Iteration

Chris La Pilla
CompSci 6
Summer 2007

Iteration
• **Iteration**: execute code repeatedly until a goal is reached
• Java has three types of statements for iteration:
  – while loops
  – do-while loops
  – for loops

while loops
• A while loop executes a statement or block repeatedly as long as a condition is true
• Syntax:
  ```java
  while (condition)
  statement
  ```
• Can loop multiple statements by enclosing them in curly braces `{ }`
• If *condition* is initially false, the loop body is never executed

while loop example
• Computing number of years until a savings account reaches a certain balance:
  ```java
  while (balance < targetBalance) {
    years++;
    double interest = balance * rate;
    balance = balance + interest;
  }
  ```

do-while loops
• Works the same as a while loop, except the condition is checked at the end of the loop body
• This means the loop always executes at least once
• Syntax:
  ```java
  do
  statement
  while (condition);
  ```
• Notice that there is a semi-colon at the end of the while line

do-while loop example
• What does this do?
  ```java
  int num = 0;
  do{
    System.out.print("Enter int: ");
    num = scanner.nextInt();
  } while (num <= 0);
  ```
do-while loop example

- A do-while loop can be written as a while loop, but you need boolean variables:

```java
boolean done = false;
while (!done){
    System.out.print("Enter int: ");
    num = scanner.nextInt();
    if (num > 0){
        done = true;
    }
}
```

for loops

- There is a very common use of the while loop that follows this pattern:

```java
int i = start;
while (i <= end){
    ... 
    i++;
}
```

- Used to execute loop a fixed number of times

for loops

- for loops are used especially for this purpose:

```java
for (int i = start; i <= end; i++){
    ... 
}
```

- In general, the syntax is:

```java
for (initialization; condition; update)
statement
```

for loop examples

- Adding up first N integers:

```java
int sum = 0;
for (int i = 1; i <= N; i++)
    sum = sum + i;
```

- Printing each character in a String on a separate line:

```java
for (int i = 0; i < str.length(); i++)
    System.out.println(str.charAt(i));
```

Nested Loops

- Loops can be nested inside each other
- When nesting for loops, make sure the loop variables are different:

```java
for (int i = 0; i < M; i++){
    for (int j = 0; j < N; j++){
        ...
    }
}
```

```

Nested Loops

- The initialization, condition, or update components of the inner loop can depend on the outer loop variable:

```java
for(int i = 1; i <= 3; i++){
    for(int j = 1; j <= i; j++){
        System.out.print("*");
    }
    System.out.println(); //print line break
}
```

- Above example prints:

```
* 
** 
*** 
```
Common loop bugs

- All loops repeat until their condition is no longer true
- If condition never becomes false, program gets stuck in an infinite loop (what kind of error is this?)
- In Eclipse, if your program gets stuck in an infinite loop, you can force it to stop by clicking the terminate button (red square) in the console window

Off-by-one bug

- A for loop of this form
  
  `for (i = start; i < end; i++)`
  
  executes `end - start` times

- A for loop of this form
  
  `for (i = start; i <= end; i++)`
  
  executes `end - start + 1` times

- A common error is for the number of loop executions to be off by one. e.g.

```java
//Execute the loop 10 times:
for(i = 0; i <= 10; i++) //logic error!
...
```

Off-by-one bug

- When you want to execute a loop \( N \) times, use either:
  
  for (i = 0; i < N; i++)
  
  or
  
  for (i = 1; i <= N; i++)