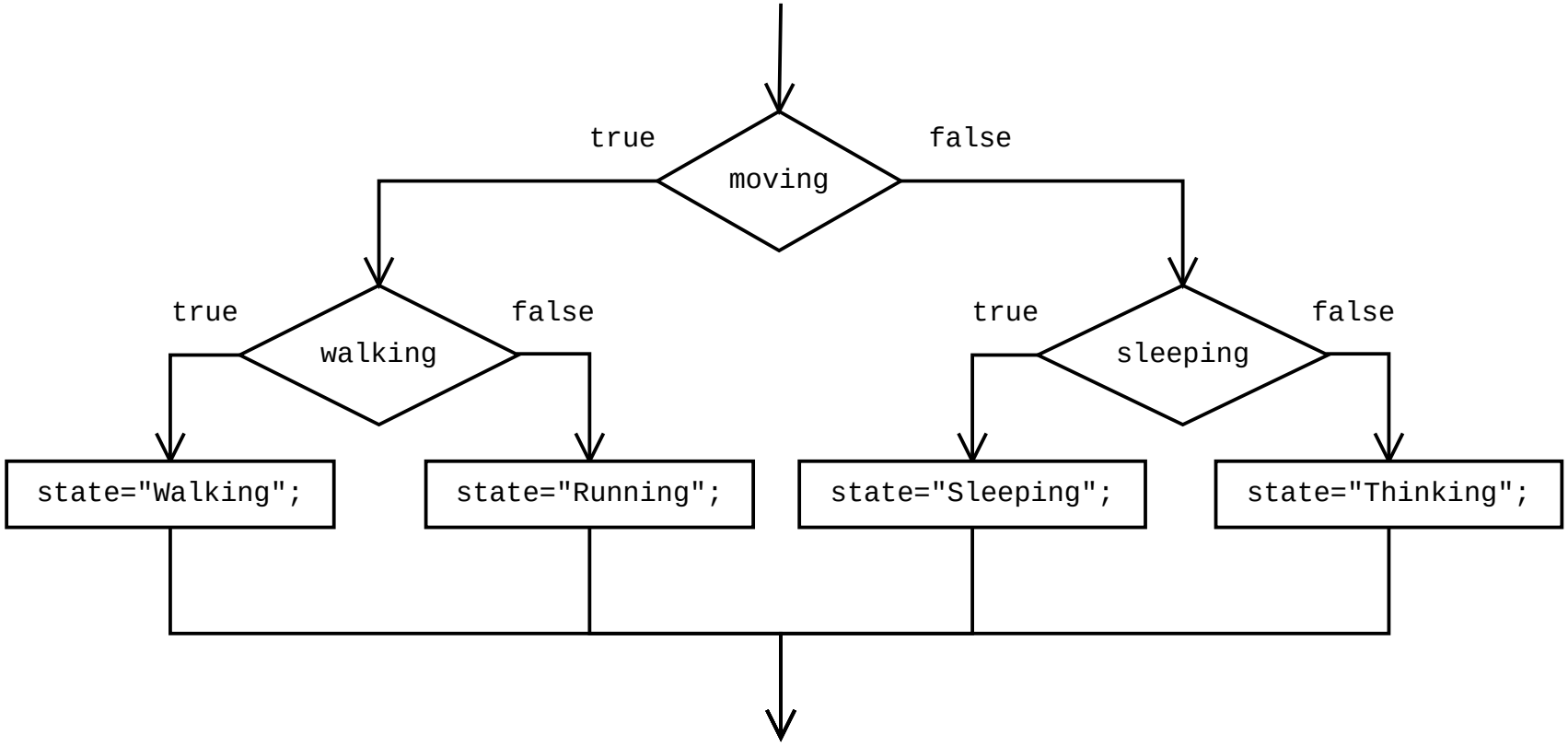
Conditionals



Conditionals

```
if (moving) {
    if (walking) {
        state = “Walking”;
    }
    else {
        state = “Running”;
    }
}
else {
    if (sleeping) {
        state = “Sleeping”;
    }
    else {
        state = “Thinking”;
    }
}
```

Decision tree



Decision trees

- Decision trees as a design technique
- Problem: Given three numbers, determine which one is the smallest
- Analysis:
  - Input: three numbers a, b and c
  - Output: a number m, which is the smallest among a, b and c
  - Definitions:
    * A number m is the *smallest* among three numbers a, b and c if m is one of the three numbers (i.e. $m = a$, $m = b$, or $m = c$), and it satisfies the condition that it is less or equal than the three numbers (i.e. $m \leqslant a$, $m \leqslant b$, and $m \leqslant c$) Note that these are not strict inequalities.
  - Open issues: what kind of numbers?

Analysis

- If there are open issues we can make assumptions as long as:
  - they are consistent with all aspects of the problem,
  - they make sense, and
  - they do not impose restrictions which modify the problem. (For instance in this case we should not assume that the numbers are all different.)

- Assumptions:
  - Numbers can always be compared (not true if we are not dealing with numbers)

- It is often useful to state the obvious
  - In this example, it is "obvious" that m must be one of the three given numbers, but this is crucial, because we could have a very easy solution: to return a number smaller than all of them. The problem is that this would not be solving the original problem.

Design

- First alternative: consider all possibilities:

1. If $a \leqslant b$ and $b \leqslant c$ then let m be a
2. If $a \leqslant c$ and $c \leqslant b$ then let m be a
3. If $b \leqslant a$ and $a \leqslant c$ then let m be b
4. If $b \leqslant c$ and $c \leqslant a$ then let m be b
5. If $c \leqslant a$ and $a \leqslant b$ then let m be c
6. If $c \leqslant b$ and $b \leqslant a$ then let m be c

- This solution is correct. It covers all possibilities, but it requires 12 comparisons in the worst case. It is not a very smart solution, and it does not scale well.

Implementation

```
import cs1.Keyboard;
public class SmallestFinder {
  public static void main(String[] args)
  {
    double a, b, c, m;

    System.out.print(“Enter the first number:”);
    a = Keyboard.readDouble();
    System.out.print(“Enter the second number:”);
    b = Keyboard.readDouble();
    System.out.print(“Enter the third number:”);
    c = Keyboard.readDouble();

    // Continues below ...
```

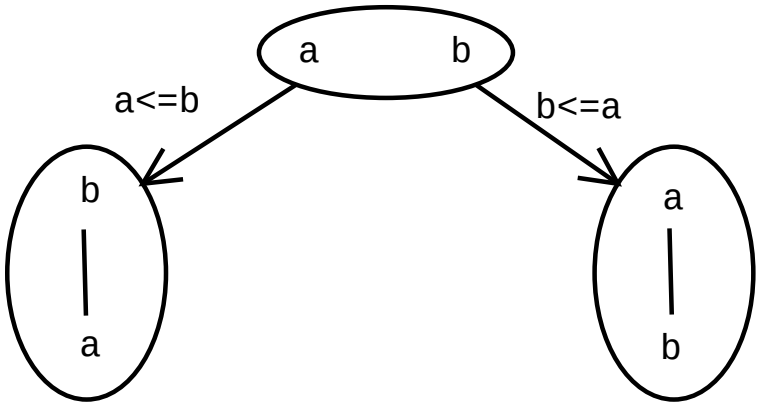
Implementation

```
    if (a <= b && b <= c) m = a;
    if (a <= c && c <= b) m = a;
    if (b <= a && a <= c) m = b;
    if (b <= c && c <= a) m = b;
    if (c <= a && a <= b) m = c;
    if (c <= b && b <= a) m = c;

    System.out.println("The smallest is " + m);

  }  // End of main method
} // End of SmallestFinder class
```

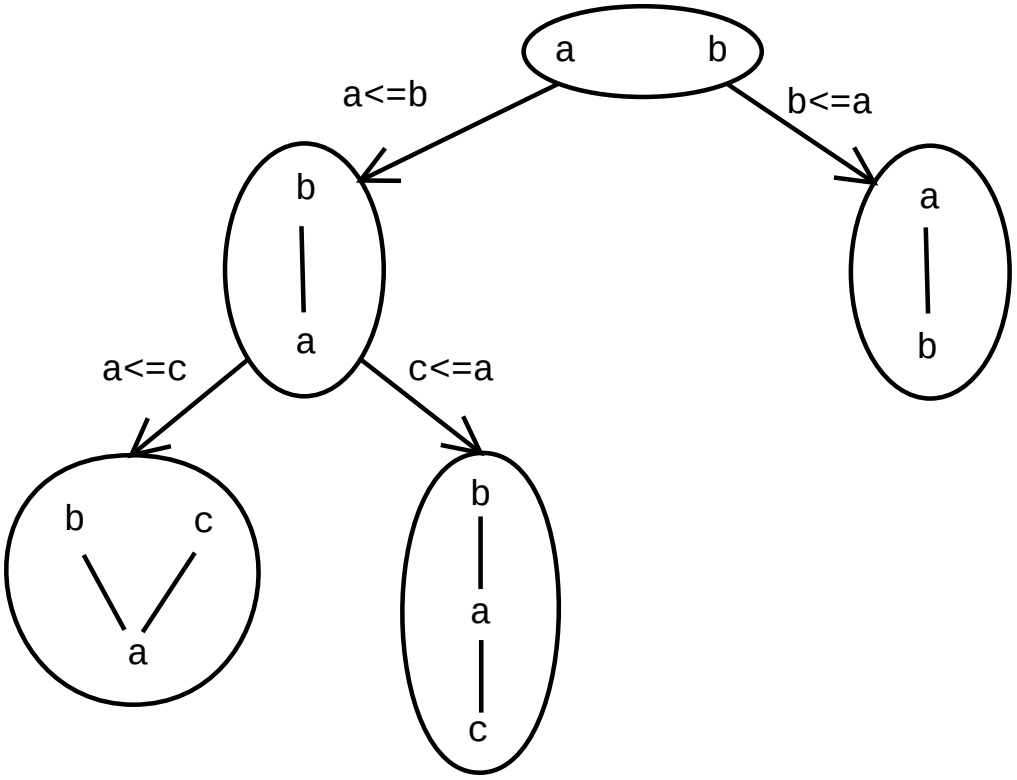
Design

- Second alternative: use decision trees to rule out possibilities, and take into account what is already known.

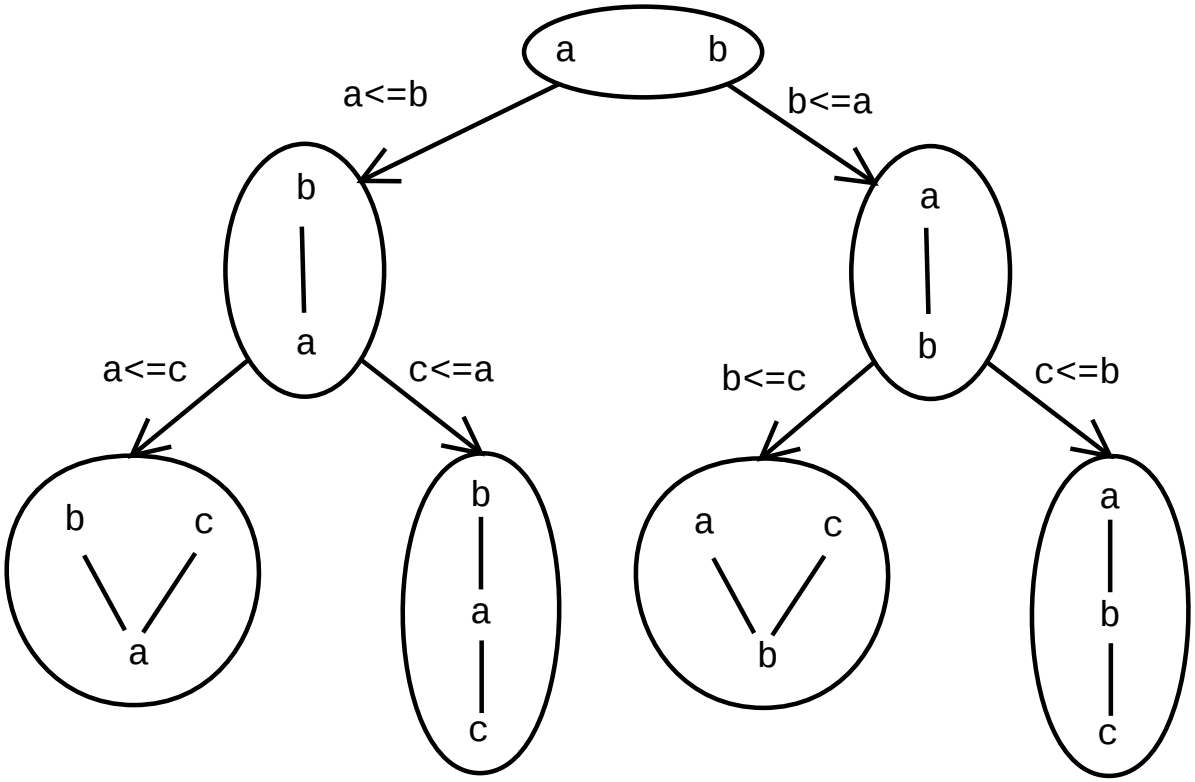
- Analysis: study the possible relationships between the data involved in the problem



Design

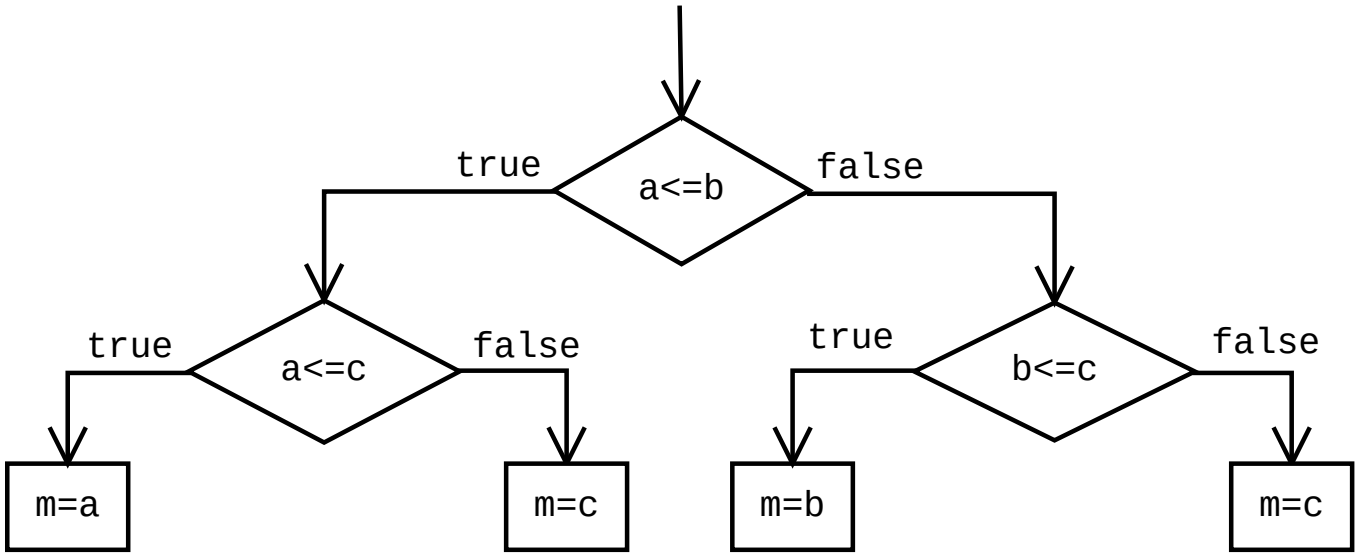


Design



Design

Decision tree



Implementation

```
import cs1.Keyboard;
public class SmallestFinder2 {
  public static void main(String[] args)
  {
    double a, b, c, m;

    System.out.print(“Enter the first number:”);
    a = Keyboard.readDouble();
    System.out.print(“Enter the second number:”);
    b = Keyboard.readDouble();
    System.out.print(“Enter the third number:”);
    c = Keyboard.readDouble();

    // Continues below ...
```

Implementation

```
    if (a <= b) {
      if (a <= c) {
        m = a;
      }
      else {
        m = c;
      }
    }
    else {
      if (b <= c) {
        m = b;
      }
      else {
        m = c;
      }
    }
    System.out.println(“The smallest is ” + m);
  } // End of main method
} // End of SmallestFinder2 class
```

Properties of conditionals

- In the following, `C` is any boolean expression, `P`, `Q`, `R`, and `S` are any list of statements.

```
P;
if (C) {
  Q;
}
else {
  R;
}
S;
```

is equivalent to

```
P;
if (!C) {
  R;
}
else {
  Q;
}
S;
```

Properties of conditionals

- In the following, `C`, `D` are any boolean expressions, `P`, `Q`, and `R` are any list of statements.

```
P;
if (C && D) {
  Q;
}
R;
```

is equivalent to

```
P;
if (C) {
  if (D) {
    Q;
  }
}
R;
```

Properties of conditionals

- In the following, `C`, `D` are any boolean expressions, `P`, `Q`, and `R` are any list of statements.

```
P;
if (C || D) {
  Q;
}
R;
```

is equivalent to

```
P;
if (C) {
  Q;
}
else {
  if (D) {
     Q;
  }
}
R;
```

The end