Lecture 13
The for-statement

- The most common form of the for-statement is as follows:

```for ( var = START_VALUE ; var <= STOP_VALUE ; var = var + INCR )
{
    /* for-body (statements) */
}
```

**Meaning:**

- The variable `var` will take on the following values:
  - `START_VALUE`
  - `START_VALUE + INCR`
  - `START_VALUE + 2*INCR`
  - ...
  - Up to the largest value that is ≤ `STOP_VALUE`

- For each of the values, the statements in the for-body are executed

- In other words:

```repeat ( for var = START_VALUE, 
          START_VALUE+INCR, 
          START_VALUE+2*INCR, 
          ...,
          STOP_VALUE )
execute statements in the for-body
(one time for each value in the list);```
Example 1:

```java
public class For01
{
    public static void main(String[] args)
    {
        int a;

        // Example "for-statement"
        for ( a = 1 ; a <= 10 ; a = a + 1 )
        {
            System.out.println(a); // Print a
        }
        System.out.println("Done");
    }
}
```

Output:

1
2 (1+1×1)
3 (1+2×1)
4 (1+3×1)
5 (1+4×1)
6 (1+5×1)
7 (1+6×1)
8 (1+7×1)
9 (1+8×1)
10 (1+9×1)
Done

Notice that in the for-statement:

- `a = 1` specifies the **starting value**
- `a <= 10` specifies **when the for-loop will stop**
- `a = a + 1` specifies **how the variable `a` will change** each time through the loop
The *general syntax* of the *for-statement* is as follows:

```
for (init-expression; loop-cont-condition; incr-expression)
  ONE-statement
```

*Body of the for-loop*

A typical *for-loop* looks like this:

```
for (init-expression; loop-cont-condition; incr-expression)
{
  statement1
  statement2
  ....
}
```

*Body of the for-loop*
Flow chart representing the for-statement:

- init-expression;
- loop-cont-condition
  - true
    - statement1
    - statement2
    - ....
  - false (terminates)
- incr-expression;

Execution repeats after body is done
Example:
```
for ( a = 1 ; a <= 10 ; a++ )
{
    System.out.println(a);
}
System.out.println("Done");
```

Flow chart of this program:
Exact Pattern Matching: the *brute force* approach

- **Exact String Matching:**
  - **Given:**
    - A text string $T = t_0 \ t_1 \ t_2 \ ... \ t_{n-1}$\hspace{1cm} length($T$) = $n$
    - A pattern string $P = p_0 \ p_1 \ p_2 \ ... \ p_{m-1}$\hspace{1cm} length($P$) = $m$
  - **Problem:**
    - Determine all the occurrences of the pattern $P$ in the text $T$
  - **Example:**
    - $T = \text{acgtagatactaggatacca}$
    - $P = \text{gata}$
    - Solution:
      - $\text{acgtagatactaggatacca}$
• The **brute force** approach

  ○ **Reminder:** the Brute Force method

    • Generate **every possible outcome**
    • For each outcome, test if the outcome is a **valid solution**

---

○ **Example:**

    ```plaintext
    T = acgtagatactaggatgcca
    P = gata
    
    Try:
    acgtagatactaggatgcca
gata
    acgtagatactaggatgcca
gata
    acgtagatactaggatgcca
gata
    acgtagatactaggatgcca
gata
    and so on...
    ```
import java.util.Scanner;

public class StringMatching1
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);

        String T, P, cand;
        int pos;

        System.out.print("Enter a text string T: ");
        T = in.nextLine();
        System.out.print("Enter a pattern string P: ");
        P = in.nextLine();

        for (pos = 0; pos <= T.length() - P.length(); pos++)
        {
            cand = T.substring(pos, pos + P.length()); // Try to match this...

            if (P.equals(cand))
            {
                System.out.println("Found pattern at position: " + pos); // Found
            }
        }
    }
}
Effect of the *break*-statement on a *for*-statement

- The execution of the *break* statement illustrated with a flow chart:

```
For-statement

init-expression;

loop-cont-condition

true

Body of for-statement

break;

break statement will cause this jump

incr-expression;
```
Effect of the `continue`-statement on a `for`-statement

- The execution of the `continue` statement illustrated with a flow chart:

```
init-expression;

loop-cont-condition

false (terminates)

true

Body of for-statement

continue;

continue statement will cause this jump

Execution repeats

incr-expression;
```
• Quizz!