CS107
Introduction to Computer Science

Loops

Instructions

Pseudocode

- Assign values to variables using basic arithmetic operations
```
x = 3
y = x/10
z = x +25
```
- Get input from user
```
Get x
```
- Print to user (screen)
```
Print "The value of x is " x
```
- Conditionals
```
if (a > b)
        print "largest is" a
else print "largest is" c
```
- **Loops**

Java

- Assign values to variables using basic arithmetic operations
```
int x, y;
x = 3;
y = x/10;
```
- Get input from user
```
a = r.readInt();
r.readLine();
```
- Print to user (screen)
```
System.out.print("x is " + x);
```
- Conditionals
```
if (a > b)  {
        ...
}  else {
        ...
}
```
- **Loops**

Conditions in If instructions

```
if (condition) {
   //these instr are executed if the condition is
      true
 } else {
 //these instr are executed if the condition is true
 }
```

Conditions are built using
- Comparison operators

==	equal
!=	not equal
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal

- Logical operators

&&	and
||	or

•Examples:
```
 //assume x, y two variables
if  ( (x <= y) && (x > 20)) ...
if ((x == 10) | | (y != 15))
```

Exercise

Write a program that asks the user for three numbers and prints out the largest. For example:

```
Enter first number: 10
Enter the second number: 25
Enter the third number: 5
The largest is 25.
Goodbye.
```

Comments on If instructions

These are some bugs that you may come across…

-
  ```
  int x = 10, y = 20;
  if (x < y)
          System.out.println(x);
          x = 0;
          y = 100;
  System.out.print(x);
  ```

-
  ```
  int x, y;
  if (x < y);
        System.out.println("x is smaller);
   System.out.println("Goodbye");
  ```

Loop instructions

- A loop instruction specifies a group of statements that may be done several times (repeated):

```
while (condition) {
        //statements to be repeated
}
```

- How does this work?
  - Condition is evaluated
  - If it is false than the loop terminates and the next instruction to be executed will be the instruction immediately following the loop
  - If it is true, then the algorithm executes the *instructions to be repeated* in order, one by one

Example

- What does this algorithm do?

```
int i;
i = 1;
while (i  <= 100) {
            System.out.println("i= " + i);
            i = i + 1;
}
```

- Note the indentation

Example

- What does this algorithm do?

```
int count, square;
count = 1;
while (count  <= 10) {
            square = count *count;
            System.out.println("square= + square + " count= " + count);
            count = count + 1;
}
```

- Note the indentation

Computing the sum 1+2+….+n

Write an algorithm which reads a positive integer from the user and computes the sum of all positive integers smaller than or equal to the number entered by the user.

Example: if the user enters 10, the algorithm should compute 1+2+3+...+10

```
Please enter a positive number:  10
The sum of all integers up to 10 is: 55
Goodbye.
```

Gauss formula

- We can actually find a formula for 1 + 2 + ……. + n

Gauss noticed that

- $1 + n = n+1$
- $2 + (n-1) = n+1$
- ….

$\Longrightarrow 1 + 2 + \ldots + (n-1) + n = n(n+1)/2$

Comments

- An algorithm is not unique!
- There are many ways to solve a problem
- Moreover, given a certain way to solve a problem, there are many ways to implement that into Java!
- Programming style:
  - Give variables meaningful names
  - Write explanations/comments of what your algorithm does
  - Separate the logical blocks of your program with spaces
  - Break long lines
  - **Keep it simple**

Exercises

Given a number n from the user, write an algorithm..

- To compute the sum of all numbers strictly smaller than n
- To compute the sum of all even numbers <= n
- To compute the sum of all odd numbers <= n
- To compute the product of all numbers <= n (starting at 1)

Exercise

Write an algorithm that asks the user for a positive number. Assume the user is dumb (or stubborn) and enters a negative number. The program should keep asking the user for a positive number until the number entered by the user is positive. For example:

```
Enter a positive number: -3
Sorry, -3 is not positive.
Enter a positive number: -10
Sorry, -10 is not positive.
Enter a positive number: -2
Sorry, -2 is not positive.
Enter a positive number: 10
Finally. Goodbye.
```

Exercise

- Modify your previous algorithm so that the user keeps track of how many times the user enters a “wrong” number. Make it print this at the end.

- Now make it terminate if the user does not enter a “right” number within 10 attempts.