Flow of Control Statements

• Comparing objects for equality
  – Example from Assignment 3

• Simple for loops
  – Iterated execution

• More on if statements
  – Blocks within ifs
  – Break statement
How to do value comparisons?

- For primitive types we use `==` to check if two values are equivalent
  ```java
  int a, b; ... if (a == b)
  ```

- For objects, we define an `equals()` method in each class which will check that all corresponding pairs of instance variables of the two objects have the same value
  ```java
  String p1;...if (p1.equals("inches"))...
  ```

Objects must be instance of the same class to be equal.
**Equals versus ==**

- **Identical twins:**
  - equal (they look alike) but not == (they are not the same person)

- **Two lectures in cs111 taught by same instructor:** equal (same lecture notes) but not == (can’t do same delivery each time)

- **MORAL:**
  - Use == for comparing primitive values
  - Use `equals()` for comparing objects when you want a comparison of values
Assumptions

• Have a class called Point
  – Instance variables: double x, double y

• Constructor
  – Point(double x, double y)
    – Creates new point with these x and y coordinates

• Need to tell if two Point objects are really the same point, meaning they represent the same (x,y) coordinate on the plane.
Example - Points in a Plane

Point p1 = new Point(0., 0.);
//make reference p2 refer to same object as p1
Point p2 = p1;

//create second object with same x,y coordinates as p1
Point p3 = new Point(0., 0.);
Example

Point p4 = new Point(1.0, 0.0);

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>p2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>p3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>p4</td>
<td>1.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

if (p1 == p3) //references to different objects; false
if (p1.equals(p3)) //instance var value check; true
if (p1 == p2) //references to same object; true
if (p1 == p4) //references to different objects; false
if (p1.equals(p4)) //instance var value check; false
Example

(p1.x == p3.x) && (p1.y == p3.y) 
but 
p1 != p3
Iterated Execution

- Need for repeating a sequence of statements
  - e.g., in Nim game to explore all Nim(2) games which start with 1 - 14 stones

\[\text{for-loop}\rightarrow\]
\[\text{for}\ (\text{<start> ; <check> ; <update>} )\ \text{<block>}\]
\[\text{<block}> \rightarrow\{\ \text{<statements>}\ }\ \} | \ \text{<statement>}\]
\[\text{<statements}> \rightarrow \text{<statement>} | \]
\[\text{<statements>} \quad \text{<statement>}\]

\[
\text{for} \ (i=0; \ i<10; \ i++) \ \text{sum} \ += \ i; \\
\text{for} \ (j=15; \ j>0; \ j--)\{ \\
\quad \text{squares} \ += \ j*j; \\
\}
\]
For Loop

• Loop variable
  – Value initialized in <start>
  – Value changed on each iteration in <update>
  – Value checked for stopping iteration in <check>
  
  \[\text{for}(\text{int } i = 1; \ i < \ n; \ \ i++)\]

• Loop execution
  
  \[<\text{start}> \ <\text{check}> \ <\text{block}> \ <\text{update}>\]
  \[<\text{check}> \ <\text{block}> \ <\text{update}>\]
  \[<\text{check}> \ <\text{block}> \ <\text{update}> \ ... \ <\text{check}>\]
Nim Loop Example

```java
public static void main (String[ ] arg) {
    // play game with each of 14 piles of stones
    // need to repeat what did before for each pile
    // for loop - (initialization; test; increment)
    for (int i=1;  i < 15; i++) {
        //create new game object initialized with i stones
        NimState st = new NimState(i);
        System.out.print(i +": ");
        //test if game is winnable by first player
        if(st.win()) System.out.println ("win, remove " + st.move());
        else System.out.println("lose, remove " + st.move());
    }
}
```

see NimState-loop.java
UStime Loop Example

public static void main(String[] args){
    for (int h = 1; h < 13; h++) {
        UStime z = new UStime (h, 0);
        System.out.print(z +“ in NYC is “);
        System.out.println((z.cvrtCentral( )) +
            “ in Chicago” );
    }
}

from main method in UStime-loop.java

• Compare to temp conversion program in Bishop, p 61.

Barbara G. Ryder © Spring 1998
What is printed?

1 hours and 0 minutes in NYC is 12 hours and 0 minutes in Chicago
2 hours and 0 minutes in NYC is 1 hours and 0 minutes in Chicago
3 hours and 0 minutes in NYC is 2 hours and 0 minutes in Chicago
4 hours and 0 minutes in NYC is 3 hours and 0 minutes in Chicago
5 hours and 0 minutes in NYC is 4 hours and 0 minutes in Chicago
6 hours and 0 minutes in NYC is 5 hours and 0 minutes in Chicago
7 hours and 0 minutes in NYC is 6 hours and 0 minutes in Chicago
8 hours and 0 minutes in NYC is 7 hours and 0 minutes in Chicago
9 hours and 0 minutes in NYC is 8 hours and 0 minutes in Chicago
10 hours and 0 minutes in NYC is 9 hours and 0 minutes in Chicago
11 hours and 0 minutes in NYC is 10 hours and 0 minutes in Chicago
12 hours and 0 minutes in NYC is 11 hours and 0 minutes in Chicago
For Loops

- Loop variable usually changes value by simple increment or decrement
  
  ```
  for (int i = 10; i > 0; i -= 2 ){
  }
  ```

- Often used to iterate loop variable over a range of values
Nested Loop Example

for (int h = 1; h < 13; h++) {
    for (int m = 0; m < 60; m++) {
        USTime z = new USTime (h, m);
        System.out.print(z + " in NYC is ");
        System.out.println((z.cvrtCentral() ) + " in Chicago");
    }
}

loops through all minutes for all possible hours

from USTime-nestloop.java
If Statement as Selection

- If statement allows selection between two alternative directions for flow of control in program - true (then clause) and false (else clause)
- Often used in conjunction with a loop
- Can nest if statements for more complex conditions
- Can group sequence of statements to be performed conditionally in a block
Example - If in a Loop

```java
class Summation extends Object{
    static final int limit = 30; // Java constant class var
    public static void main(String[] args) {
        int sumeven=0;
        int sumodd =0;
        for (int i=0; i<=limit; i++)
            if ((i%2)==0) sumeven += i;
            else sumodd += i;
        System.out.println("sum of even numbers from 0 to" + limit + " is " + sumeven);
        System.out.println("sum of odd numbers from 0 to" + limit + " is " + sumodd);
    }
}
```
Example - Blocks in If Stmts

class Summation extends Object{
    static final int limit = 30;
    public static void main(String[] args) {
        // sums all even nos and odd nos <= limit, separately
        int sumeven=0; int sumodd =0;
        int evencnt = 0; int oddcnt = 0;
        for (int i=0; i<=limit; i++)
            if ((i%2)==0){sumeven += i;
                evencnt +=1;}
            else {sumodd += i;
                oddcnt +=1;}
        System.out.println("sum of" + evencnt + " even numbers from 0 to " + limit + " is " + sumeven);
        System.out.println("sum of" + oddcnt + " odd numbers from 0 to " + limit+ " is " + sumodd);
    }
}

from sum.java
Blocks in If Statements

\[\text{<block>} \rightarrow \{ \text{<statements>} \} \mid \text{<statement>}\]

- Old if statement BNF:
  \[\text{<if\_stmt>} \rightarrow \text{if (} \text{<condition>} \text{)} \text{<statement>}\]
  \[\quad [ \text{else } \text{<statement>} ]\]
- Replaced by new if statement BNF:
  \[\text{<if\_stmt>} \rightarrow \text{if (} \text{<condition>} \text{)} \text{<block>}\]
  \[\quad [ \text{else } \text{<block>} ]\]
Nested If Statements

- Check on related conditions or membership in a range of values

if (<cond1>) <block>  // <cond1> is true
  else if (<cond2>)  <block>
    // !<cond1> && <cond2> is true
  else if (<cond3>) <block>
    // !<cond1> && !<cond2> && <cond3>  // is true
  else <block>  // !<cond1> && !<cond2> && !<cond3>  // is true
Possible Ambiguity in Meaning

if (x>0) if (y<-1) y += 2; else y +=3;

is this generated by this rule?

<if_stmt> → if (<condition>) <statement>

or this rule?

<if_stmt> → if (<condition>) <statement>
else <statement>

Under what conditions is y incremented by 3, when x<=0 OR when x>0 && y>=-1?
Possible Ambiguity -2

if (x>0) if (y<-1) y += 2; else y +=3;
in Java means: if (<condition>) <statement>

where <statement> matches the “inner” <if-stmt>:

if (y<-1) y += 2; else y += 3; matches
if (<condition>) <statement> else <statement>

where these expand to <assign-stmt> and
<assign-stmt>, respectively